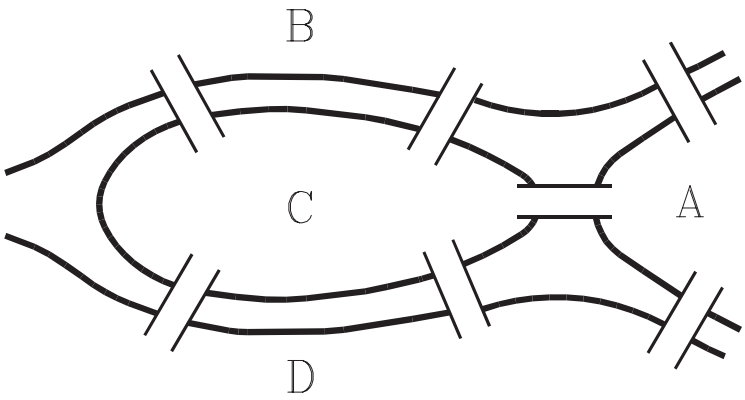
Prof. Dr.-Ing. Firoz Kaderali
Prof. Dr. rer. nat. Werner Poguntke

# Graphen, Algorithmen und Netze



# Vorwort zum Kurs

Die Graphentheorie ist heute ein wichtiges Hilfsmittel beim Studium komplexer Probleme in verschiedenen Wissenschaften wie auch in direkten Anwendungsbereichen.
Der universelle Charakter der Graphentheorie hat seinen Ursprung in der Einfachheit der Struktur von Graphen: die Konzepte und Ergebnisse der Graphentheorie sind immer dann anwendbar, wenn ein System zu modellieren ist, in dem Paare von Objekten in einer Beziehung stehen können. Die strukturelle Einfachheit (und damit auch Anschaulichkeit) zusammen mit dem interdisziplinären Charakter geben der Graphentheorie viel von ihrem besonderen Reiz. Bei einer Modellierung durch Graphen bleiben natürlich stets (mitunter wichtige) Aspekte des zu untersuchenden Systems unberücksichtigt, weshalb die erzielten Ergebnisse mit Zurückhaltung interpretiert werden müssen. Dies dürfte besonders für sozialwissenschaftliche Anwendungen der Graphentheorie zutreffen.

Historisch hat die Graphentheorie viele Ursprünge, die oft aus Rätseln oder Spielen erwachsen sind. Viele Konzepte und Ergebnisse sind dabei mehrfach eingeführt bzw. erzielt worden. Einige markante Stationen sollen hier aufgeführt werden:

1737 Euler löst das Königsberger Brückenproblem.

1847 Kirchhoff verwendet graphentheoretische überlegungen zur Analyse elektrischer Netzwerke.

1852 Guthrie wirft gegenüber deMorgen die Vierfarbenvermutung als Problem auf, das 1878/79 von Cayley noch einmal öffentlich gestellt wird.

1857 Cayley untersucht die Isomeren gesättigter Kohlenwasserstoffe und bestimmt die Anzahl der Gerüste vollständiger Graphen.

1859 Hamilton erfindet ein Spiel, bei dem entlang der Kanten eines regulären Dodekaeders eine geschlossene Linie zu finden ist, die jede Ecke genau einmal berührt.

1890 Heawood beweist, dass jeder planare Graph $5$-färbbar ist.

1927 Menger beweist, dass in jedem zusammenhängenden Graphen die Mindestanzahl von Ecken, deren Wegnahme zwei nicht benachbarte Punkte unverbindbar macht, gleich der Maximalzahl eckendisjunkter Wege zwischen diesen Punkten ist.

1930 Kuratowski beweist, dass ein Graph genau dann planar ist, wenn er bis auf Homöomorphie $K_5$ oder $K_{3,3}$ nicht als Teilgraphen enthält.

1936 Das erste Buch über Graphentheorie erscheint in Leipzig: D. König, Theorie der endlichen und unendlichen Graphen.

Hier wollen wir die Aufzählung abbrechen. Die folgende Zeit ist gekennzeichnet durch das Eindringen der Graphentheorie in immer mehr Anwendungsbereiche, auf

der anderen Seite durch eine intensive innermathematische Entwicklung der Graphentheorie selbst. Dabei bestimmt neben den Anstössen von außen zunehmend auch eine innermathematische Dynamik diese Entwicklung.

Besonders stürmisch wurde die Entwicklung der Graphentheorie in den letzten Jahrzehnten durch die Verfügbarkeit immer leistungsfähigerer Rechner. Wie allgemein für die sog. Kombinatorische Optimierung gilt insbesondere für die Graphentheorie, dass viele Probleme praktisch lösbar wurden, nachdem sie vorher wegen der großen Anzahl durchzuführender Rechenoperationen nicht in vertretbarer Zeit bearbeitet werden konnten. Viele dieser Probleme kommen aus Operations Research oder Informatik.

Eine besondere Station in der Entwicklung der Graphentheorie muss allerdings noch herausgehoben werden: im Jahre 1976 bewiesen Appel und Haken die Richtigkeit der Vierfarbenvermutung, die mehr als hundert Jahre lang als einfachsten und zugleich faszinierendsten ungelösten Probleme der Mathematik gegolten hatte. Brisant an dem Beweis ist, dass zur Untersuchung einer großen Anzahl gleichartiger Fälle die Hilfe eines Computers in Anspruch genommen wurde und es für Menschen praktisch kaum möglich ist, diese Fälle einzeln (ohne Hilfe eines Rechners) nachzuprüfen.

Im vorliegenden Kurs ist die Auswahl des gebotenen Stoffes hauptsächlich unter dem Aspekt der Anwendungen in der Elektrotechnik erfolgt. Dies konnte nur eine grobe Leitlinie sein, denn schon die Einordnung des Stoffes in das ”Gebäude” der Graphentheorie verlangt auch ein Eingehen auf nicht unmittelbar praxisrelevante Bereiche. Im zweiten Teil des Kurses spielen insbesondere "Zufallsgraphen" eine große Rolle. Auf Färbungsprobleme wird in dem Kurs nicht eingegangen, obwohl das Vierfarbenproblem auch für die Herausbildung der Graphentheorie von zentraler Bedeutung war.

Methodisch wird in dem Kurs der Stoff vorwiegend vom algorithmischen Standpunkt her entwickelt. Für ein solches Vorgehen ist ein kurzes Eingehen auf die Theorie der Algorithmen, wie sie in Logik und Theoretischer Informatik betrieben wird, nötig.

Der erste Teil des Kurses hat neben der Vermittlung der grundlegenden Begriffe der Graphentheorie das Herstellen eines Grundverständnisses für die Theorie der Algorithmen zum Inhalt. Ferner werden einige der ”klassischen” Graphenalgorithmen behandelt (z. B. zur Bestimmung kürzester Wege), die in verschiedensten Anwendungen ein Rolle spielen.
Der zweite Teil des Kurses ist einigen speziellen Anwendungen der Graphentheorie in der Elektrotechnik gewidmet.
Die behandelten mathematischen Sätze werden in der Regel vollständig bewiesen. Es gibt zwei Ausnahmen von dieser Regel: Handelt es sich um einen Satz mit einem technisch komplizierten Beweis, so dass der große Aufwand des Beweises zu der Bedeutung des Satzes für diesen Kurs in einem Missverhältnis steht, so wird nur die Beweisidee angedeutet. Geht es gar um einen Satz, der eigentlich nicht zu dem

Stoff dieses Kurses gehört, sondern mehr dem Ausblick auf angrenzende Bereiche der Graphentheorie dient, so ist der Beweis ganz weggelassen.

An Voraussetzungen für das Studium des Kurses sind nötig die Vertrautheit mit der Mengensprache sowie Kenntnis der Grundlagen der Linearen Algebra (einschließlich des Umgangs mit Matrizen). Die wichtigsten Begriffe aus diesen Bereichen sind - sozusagen zur Erinnerung - in Anhängen kurz erläutert.

Das Literaturverzeichnis führt die bei der Erstellung des ersten Teils verwendete Literatur sowie einige weitere Bücher auf, die anzusehen für einen Leser sicher lohnend ist. Es erhebt jedoch nicht den Anspruch, eine vollständige Bibliographie des Gebietes der Graphentheorie zu sein, eine solche wäre wesentlich umfangreicher.

Die vorliegende Fassung des Kurses ist eine Überarbeitung der ersten Version, die ab 1990 an der Fernuniversität angeboten wurde. Für den ersten Teil des Kurses wurden auch die Unterlagen zu Vorlesungen verwendet, die - jeweils an der Technischen Hochschule Darmstadt - vom ersten Autor zusammen mit Prof. Dr. P. Burmeister im Jahre 1975 und vom zweiten Autor im Jahre 1982 gehalten wurden.

# Autorenvorstellung

## Prof. Dr.-Ing. Firoz Kaderali

| | |
|---|---|
| 1963 - 69 | Studium der Theoretischen Elektrotechnik an der Technischen Hochschule Darmstadt |
| 1969 - 74 | Assistent/Dozent für Grundlagen der Elektrotechnik an der Technischen Hochschule Darmstadt |
| 1974 | Promotion auf dem Gebiet der Netzwerktheorie an der Technischen Hochschule Darmstadt |
| 1974 - 76 | Dozent für Statistische Signaltheorie an der Technischen Hochschule Darmstadt |
| 1976 - 81 | Projektleiter (Digitales Ortsnetz) im Forschungszentrum der Firma SEL (ITT)/Stuttgart |
| 1981 - 86 | Hauptabteilungsleiter (Systementwicklung Großsysteme) bei (Bosch) Telefonbau und Normalzeit/Frankfurt |
| Seit 1986 | Professor für Kommunikationssysteme an der FernUniversität Hagen Arbeitsgebiete: Kommunikationssysteme, -netze und -protokolle; Datenschutz und Datensicherheit in Kommunikationsnetzen; Einsatz von neuen Medien in der Lehre |
| 1989 - 94 | Leiter der Projektträgerschaft TELETECH NRW |
| 1990 - 96 | Mitglied der ISDN Forschungskommission des Landes NRW |
| Seit 1992 | Direktor des Forschungsinstituts für Telekommunikation (FTK)/Dortmund. |
| 1995 - 2001 | Mitglied der Steuerungsgruppe der Landesinitiative media NRW |
| 1999 - 2003 | Sprecher des Forschungsverbundes Datensicherheit NRW |
| 2000-2002 | Vorsitzender des Beirates der Gesellschaft für IT-Sicherheit in Bochum |
| Seit 2002 | Vorsitzender der Open Source Initiative CampusSource |

# Prof. Dr. rer. nat. Werner Poguntke

| | |
|---|---|
| 1968 - 1971 | Studium der Mathematik und Physik an der Universität Bonn |
| 1971 - 1972 | Studium der Mathematik und Physik an der TH Darmstadt |
| 1974 | Promotion in Mathematik auf dem Gebiet der Verbandstheorie an der TH Darmstadt |
| 1974 - 1980 | Wissenschaftlicher Angestellter im Fachbereich Mathematik der TH Darmstadt |
| 1978/79 | Forschungsaufenthalt in Kanada |
| 1980 - 1985 | Hochschulassistent im Fachbereich Mathematik der TH Darmstadt |
| 1981 | Habilitation in Mathematik an der TH Darmstadt |
| 1985 - 1988 | Tätigkeit in der Software-Entwicklung für Vermittlungsanlagen bei der Firma Telenorma in Frankfurt |
| 1988 - 1994 | Wissenschaftlicher Angestellter im Lehrgebiet Kommunikationssysteme der Fernuniversität Hagen |
| Seit 1994 | Professor für Angewandte Informatik an der Fachhochschule Südwestfalen |

# Gliederung

**Vorwort zum Kurs** ..... **iii**

**Autorenvorstellung** ..... **vi**

Prof. Dr.-Ing. Firoz Kaderali ..... vi

Prof. Dr. rer. nat. Werner Poguntke ..... vii

**1 Grundbegriffe** ..... **1**

1.1 Pseudographen, Multigraphen, Graphen ..... 1

1.2 Wege, Kreise, Zusammenhang ..... 12

1.3 Kreise und Schnitte ..... 23

**2 Darstellung von Graphen** ..... **34**

2.1 Diagramme und Planarität ..... 34

2.2 Matrizen ..... 38

2.3 Weitere Matrizen und deren Eigenschaften ..... 44

**3 Algorithmen** ..... **48**

3.1 Das Erkennen und Suchen von Bäumen ..... 48

3.2 Algorithmen und deren Komplexität ..... 51

3.3 Weitere Algorithmen und Begriffe ..... 58

**4 Pseudodigraphen** ..... **65**

4.1 Grundbegriffe ..... 65

4.2 Multigraphen und Matrizen ..... 74

**5 Bewertungen** ..... **84**

5.1 Ecken-, Kanten- und Bogenbewertungen ..... 84

5.2 Die algebraische Struktur von Bewertungen ..... 86

**6 Kürsteste Wege und minimale Gerüste** ..... **96**

6.1 Kürzeste Wege ..... 96

6.2 Minimale Gerüste ..... 112

**7 Flüsse** ..... **122**

7.1 Einführung ..... 122

7.2 Die Sätze von Ford und Fulkerson ..... 127

7.3 Der Satz von Edmonds und Karp ..... 137

7.4 Eine kombinatorische Anwendung: Der Satz von Menger ..... 140

7.5 Weitere kombinatorische Anwendungen ..... 143

7.6 Zulässige Flüsse und Zirkulationen ..... 149

7.7 Synthese minimaler Netze ..... 156

**8 Wegauswahl in Netzen** ..... **164**

8.1 Das Problem der Wegauswahl im Kommunikationsnetzen ..... 164

8.2 Algorithmen zur Bestimmung kürzester Wege ..... 170

8.3 Das Stabilitätsproblem bei der Nutzung kürzester Wege ..... 185

8.4 Zur Übertragung von Routing-Informationen ..... 193

8.5 Das Routing im Internet ... 199
8.6 Das Routing im Zeichengabesystem Nr. 7 ... 211
8.7 Optimales Routing ... 225

**9 Zuverlässigkeit von Netzen ... 229**
9.1 Zufallsgraphen ... 229
9.2 Der Zusammenhang von Zufallsgraphen ... 234
9.3 Zuverlässigkeitsmaße und -polynome ... 242
9.4 Zur Komplexität des Zuverlässigkeitsproblems ... 251
9.5 Abschätzungen für das Zuverlässigkeitspolynom ... 260
9.6 Routing und Zuverlässigkeit ... 271
9.7 Synthese extremaler Netze ... 287

**10 Kleine Welten ... 292**
10.1 Soziale und andere Netze ... 292
10.2 Durchmesser, Cluster-Koeffizient, und die Rolle von Zufallsgraphen ... 294
10.3 Das Modell von Watts und Strogatz ... 298
10.4 Modelle mit effizientem Routing ... 302
10.5 Das Web als Graph ... 313
10.6 Der Graphen-Erzeugungsprozess nach Barabasi, Albert und Jeong 318
10.7 Skalenfreie Graphen ... 321

**11 Random Walks und Elektrische Netzwerke ... 332**
11.1 Zwei einführende Beispiele ... 332
11.2 Random Walks in einer und zwei Dimensionen ... 335
11.3 Random Walks in endlichen Graphen ... 344
11.4 Random Walks in unendlichen Graphen ... 353

A **Kurze Übersicht der verwendeten Begriffe und Symbole aus der Mengenlehre ... 359**

B **Kurze Erläuterung der verwendeten Begriffe aus der Linearen Algebra ... 361**

C **Kurze Erläuterung der verwendeten Begriffe aus der Theorie der Matrizen ... 365**

D **Pascal-Programme zu den Algorithmen von Dijkstra und Kruskal ... 368**

E **Pascal-Programm zum Algorithmus von Ford und Fulkerson ... 377**

F **Boolesche Ausdrücke und ihre Umformung ... 384**

G **Gerüste eines Graphen ... 391**

H **Pascal-Programm zur Berechnung des Zuverlässigkeitspolynoms ... 395**

**Lösungshinweise zu Selbsttestaufgaben ... 399**

**Übungsaufgaben ... 415**

**Lösungshinweise zu Übungsaufgaben ... 429**

**Literaturverzeichnis** ..... **453**

# 1 Grundbegriffe

## 1.1 Pseudographen, Multigraphen, Graphen

**Definition 1.1-1: Pseudograph**

*Ein **Pseudograph** ist ein Tripel $P = (E, K, v)$ bestehend aus einer Eckenmenge E, einer Kantenmenge K und einer (Inzidenz-) Abbildung*

$$v : K \to \{\{x, y\} | x, y \in E\}.$$

**Pseudograph**

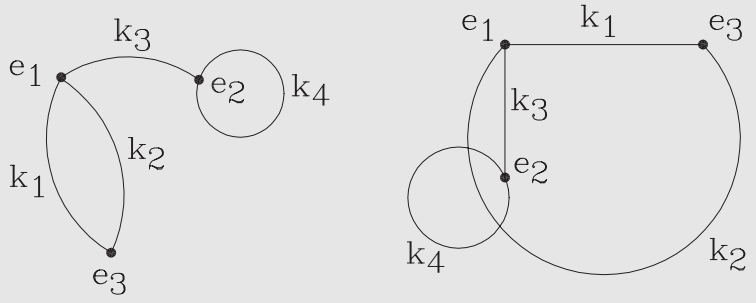
*Die Elemente von $E$ heißen **Ecken**, die von $K$ **Kanten**.*
*Ist $k \in K$ mit $v(k) = \{x, y\}$, so heißen $x$ und $y$ die **Endecken** von $k$. (Man sagt auch: $x$ und $y$ **inzidieren** mit $k$ bzw. $k$ inzidiert mit $x$ und $y$.)*

**Ecken**
**Kanten**
**Endecken einer Kante**

Es werden im folgenden nur **endliche Pseudographen** (d. h. $E$ und $K$ sind endliche Mengen) betrachtet. Einen endlichen Pseudographen kann man sich stets durch ein **Diagramm** veranschaulichen: die Ecken werden durch Punkte der Zeichenebene dargestellt, die Kanten als Linien, die die Endecken der Kante verbinden.

**endliche Pseudographen**
**Diagramme**

**Beispiel 1.1-1:**
Der Pseudograph $P$ mit $E = \{e_1, e_2, e_3\}$, $K = \{k_1, k_2, k_3, k_4\}$ und $v(k_1) = v(k_2) = \{e_1, e_3\}$, $v(k_3) = \{e_1, e_2\}$, $v(k_4) = \{e_2\}$ wird durch jedes der beiden folgenden Diagramme dargestellt:




**Definition 1.1-2: Parallelkante**

*Zwei oder mehrere Kanten, die mit denselben Ecken inzidieren, heißen **Parallelkanten**.*

**Parallelkanten**

**Definition 1.1-3: Schleife**

*Eine Kante $k$ mit zwei gleichen Endecken (d. h. es gilt $v(k) = \{x\}$) ist eine (Selbst-) **Schleife**.*

**Schleifen**

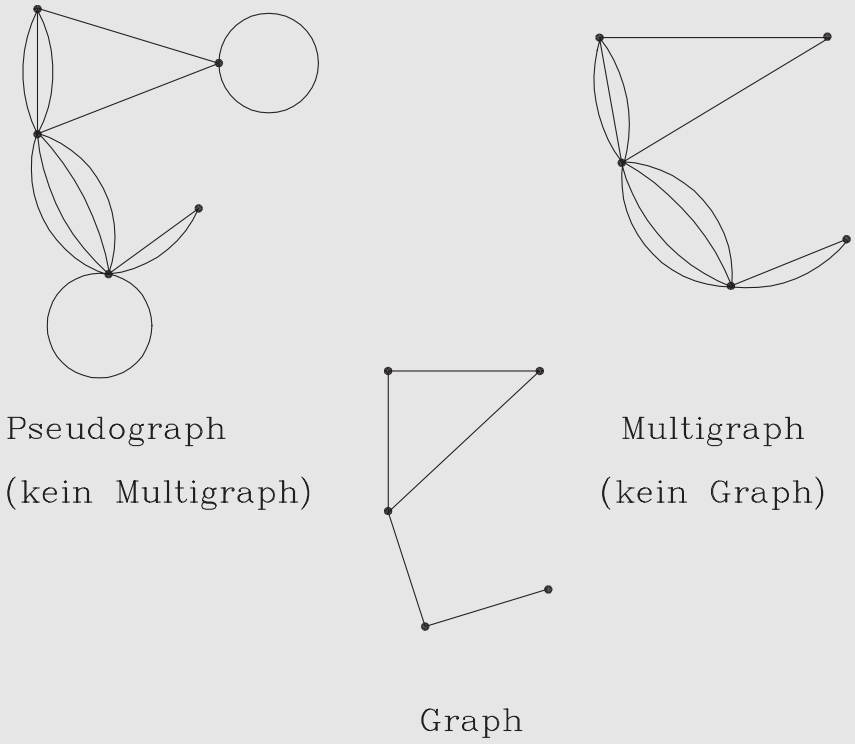
**Definition 1.1-4: Multigraph**

**Multigraph** *Ein Pseudograph ohne Schleifen heißt **Multigraph**.*

**Definition 1.1-5: Graph**

**Graph** *Ein Multigraph ohne Parallelkanten heißt **Graph**.*

**Beispiel 1.1-2:**
Beispiele durch Diagramme:




Da in einem Graphen jede Kante durch ihre zwei verschiedenen Endecken eindeutig bestimmt ist, kann ein Graph auch als ein Paar $G = (E, K)$ mit

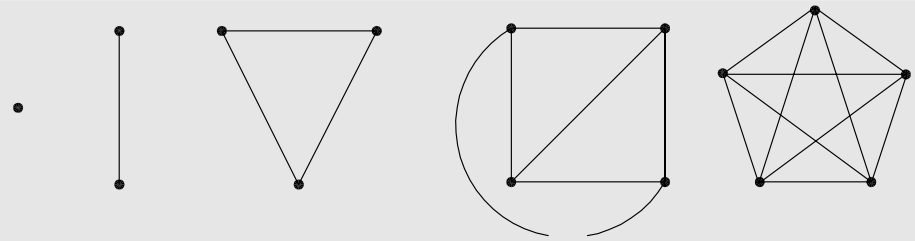
$$K \subseteq \{\{x, y\} | x, y \in E, x \neq y\}$$

aufgefasst werden. Diese Definition wird in der Literatur häufig verwendet. Einen Graphen im Sinne von Definition 1.1-1 bekommt man dann, indem man $v(k) := k$ für jedes $k \in K$ setzt.

**Definition 1.1-6: Vollständiger Graph**

**vollständiger Graph** **Simplex** *Ein Graph heißt **vollständig** oder auch ein **Simplex**, wenn je zwei verschiedene Ecken des Graphen durch eine Kante verbunden sind.*

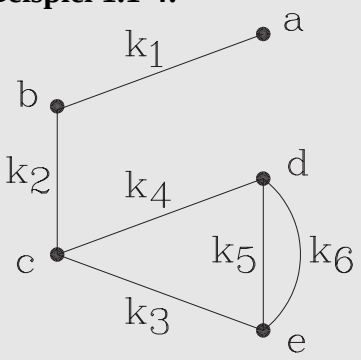
**Beispiel 1.1-3:**
In den folgenden Diagrammen sind vollständige Graphen dargestellt.



**Definition 1.1-7: Benachbarte Ecken**
*Zwei **Ecken** $a, b \in E$ in einem Pseudographen $P$ heißen **benachbart**, wenn ein $k \in K$ mit $v(k) = \{a, b\}$ existiert.* **benachbarte Ecken**

**Definition 1.1-8: Benachbarte Kanten**
*Zwei **Kanten** $k_1, k_2 \in K$ heißen **benachbart** in $P$, wenn $v(k_1) \cap v(k_2) \neq \emptyset$ gilt, d. h. wenn sie in mindestens einer Ecke gemeinsam inzident sind.* **benachbarte Kanten**

**Beispiel 1.1-4:**




$a$ und $b$ sind benachbart,
$a$ und $d$ nicht,
$k_4$ und $k_5$ sind benachbart,
$k_5$ und $k_6$ sind benachbart,
$k_1$ und $k_4$ nicht.

**Definition 1.1-9:** *$P$ sei ein Pseudograph, $a \in E$. Dann heißt die Anzahl der an $a$ inzidenten Kanten (wobei eine Kante, die zweifach an $a$ inzident ist, auch doppelt gezählt wird) der Eckengrad (kurz: **Grad**) von $a$ in $P$ und wird mit $\gamma(a, P)$ bzw. nur $\gamma(a)$ bezeichnet.* **Grad**

*Für einige Sonderfälle werden zusätzliche Begriffe benutzt:*

**Definition 1.1-10: Isolierte Ecke**

isolierte Ecke

*Ist $\gamma(a, P) = 0$, so heißt a eine **isolierte Ecke** von $P$.*

**Definition 1.1-11: n-regulär**

n-regulärer Pseudograph

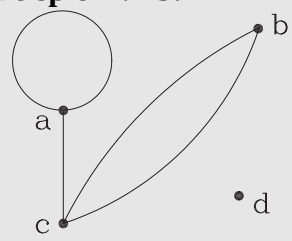
*Haben alle Ecken von $P$ den gleichen Grad n, so heißt $P$ **n-regulär**.*

**Definition 1.1-12: Nullgraph**

Nullgraph

*Der Pseudograph ohne Ecken und Kanten $(E = K = \emptyset)$ heißt auch **leerer Graph** oder **Nullgraph**.*

**Beispiel 1.1-5:**




Es ist $\gamma(a) = \gamma(c) = 3, \gamma(b) = 2, \gamma(d) = 0$; $d$ ist eine isolierte Ecke.

Als nächstes werden zwei Gesetzmäßigkeiten nachgewiesen, denen die Zahlen $\gamma(a, P)$in einem beliebigen Pseudographen unterliegen.

**Satz 1.1-1:** *Für jeden Pseudographen $P$ gilt*

$$\sum_{a \in E} \gamma(a, P) = 2|K|.$$

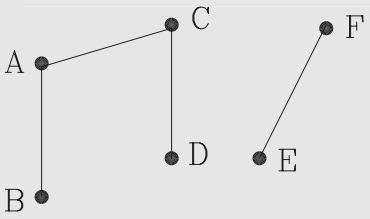
*(Man prüfe die Formel an Beispiel 1.1-5 nach.)*

**Beweis:** Da jede Kante bei der Bildung der Summe der Eckengrade genau zweimal gezählt wird (je einmal bei jeder der beiden Endecken), ergibt sich die Behauptung sofort.

**Satz 1.1-2:** *In jedem Pseudographen $P$ gibt es eine gerade Anzahl von Ecken mit ungeradem Eckengrad.*

**Beweis:** Aufgrund von Satz 1.1-1 ist die Summe aller Eckengrade eine gerade Zahl. Durch Subtrahieren aller geraden Eckengrade von dieser Zahl bleibt eine gerade Zahl übrig, die nun die Summe aller ungeraden Eckengrade ist. Dies kann aber nur dann der Fall sein, wenn die Summe aller ungeraden Eckengrade eine gerade Anzahl von Summanden hat.

**Beispiel 1.1-6:**
Wir nehmen an, unter sechs Freunden A, B, C, D, E und F hätten an einem Tag eine Reihe von Telefongesprächen stattgefunden. Die Personen können als Ecken und die stattgefundenen Telefongespräche als Kanten eines Graphen aufgefasst werden. So könnte sich z. B. folgender Graph ergeben:

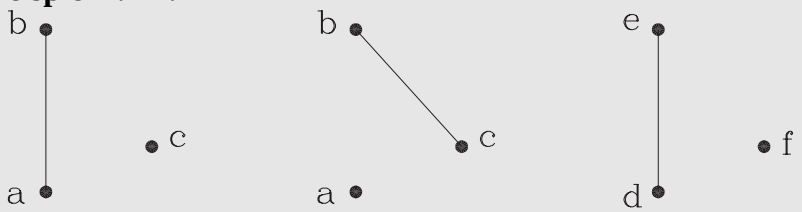



($A$ und $B$ haben miteinander telefoniert, $D$ und $E$ nicht, etc.)

Satz 1.1-1 und Satz 1.1-2 ergeben nun allgemeingültige Aussagen über die Anzahlen der geführten Telefongespräche. So wäre es z. B. nicht möglich, dass A drei, B zwei, C ein, D vier, E zwei und F drei Telefongespräche geführt hat, denn $3 + 2 + 1 + 4 + 2 + 3$ ist $15$ und somit ungerade.

Wir kommen nun zu dem zentralen Begriff der **Isomorphie** von Pseudographen. Wie auch bei anderen mathematischen Objekten üblich, ist hier die Grundidee, verschiedene Objekte in gewissen Zusammenhängen als ein Objekt ansehen zu dürfen, wenn sie sich streng mengentheoretisch unterscheiden, von der Struktur her aber identisch sind. **Isomorphie**

**Beispiel 1.1-7:**




Die drei in den Diagrammen dargestellten Graphen sind zwar verschieden, jedoch von ihrer Struktur her gleich.

**Definition 1.1-13: Isomorphismus**
*$P = (E, K, v)$ und $P' = (E', K', v')$ seien Pseudographen. Ein Paar von Abbildungen*

$$\phi : E \to E', \psi : K \to K'$$

*heißt ein **Isomorphismus** von $P$ auf $P'$, wenn $\phi$ und $\psi$ bijektiv sind und wenn für alle $k \in K$ und $x, y \in E$ gilt:* **Isomorphismus**

$$v(k) = \{x, y\} \Leftrightarrow v'(\psi(k)) = \{\phi(x), \phi(y)\}.$$

**isomorph** Existiert ein Isomorphismus von $P$ auf $P'$, so heißen $P$ und $P'$ **isomorph** ( in Zeichen: $P \simeq P'$ ).

Für Graphen lässt sich der Begriff Isomorphismus einfacher fassen, wenn man die Kanten als Eckmenge auffasst:

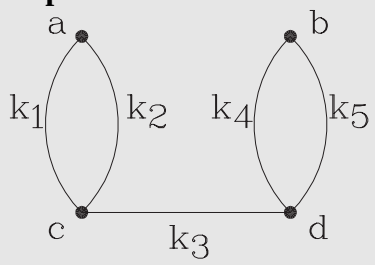
$G$ und $G'$ seien Graphen. Eine Abbildung

$$\phi : E \to E'$$

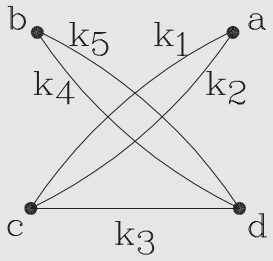
heißt Isomorphismus von G auf $G'$, wenn $\phi$ bijektiv ist und für alle $\{x, y\} \in E$ gilt

$$\{x, y\} \in K \Leftrightarrow \{\phi(x), \phi(y)\} \in K'.$$

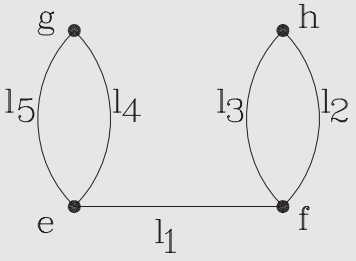
**Beispiel 1.1-8:**




$P_1$




$P_1$




$P_2$

Man beachte, dass die beiden ersten Diagramme Darstellungen desselben Multigraphen $P_1$ sind. Ein Isomorphismus von $P_1$ auf $P_2$ ist z . B. gegeben durch

$$\phi(a) = g, \phi(b) = h, \phi(c) = e, \phi(d) = f, \psi(k_1) = l_5,$$
$$\psi(k_2) = l_4, \psi(k_3) = l_1, \psi(k_4) = l_3, \psi(k_5) = l_2.$$

Bei Beispiel 1.1-7 reicht es, da es sich um Graphen handelt, eine Abbildung zwischen den Eckenmengen anzugeben, um einen Isomorphismus zu beschreiben. So ist z. B. die Abbildung $\phi$ mit $\phi(b) = b, \phi(a) = c, \phi(c) = a$ ein Isomorphismus des im ersten Diagramm dargestellten Graphen auf den zweiten.
Es ist wichtig anzumerken, dass beim Umgang mit Pseudographen die Begriffe "Gleichheit" und "Isomorphie" oft nicht sauber getrennt werden, was jedoch in der Regel nicht zu falschen Schlussfolgerungen führt: hinsichtlich aller mathematischen Eigenschaften (und nur abgesehen von der "Natur" der Elemente) sind zwei isomorphe Pseudographen im Grunde "gleich".
Stillschweigend haben wir den Begriff der Isomorphie auch bereits bei den Beispielen Beispiel 1.1-2 und Beispiel 1.1-3 benutzt, wo Pseudographen durch ihre Diagramme ohne Bezeichnungen für Ecken und Kanten definiert wurden. Wenn man einen Pseudographen durch ein Diagramm einführt, das nicht für alle Ecken und Kanten Namen enthält, so ist damit strenggenommen gemeint, dass von irgendeinem der vielen zueinander isomorphen Pseudographen die Rede ist, die durch das Diagramm beschrieben werden; man kann auch die Auffassung haben, dass durch das Diagramm eine **Isomorphieklasse** von Pseudographen festgelegt wird. Es ist aber trotzdem üblich, in dieser Situation von dem in dem betreffenden Diagramm dargestellten Pseudographen zu sprechen.

**Isomorphieklasse**

Als nächstes werden die grundlegenden Operationen beschrieben, mit denen aus vorgegebenen Pseudographen weitere konstruiert werden können.

**Definition 1.1-14: Teilgraph**

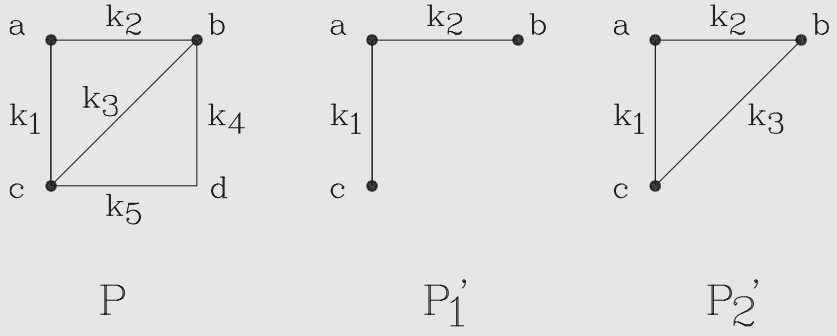
*Ein Pseudograph $P' = (E', K', v')$ heißt ein **Teilgraph** von $P = (E, K, v)$ (in Zeichen $P' \subseteq P$), wenn $E' \subseteq E, K' \subseteq K$ und $v'(k) = v(k)$ für alle $k \in K'$. Ist $P' \neq P$, so heißt $P'$ ein **echter Teilgraph** von $P$ ($P' \subset P$). Gehört jede Kante von $P$, die zwei Ecken von $P'$ verbindet, zu $P'$, so heißt $P'$ **der von** $E'$ **induzierte** (oder: **aufgespannte**) **Untergraph** von $P$. (Der Kürze wegen wählt man hier nicht die Bezeichnungen "Teilpseudograph" oder "Unterpseudograph".)*

**Teilgraph**

**echter Teilgraph**

**induzierter Untergraph**

**Beispiel 1.1-9:**




$P_1'$ ist Teilgraph, jedoch kein Untergraph, weil die Kante $k_3$ nicht zu $P_1'$ gehört, aber die zu $P_1'$ gehörenden Ecken $b$ und $c$ verbindet. $P_2'$ ist der von $\{a, b, c\}$ induzierte Untergraph.

Es ist leicht einzusehen, dass ein von $E'$ induzierter Untergraph von $P$ durch die Angabe von $E'$ eindeutig festliegt. Demgegenüber ist ein Teilgraph bis auf isolierte Ecken durch seine Kanten festgelegt:

**Definition 1.1-15: Induzierter Teilgraph**
*Ist $P' \subseteq P$, und enthält $P'$ keine isolierten Ecken (d. h. für alle $a \in E'$ gilt $\gamma(a, P') \neq 0$), so kann man $P'$ als den **von** $K'$ **induzierten Teilgraphen** von $P$ auffassen, welcher durch $K'$ eindeutig bestimmt ist.*

$P_1'$ aus Beispiel 1.1-9 ist durch die Menge $K' = \{k_1, k_2\}$ induziert.

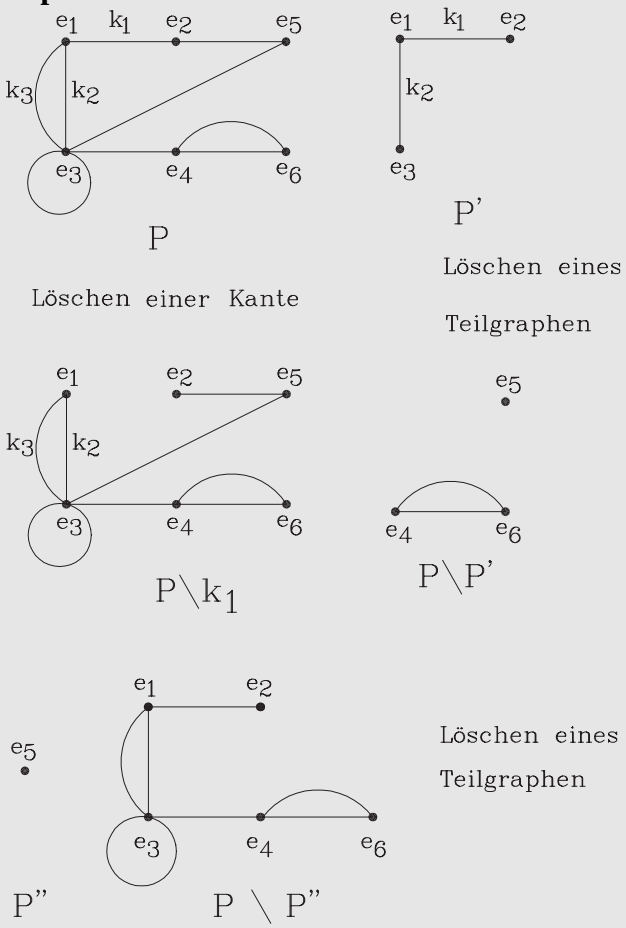
**Definition 1.1-16: Löschen einer Kante**

**Löschen einer Kante von $P$**

**Löschen eines Teilgraphen von $P$**

*Sei $P = (E, K, v)$ ein Pseudograph. Unter dem **Löschen einer Kante von** $P$ versteht man (für $k \in K$) die Bildung des Pseudographen $P\backslash k := (E, \tilde{K}, \tilde{v})$ mit $\tilde{K} = K\backslash\{k\}$ und $\tilde{v} = v/\tilde{K}$. Entsprechend ist das **Löschen eines Teilgraphen von** $P$(für $P' \subseteq P$) die Bildung des Pseudographen $P\backslash P' = (\tilde{E}, \tilde{K}, \tilde{v})$ mit $\tilde{E} = E\backslash E', \tilde{K} = \{k \in K | v(k) \cap E' = \emptyset\}$ und $\tilde{v} = v/\tilde{K}$.*

**Beispiel 1.1-10:**




Besteht, wie im letzten Beispiel, der gelöschte Teilgraph nur aus einer Ecke e, so schreibt man für das Ergebnis auch kurz $P\backslash e$ (hier: $P\backslash e_5$ statt $P\backslash P''$).

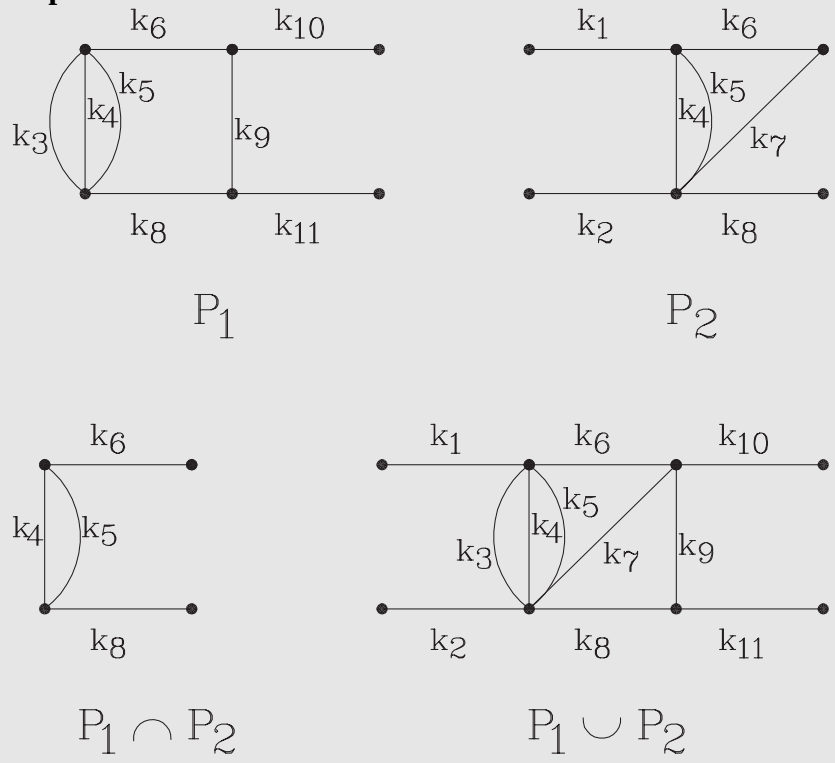
**Definition 1.1-17: Durchschnitt der Pseudographen**
*$P_i = (E_i, K_i, v_i), i \in I$, sei eine Familie von Pseudographen, und für $k \in K_i \cap K_j$ $(i, j \in I)$ sei stets $v_i(k) = v_j(k)$. Unter dem* ***Durchschnitt der Pseudographen*** *$P_i$ $(i \in I)$ versteht man den Pseudographen*

$$\bigcap_{i\in I} P_i := (\bigcap_{i\in I} E_i, \bigcap_{i\in I} K_i, \bigcap_{i\in I} v_i),$$

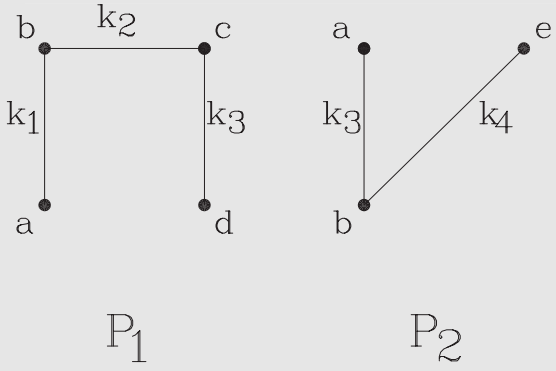
*unter ihrer* ***Vereinigung*** *den Pseudographen*

$$\bigcup_{i\in I} P_i := (\bigcup_{i\in I} E_i, \bigcup_{i\in I} K_i, \bigcup_{i\in I} v_i).$$

**Beispiel 1.1-11:**




**Beispiel 1.1-12:**
Wir illustrieren die Notwendigkeit der Bedingung $v_i(k) = v_j(k)$ für $k \in K$ bei der Definition von Durchschnitt und Vereinigung an folgenden Graphen:




Da die Kante $k_3$ in $P_1$ die Ecken c und d, in $P_2$ aber die Ecken a und b verbindet, lassen sich $P_1 \cap P_2$ und $P_1 \cup P_2$ nicht sinnvoll definieren.

**Definition 1.1-18: Summe**

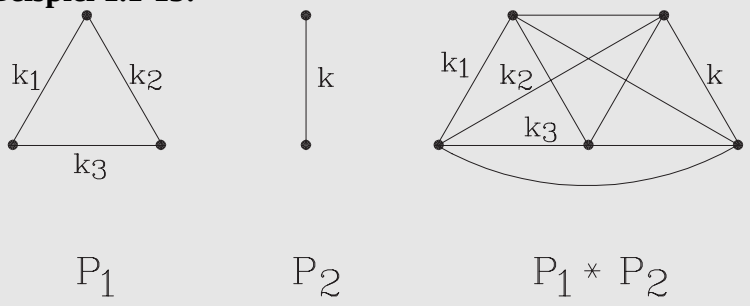
**Summe von zwei Graphen**

*$P_1$ und $P_2$ seien zwei disjunkte Pseudographen (d. h. $E_1 \cap E_2 = \emptyset$ und $K_1 \cap K_2 = \emptyset$). Dann verstehen wir unter der* ***Summe***

$$P_1 * P_2$$

*den (bis auf Isomorphie eindeutig bestimmten) Pseudographen, der aus $P_1 \cup P_2$ dadurch entsteht, dass jede Ecke von $P_1$ mit jeder Ecke von $P_2$ zusätzlich durch eine (neue) Kante verbunden wird.*

**Beispiel 1.1-13:**




In Verallgemeinerung von Definition 1.1-18 wird definiert:

**Definition 1.1-19: Summe der $P_i$ über dem Graphen**
*Ist $G$ ein Graph mit der Eckenmenge $E = \{e_i | i \in I\}$, und ist $P_i = (E_i, K_i, v_i)$ $(i \in I)$ eine disjunkte Familie von Pseudographen, so verstehen wir unter der **Summe der $P_i$ über dem Graphen** $G$, $G_{i\in I}P_i$, den (bis auf Isomorphie eindeutig bestimmten) Pseudographen, der aus $\bigcup_{i\in I} P_i$ dadurch hervorgeht, dass alle $a \in E_i$ und $b \in E_j$ für $i \neq j$ genau dann zusätzlich durch eine neue Kante verbunden werden, wenn die Ecken $e_i$ und $e_j$ in $G$ benachbart sind.*

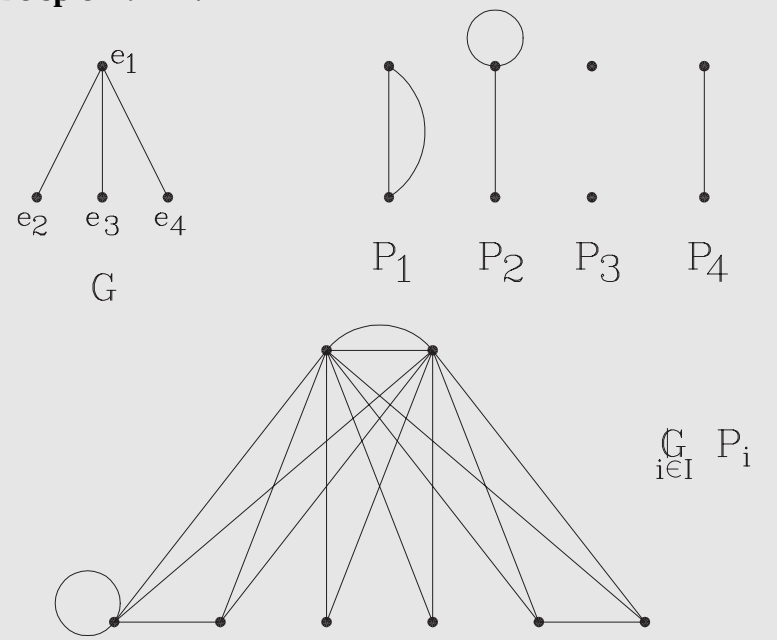
Setzt man in Definition 1.1-19 speziell $I = \{1, 2\}$ und wählt für $G$ den 2-eckigen Graphen mit einer Kante, so ergibt sich Definition 1.1-18.

In $G_{i\in I}P_i$ heißt $G$ der **äußere Graph**, $P_i$ sind die **inneren Pseudographen**. Ist $G$ ein Simplex, und sind alle $P_i$ kantenlos, so heißt $G_{i\in I}P_i$ **simplexartig**.

**äußerer Graph, innerer Pseudograph**

**simplexartig**

**Beispiel 1.1-14:**




**Selbsttestaufgabe 1.1-1:**

*Erläutern Sie den Unterschied zwischen den Begriffen Teilgraph und Untergraph und illustrieren Sie dies anhand eines Beispiels.*

## 1.2 Wege, Kreise, Zusammenhang

**Definition 1.2-1: Kantenfolge**

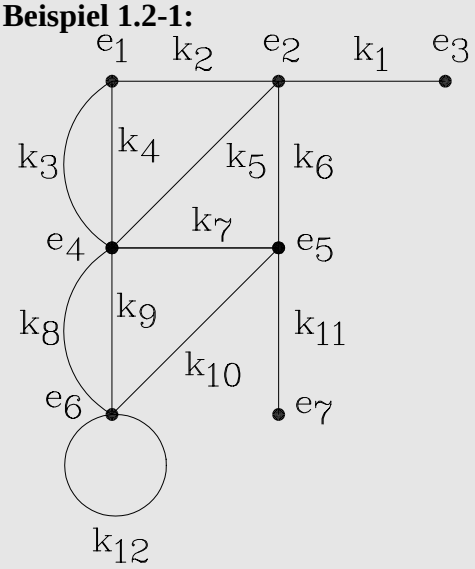
$P = (E, K, v) =$ *sei ein Pseudograph.*

*Eine Folge* $(e_0, k_1, e_1, k_2, e_2, \ldots, e_{i-1}, k_i, e_i, \ldots, k_n, e_n)$ *mit* $e_i \in E$ *und* $k_j \in K$ *heißt eine Kantenfolge , wenn für alle* $i$ $(1 \leq i \leq n)$ *gilt* $v(k_i) = \{e_{i-1}, e_i\}$*. Die Ecke* $e_0$ *heißt die* ***Anfangsecke****, die Ecke* $e_n$ *die* ***Endecke*** *der Folge.*

**Kantenfolge**

**Anfangsecke, Endecke einer Kantenfolge**

**Beispiel 1.2-1:**




Kantenfolgen sind z. B.:
$F_1 = (e_3, k_1, e_2, k_6, e_5, k_7, e_4)$,
$F_2 = (e_4, k_8, e_6, k_{12}, e_6, k_{10}, e_5, k_7, e_4)$,
$F_3 = (e_1, k_3, e_4, k_9, e_6, k_9, e_4, k_5, e_2)$.

Eine Kantenfolge wird häufig nur durch die Folge ihrer Kanten (ohne explizite Nennung der Ecken) dargestellt. Für das obige Beispiel ergibt sich $F_1^{'} = (k_1, k_6, k_7)$, $F_2^{'} = (k_8, k_{12}, k_{10}, k_7)$, $F_3^{'} = (k_3, k_9, k_9, k_5)$; in allen diesen drei Fällen ist die ursprüngliche Kantenfolge aus der abgekürzten Folge rekonstruierbar. Die Kantenfolge $F_4 = (e_4, k_8, e_6, k_8, e_4)$ könnte aber z. B. nicht aus $F_4^{'} = (k_8, k_8)$ zurückgewonnen werden.

**Definition 1.2-2: Kantenzug**

*Eine Kantenfolge ist **geschlossen**, falls ihre Anfangsecke gleich ihrer Endecke ist, andernfalls ist sie **offen**. Die Anzahl der Kanten in einer Kantenfolge $F$ wird als ihre **Länge** $|F|$ bezeichnet. Ein **Kantenzug** ist eine Kantenfolge, die keine Kante zweimal enthält. Ein **Weg** ist ein Kantenzug, der auch keine Ecke zweimal enthält. Ein **Kreis** ist ein geschlossener Kantenzug, der (von Anfangs- und Endecke abgesehen) keine Ecke zweimal enthält.*

**geschlossene, offene Kantenfolge**

**Länge einer Kantenfolge**

**Kantenzug**

**Weg**

**Kreis**

**Beispiel 1.2-2:**
$F_1$ (aus Beispiel 1.2-1) ist ein Weg mit $|F_1| = 3$.
$F_2$ (mit $|F_2| = 4$) ist ein geschlossener Kantenzug, jedoch kein Kreis, da außer $e_4$ auch $e_6$ zweimal enthalten ist.
$F_3$ (mit $|F_3| = 4$) ist eine offene Kantenfolge, aber kein Kantenzug.

Ist in dem Pseudographen $P$ ein Weg oder ein Kreis $F$ gegeben, so bestimmt dieser eindeutig einen Teilgraphen von $P$, der genau die in $F$ vorkommenden Ecken und

Kanten enthält. Beim Übergang zu diesem Teilgraphen geht allerdings die "Richtung" verloren, im Falle eines Kreises auch die Festlegung der Anfangs- bzw. Endecke.

**Beispiel 1.2-3:**
Es sei wieder der Pseudograph aus Beispiel 1.2-1 zugrundegelegt.
$F_5 = (e_1, k_3, e_4, k_5, e_2, k_2, e_1)$ und
$F_6 = (e_2, k_5, e_4, k_3, e_1, k_2, e_2)$ bestimmen denselben Teilgraphen.

Im folgenden werden, trotz der angesprochenen Probleme, von Wegen bzw. Kreisen herkommende Teilgraphen auch als "Wege" bzw. "Kreise" bezeichnet. Mit dieser Vereinbarung kann dann auch von der Vereinigung oder dem Durchschnitt von Wegen bzw. Kreisen gesprochen werden.

In einem Graphen kann eine Kantenfolge auch durch die Folge der beteiligten Ecken angegeben werden, denn dadurch ist die Kantenfolge bereits eindeutig bestimmt.

**Zusammenhang**

Wir kommen nun zu dem grundlegenden Begriff des **Zusammenhang** eines Pseudographen. Kurz gesagt ist ein Pseudograph zusammenhängend, wenn er "an einem Stück" ist; der in Beispiel 1.1-6 vorkommende Graph ist z. B. nicht zusammenhängend.
Die präzise Begriffsdefinition lautet:

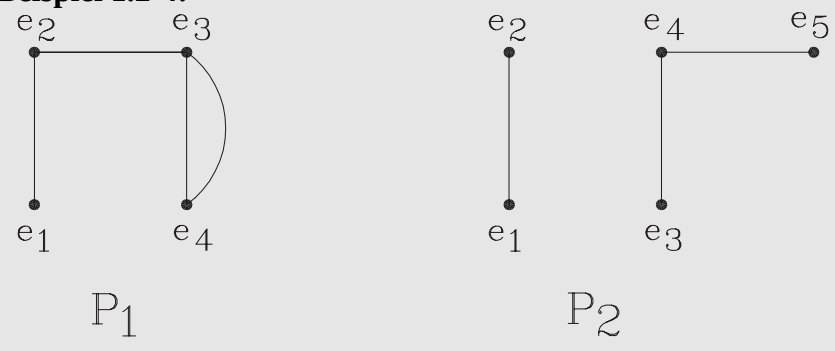
**Definition 1.2-3: Zusammenhang**

**verbindbare Ecken**

*Zwei Ecken $a, b$ eines Pseudographen $P$ heißen **verbindbar** in $P$, wenn $a = b$ ist oder es einen Weg in $P$ von $a$ nach $b$ gibt. Sind je zwei Ecken von $P$ verbindbar, so heißt $P$ **zusammenhängend**.*

**zusammenhängend**

**Beispiel 1.2-4:**




$P_1$ ist zusammenhängend, $P_2$ nicht.

Ein Pseudograph $P = (E, K, v)$ sei gegeben, die Verbindbarkeit von $a, b \in E$ werde durch $a \sim b$ gekennzeichnet.

**Satz 1.2-1:** *Die Verbindbarkeit $\sim$ ist eine Äquivalenzrelation auf $E$.*

**Beweis:** Die Reflexivität ist aufgrund der Definition sofort klar.
Symmetrie: Ist $a \sim b$, so gibt es in $P$ einen Weg von $a$ nach $b$. Durchlaufen dieser Kantenfolge in umgekehrter Richtung ergibt einen Weg von $b$ nach $a$, folglich ist $b \sim a$.
Transitivität: Ist $a \sim b$ und $b \sim c$, so hat man jeweils einen Weg von $a$ nach $b$ und von $b$ nach $c$. Die Aneinanderreihung dieser beiden Wege ergibt eine Kantenfolge von $a$ nach $c$. Als Teilfolge lässt sich dann ein Weg von $a$ nach $c$ finden, somit ist auch $a \sim c$.

**Definition 1.2-4: Zusammenhangskomponenten**
*Die $\sim$-Äquivalenzklassen bilden eine disjunkte Zerlegung der Eckenmenge $E$. Die von den Äquivalenzklassen aufgespannten Untergraphen von $P$ heißen die **(Zusammenhangs-) Komponenten** von $P$.*

**Zusammenhangskomponenten**

Jede Zusammenhangskomponente von $P$ ist ein maximaler zusammenhängender Untergraph von $P$. $P$ ist die disjunkte Vereinigung aller Komponenten.
$P$ selbst ist zusammenhängend, wenn nur eine Komponente existiert.

**Beispiel 1.2-5:**
$P_2$ aus Beispiel 1.2-4 besteht aus zwei Komponenten, die von den $\sim$-Äquivalenzklassen $\{e_1, e_2\}$ und $\{e_3, e_4, e_5\}$ aufgespannt werden.

**Beispiel 1.2-6:**
Nimmt man alle Hauptanschlüsse im öffentlichen Telefonnetz der Deutschen Bundespost sowie die Vermittlungsstellen als Ecken und die dazwischen liegenden Leitungen als Kanten, so ergibt sich ein Multigraph. Die Aussage, dass dieser Multigraph zusammenhängend ist, bedeutet, dass von jedem Punkt zu jedem anderen telefoniert werden kann.

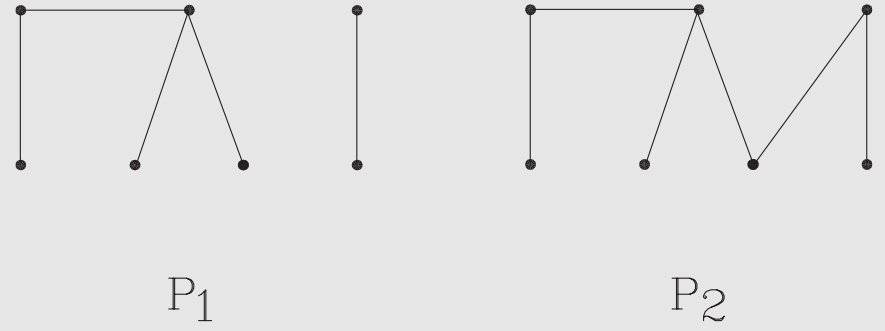
In vielen Anwendungen hat man es mit Pseudographen zu tun, von denen man von vornherein weiß, dass sie bestimmte spezielle Eigenschaften haben. Betrachtet man z. B. bei dem das Telefonnetz darstellenden Multigraphen nur eine Ortsvermittlungsstelle mit den daran angeschlossenen Teilnehmern, so enthält dieser Teilgraph keinen Kreis.
Besonders häufig tauchen Klassen von Pseudographen auf, die durch das Nicht-Enthalten bestimmter Teilgraphen charakterisiert sind. Im folgenden werden einige wichtige dieser Klassen eingeführt.

**Definition 1.2-5: Kreisloser Pseudograph**
*Ein Pseudograph heißt **kreislos**, wenn er keinen Kreis enthält. Ein kreisloser Pseudograph heißt auch **Wald**.*
*Ein zusammenhängender Wald ist ein **Baum**.*
*Ein Wald, der aus $n$ Komponenten besteht, heißt auch ein $n$-**Baum**.*

**kreisloser Pseudograph**
**Wald**
**Baum**
**$n$-Baum**

Man beachte: ein kreisloser Pseudograph ist stets ein Graph.

**Beispiel 1.2-7:**



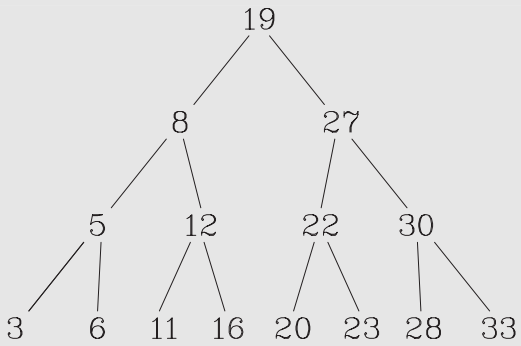

$P_1$ ist ein Wald bzw. ein 2-Baum, $P_2$ ist ein Baum. Der Pseudograph aus Beispiel 1.2-1 enthält Kreise und ist somit kein Wald.

Auf einen Baum bzw. Wald trifft man oft dann, wenn man es mit einer Grundgesamtheit zu tun hat, die "baumartig" hierarchisch geordnet ist. Man denke etwa an die sogenannten Familienstammbäume.

**binäres Suchen**

Auch beim sogenannten **binären Suchen** ist eine Baumstruktur (binärer Suchbaum) im Spiel. Soll z. B. eine weitere Zahl in eine bereits geordnete Liste von Zahlen (etwa Messwerten) an der richtigen Stelle eingeordnet werden, so empfiehlt es sich als Strategie, die neue Zahl zunächst mit der Zahl zu vergleichen, die genau in der Mitte der bisherigen Liste steht, um dann zu wissen, ob man "links" oder "rechts" davon weitersuchen muss; mit der übriggebliebenen Hälfte der Liste wird dann entsprechend verfahren usw. Diese Vorgehensweise entspricht dem Durchlaufen eines Weges in einem Baum, dessen Ecken die Mittelpunkte der Intervalle repräsentieren.

**Beispiel 1.2-8:**
Zu der geordneten Liste
$(3, 5, 6, 8, 11, 12, 16, 19, 20, 22, 23, 27, 28, 30, 33)$ gehört der im folgenden Diagramm dargestellte Suchbaum:



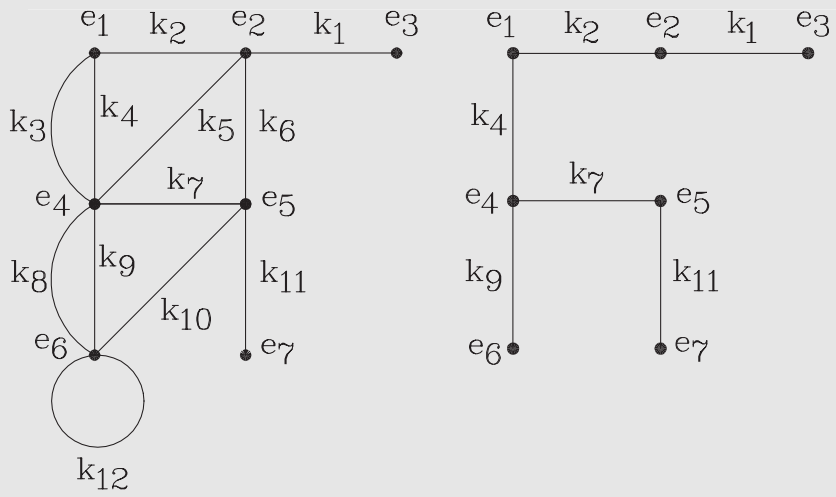
Die Zahl $19$ ist die mittlere aller gegebenen Zahlen, $8$ ist die Mitte des "links" von $19$, $27$ die Mitte des "rechts" von $19$ liegenden Intervalls etc. Eine neu einzuordnende Zahl muss nun nur mit $4$ Zahlen verglichen werden, um an die richtige Stelle eingeordnet werden zu können; z. B. wird $24$ mit $19, 27, 22$ und $23$ verglichen.

Von Interesse ist auch das Problem, innerhalb eines nicht kreislosen Pseudographen einen möglichst großen Baum als Teilgraphen aufzuspüren.

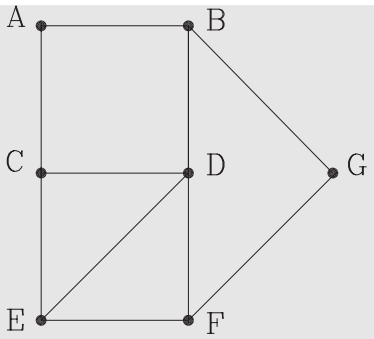
**Definition 1.2-6: Gerüst**
*$P$ sei ein Pseudograph. Ein Baum $B$, der Teilgraph von $P$ ist und jede Ecke von $P$ enthält, heißt ein Gerüst (oder spannender Baum) von $P$.*

**Gerüst**
**spannender Baum**

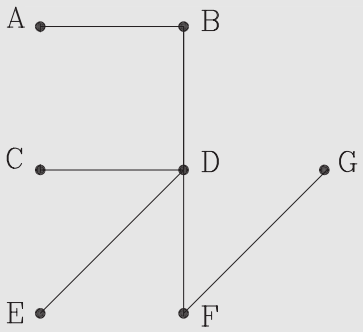
**Beispiel 1.2-9:**
Das Diagramm stellt den Pseudographen aus Beispiel 1.2-1 sowie ein Gerüst dar.




**Beispiel 1.2-10:**
Ein Datennetz mit Netzknoten $A, B, \ldots, G$ habe die im folgendem Diagramm gezeigte Graphenstruktur:




Soll nur sichergestellt werden, dass jeder Knoten mit jedem anderen kommunizieren kann (eventuell über Zwischenknoten), so müssen lediglich die Kanten eines Gerüsts genutzt werden. Das folgende Bild zeigt ein Gerüst:




Dieses spezielle Gerüst würde sich z. B. dann zur Nutzung anbieten, wenn $D$ im "Broadcasting"-Verfahren Meldungen an alle anderen Knoten schicken will.

Als nächstes werden zwei wichtige Eigenschaften von Bäumen bzw. Gerüsten nachgewiesen, die von der Anschauung her unmittelbar einleuchten: unter allen zusammenhängenden Graphen mit einer festen Eckenanzahl haben Bäume die wenigsten Kanten, und jeder zusammenhängende Pseudograph enthält ein Gerüst. Zunächst sind einige Vorüberlegungen nötig.

**Hilfssatz 1.2-1:** *Sei $F$ ein Kreis in dem zusammenhängenden Pseudographen $P$, $k$ eine in $F$ vorkommende Kante. Dann ist auch der Pseudograph $P' = P \backslash k$ zusammenhängend.*

**Beweis:** Es seien $a, b$ zwei beliebige Ecken von $P'$ bzw. $P$. Es ist zu zeigen, dass $a$ und $b$ in $P'$ verbindbar sind. Da $P$ zusammenhängend ist, gibt es in $P$ einen Weg von $a$ nach $b$. Benutzt dieser Weg die Kante $k$ nicht, so ist dies auch in $P'$ ein Weg von $a$ nach $b$. Enthält dieser Weg in $P$ von $a$ nach $b$ jedoch die Kante $k$, so kann statt der Kante $k$ der ganz in $P'$ enthaltene Rest des Kreises $F$ (außer $k$) benutzt werden, um nun eine Kantenfolge in $P'$ von $a$ nach $b$ zu erhalten, die einen Weg als Teilfolge enthalten muss.

**Satz 1.2-2:** *Jeder zusammenhängende Pseudograph besitzt ein Gerüst.*

**Beweis:** Der Beweis wird durch vollständige Induktion über die Anzahl der Kanten $|K|$ von Pseudographen $P = (E, K, v)$ geführt. Im Falle $|K| = 1$ ist der Satz offenbar richtig.
Nun wird vorausgesetzt, dass für alle Pseudographen mit $n$ $(n \geq 1)$ vielen Kanten die Behauptung bereits richtig sei, ferner sei ein zusammenhängender Pseudograph $P$ mit $|K| = n + 1$ gegeben. Ist $P$ ein Baum, so ist nichts zu zeigen, $P$ ist "sein eigenes Gerüst".
Enthält $P$ aber einen Kreis $F$, so gilt für eine beliebige Kante $k$ aus diesem Kreis, dass $P' = P \backslash k$ ein zusammenhängender Pseudograph (nach Hilfssatz 1.2-1) mit $n$ vielen Kanten ist, der nach Induktionsvoraussetzung ein Gerüst besitzt. Dieses Gerüst ist auch ein Gerüst von $P$.

**Definition 1.2-7: Endkante**

*Eine Ecke $a$ eines Pseudographen $P$ mit $\gamma(a, P) = 1$ heißt eine **Endecke** von $P$.* **Endecke**

*Eine Kante, die mit einer Endecke inzidiert, heißt **Endkante**.* **Endkante**

*Für je zwei Ecken $b, c$ von $P$ definiert man den **Abstand** $|b, c|_P$ zwischen $b$ und $c$ in $P$ wie folgt:* **Abstand**

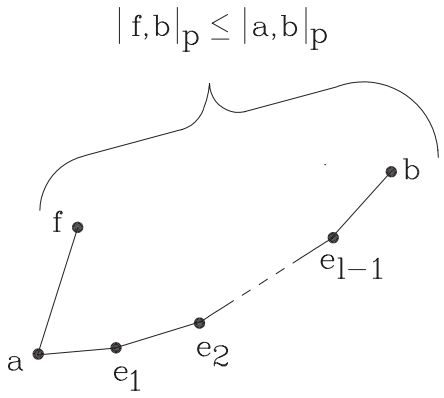
$$|b, c|_P = \begin{cases} 0, & \text{wenn } b = c \text{ ist;} \\ \infty, & \text{wenn } b \text{ und } c \text{ in } P \text{ nicht verbindbar sind;} \\ l, & \text{wenn } l \text{ die kürzestmögliche Länge eines} \\ & \text{Weges in } P \text{ von } b \text{ nach } c \text{ ist.} \end{cases}$$

**Hilfssatz 1.2-2:** *$P = (E, K, v)$ sei ein Baum. Dann gilt:*

*i.* *$P$ enthält mindestens zwei Endecken.*

*ii.* $|K| = |E| - 1$

**Beweis:** i. $a, b \in E$ seien derart gewählt, dass $|a, b|_P$ für $P$ maximal ist, d. h. es gibt keine zwei Ecken, die einen größeren Abstand voneinander haben. Wir zeigen, dass $a$ (und aus Symmetriegründen auch $b$) eine Endecke ist. $(a, k_1, e_1, k_2, e_2, \ldots, k_l, b)$ sei ein kürzester Weg in $P$ von $a$ nach $b$. Hätte $a$ außer $e_1$ eine weitere Nachbarecke $f$ (s. das Diagramm), so müsste es wegen $|f, b|_P \leq |a, b|_P$




einen kürzesten Weg von $f$ nach $b$ geben, der nicht durch die Ecke $a$ verläuft. Dann gäbe es jedoch einen Kreis in $P$, im Widerspruch zur Voraussetzung.

ii. Dies folgt nun leicht mit Induktion: Entfernen einer
Endecke und der zugehörigen Kante führt zu einem Teilbaum $P' = (E', K', v')$ mit $|E'| = |E| - 1, |K'| = |K| - 1$ und $|K'| = |E'| - 1$ (nach Induktionsvoraussetzung).

Damit kann nun folgendes gezeigt werden.

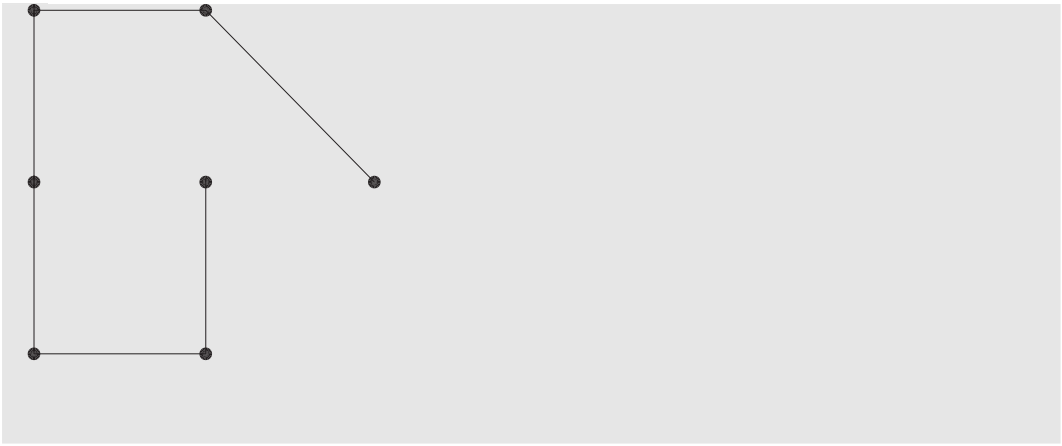
**Satz 1.2-3:** *$P = (E, K, v)$ sei ein zusammenhängender Pseudograph. Dann gilt $|K| \geq |E| - 1$, und es ist $|K| = |E| - 1$ genau dann, wenn $P$ ein Baum ist.*

**Beweis:** Die allgemeine Ungleichung $|K| \geq |E| - 1$ folgt sofort aus Satz 1.2-2 und Hilfssatz 1.2-2 ii), ferner auch die Gleichung $|K| = |E| - 1$ für einen Baum. Sei nun umgekehrt $|K| = |E| - 1$.
Wir nehmen an, $P$ sei kein Baum. Nach Hilfssatz 1.2-1 kann aus $P$ eine Kante $k$ entfernt werden, so dass $P' = P \backslash k$ zusammenhängend ist. In dem zusammenhängenden Pseudographen $P'$ mit Eckenmenge $E$ und Kantenmenge $K' = K \backslash \{k\}$ gilt also $|K'| < |E| - 1$.
Dies ist nun ein Widerspruch, denn nach Satz 1.2-2 müsste es in $P'$ ein Gerüst geben, welches nach Hilfssatz 1.2-2 $|E| - 1$ viele Kanten hat, die sämtlich in der Menge $K'$ enthalten sind.

**Folgerung 1.2-1:** *Alle Gerüste eines zusammenhängenden Pseudographen $P = (E, K, V)$ haben dieselbe Anzahl von Kanten (nämlich $|E| - 1$).*

**Beispiel 1.2-11:**

Im folgenden Diagramm ist ein weiteres Gerüst des Graphen aus Beispiel 1.2-10 dargestellt. Beide Gerüste haben $6$ Kanten.



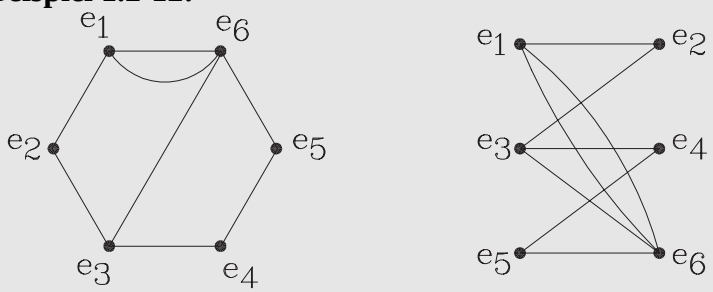
Als nächstes wird eine weitere interessante Klasse von Pseudographen betrachtet, die sich (allerdings erst auf den zweiten Blick) durch das Nicht-Enthalten gewisser Kreise charakterisieren lassen.

**Definition 1.2-8: Bipartiter Pseudograph**

**bipartiter (paarer) Pseudograph**

*Ein Pseudograph $P = (E, K, v)$ heißt **bipartit** (oder **paar**), wenn die Eckenmenge $E$ die disjunkte Vereinigung zweier nicht-leerer Teilmengen $E'$ und $E''$ ist derart, dass $E'$ und $E''$ kantenlose Untergraphen von $P$ aufspannen, wenn also alle Kanten von $P$ einen Endpunkt in $E'$ und einen in $E''$ haben.*

**Beispiel 1.2-12:**




Beide Diagramme stellen denselben Pseudographen dar. Im bipartiten Fall ist es üblich, wie hier im rechten Diagramm, die Mengen $E'$ und $E''$ (hier $\{e_1, e_3, e_5\}$ und $\{e_2, e_4, e_6\}$) auch optisch voneinander zu trennen.

**Satz 1.2-4:** *Ein Pseudograph $P = (E, K, v)$ ist genau dann bipartit, wenn $P$ keinen Kreis ungerader Länge enthält.*

**Beweis:** Ist $P$ bipartit, so kann $P$ sicher nur Kreise gerader Länge enthalten (wie im obigen Beispiel), denn für einen Kreis mit einer ungeraden Anzahl von Ecken ist es nicht möglich, dass diese Ecken abwechselnd zu zwei disjunkten Mengen $E'$ und $E''$ gehören.
Umgekehrt enthalte $P$ keinen Kreis ungerader Länge. Der Nachweis, dass $P$ bipartit ist, wird durch Induktion über $|K|$ (Anzahl der Kanten) geführt, wobei

für den Induktionsanfang $|K| = 0$ nichts zu zeigen bleibt. Als Induktionsvoraussetzung wird nun angenommen, dass die Behauptung für alle Pseudographen mit $n$ vielen Kanten richtig ist. Es sei $P$ ein Pseudograph mit $n+1$ vielen Kanten, der keine ungeraden Kreise enthält, ferner sei $k \in K$ eine beliebige Kante und $\tilde{P} := P \backslash k$. $\tilde{P}$ enthält keinen ungeraden Kreis, denn durch Löschen einer Kante werden keine neuen Kreise erzeugt. Da $\tilde{P}$ $n$ viele Kanten hat, ist $\tilde{P}$ nach Induktionsvoraussetzung bipartit, d. h. man hat eine Einteilung der Eckenmenge $E$ von $\tilde{P}$ (die auch die Eckenmenge von $P$ ist) in $E'$ und $E''$, so dass jede Kante von $K - \{k\}$ eine Ecke von $E'$ mit einer aus $E''$ verbindet. Gelingt es nun zu zeigen, dass von den Endpunkten $a$ und $b$ von $k$ einer zu $E'$ und der andere zu $E''$ gehört, so ist auch $P$ als bipartit nachgewiesen.
Es sind zwei Fälle zu betrachten.
Sind $a$ und $b$ in $\tilde{P}$ nicht verbindbar (d. h. sie gehören zu verschiedenen Komponenten von $\tilde{P}$), so kann über die Zerlegung $E'$ und $E''$ o.B.d.A. vorausgesetzt werden $a \in E'$ und $b \in E''$. Sind $a$ und $b$ in $\tilde{P}$ verbindbar, so muss jeder Weg von $a$ nach $b$ in $\tilde{P}$ eine ungerade Anzahl von Kanten enthalten, denn sonst hätte man (durch Hinzunahme von $k$) einen ungeraden Kreis in $P$. Anfangs- und Endknoten eines ungeraden Weges in $\tilde{P}$ gehören offenbar zu verschiedenen unter den Mengen $E'$ und $E''$. Damit ist gezeigt, dass $P$ bipartit ist.

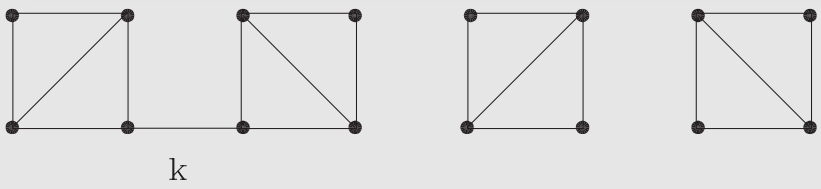
Zum Schluss dieses Abschnitts soll ein weiterer Begriff eingeführt werden, der anschaulich leicht zu fassen ist.

**Definition 1.2-9: Brücke**

**Brücke (Isthmus)** *Eine Kante $k$ des Pseudographen $P$ heißt **Brücke** (oder **Isthmus**) von $P$, wenn $k$ keine Endkante von $P$ ist und wenn die $k$ enthaltende Komponente von $P$ nach Wegnahme von $k$ (d. h. in $P \backslash k$) in zwei Komponenten zerfällt. Diese beiden Kom-*

**Blatt** *ponenten von $P \backslash k$ heißen die **Blätter** zur Brücke $k$.*

**Beispiel:**




Graph mit Brücke k  Blätter zur Brücke k

Ohne Beweis merken wir an, dass $k$ stets genau dann eine Brücke von $P$ ist, wenn $P$ keinen Kreis durch $k$ enthält und $k$ keine Endkante von $P$ ist. Auch der Beweis der folgenden Charakterisierung von Bäumen sei dem Leser überlassen.

**Satz 1.2-5:** *Für einen Multigraphen $P$ sind folgende Aussagen äquivalent:*

*i.* $P$ *ist ein Baum.*

*ii.* $P$ *ist zusammenhängend, und jede Kante von* $P$ *ist Endkante oder Brücke.*

*iii. Je zwei Ecken* $a$ *und* $b$ *sind durch genau einen Weg in* $P$ *miteinander verbunden.*

**Selbsttestaufgabe 1.2-1:**

*i. Erläutern Sie die Begriffe Kreis und Baum.*
*ii. Bestimmen Sie die Anzahl der verschiedenen Gerüste des vollständigen Graphen $K_4$ mit 4 Ecken.*

## 1.3 Kreise und Schnitte

**Definition 1.3-1: Spannender Wald**
*Ein Multigraph $M = (E, K, v)$ sei gegeben mit $n$ Ecken und $m$ Kanten, also $n = |E|$ und $m = |K|$. $M$ bestehe aus $k$ Komponenten.*

*Sind $M_1, \ldots, M_k$ die Komponenten von $M$ und $B_1, \ldots, B_k$ Gerüste, also $B_i$ jeweils ein Gerüst in $M_i$, so ist die Vereinigung $B = B_1 \cup \ldots \cup B_k$ ein k-Baum bzw. Wald. Da $B$ ganz $M$ aufspannt, wird $B$ auch als **spannender Wald** bezeichnet. Mithilfe von Folgerung 1.2-1 ist leicht einzusehen, dass $Bn - k$ viele Kanten hat. Die Zahl*

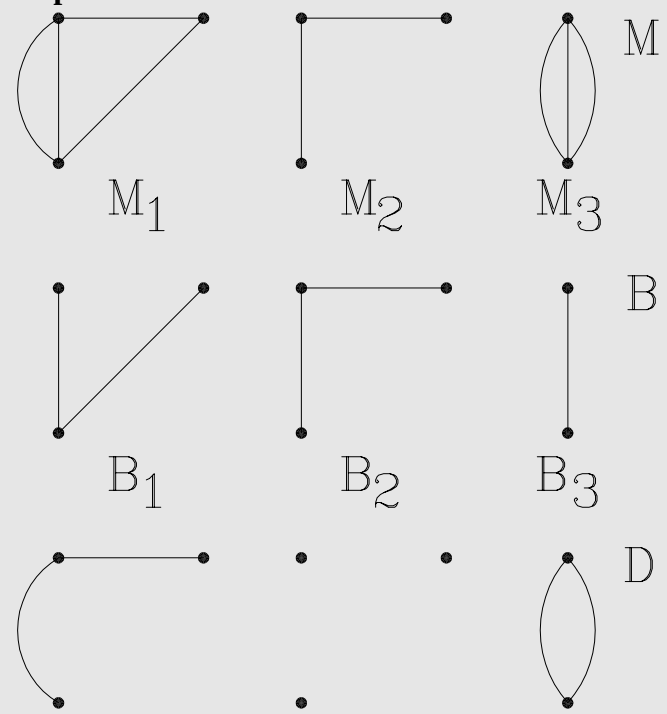
**spannender Wald**

$$\rho(M) = n - k$$

*heißt der **Rang** von $M$.*

**Rang**

**Beispiel 1.3-1:**




$n = 8$
$m = 9$
$k = 3$

$\varrho(M) = 5$

$\mu(M) = 4$

**Definition 1.3-2:** *In obigem Beispiel ist im dritten Diagramm der Teilgraph $D$ von $M$ dargestellt, der genau die nicht zu dem spannenden Wald $B$ gehörenden Kanten enthält, dies müssen natürlich $m - n + k$ viele sein.*

**spannender Ko-Wald**

*$D$ ist ein **spannender Ko-Wald**. (Das Wort "spannend" bezieht sich hier auf den Wald $B$.) Die Zahl*

$$\mu(M) = m - n + k$$

**Nullität**
**Rangabfall**

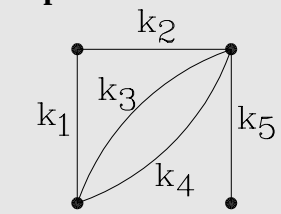
*heißt die **Nullität** (auch: Rangabfall) von $M$.*

Die Begriffe "spannender Wald" und "spannender Ko-Wald" sind zueinander dual. Es ist nicht so leicht einzusehen, wird sich aber bald zeigen, dass der nun einzuführende Begriff eines Schnittes in gewissem Sinne dual zu dem eines Kreises ist:

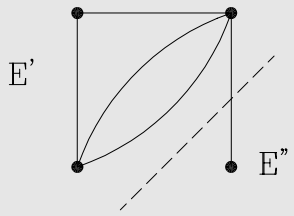
**Definition 1.3-3:** *Ist die Eckenmenge $E$ von $M$ die disjunkte Vereinigung zweier nicht-leerer Teilmengen $E'$ und $E''$, so heißt die Menge $S \subseteq K$ aller Kanten, von denen ein Endpunkt in $E'$ und einer in $E''$ liegt, ein **Schnitt** von $M$.*

**Schnitt**

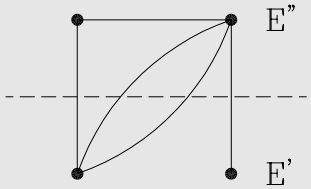
**Beispiel 1.3-2:**




Multigraph $M$




$\rightarrow$ zugehöriger Schnitt
$S_1 = \{k_5\}$




$\rightarrow$ zugehöriger Schnitt
$S_2 = \{k_1, k_3, k_4, k_5\}$

**Definition 1.3-4: Fundamentaler Schnitt**

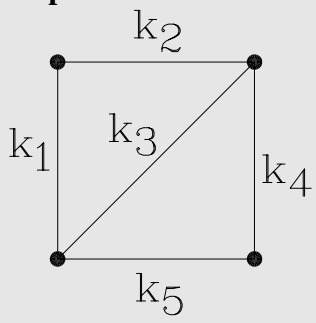
*Zusätzlich zu dem Multigraphen $M$ sei nun ein spannender Wald $B = B_1 \cup \ldots \cup B_k$ von $M$ vorgegeben. Eine Kante von $M$ gehört entweder zu $B$ und heißt dann ein **Zweig** von $M$ bezüglich $B$, oder die Kante gehört nicht zu $B$ und ist eine **Sehne** von $M$ bezüglich $B$.*

**Zweig**

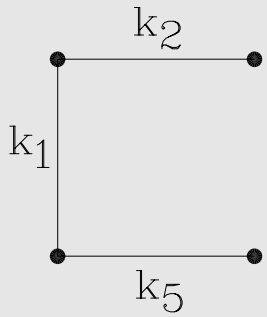
**Sehne**

*Offenbar hat man auf diese Weise $\varrho(M)$ viele Zweige und $\mu(M)$ viele Sehnen.*

**Beispiel 1.3-3:**




Graph $G$ mit $n = 4$, $m = 5$, $k = 1$;
$\varrho(G) = 3$, $\mu(G) = 2$.




Gerüst bzw. spannender Wald $B$;
$k_1, k_2, k_5$ sind Zweige,
$k_3$ und $k_4$ Sehnen.

Man beachte, dass eine andere Auswahl eines spannenden Baumes zu einer anderen Einteilung in Zweige und Sehnen führt, deren Anzahlen sind jedoch von dieser Auswahl unabhängig.

**Definition 1.3-5: Fundamentaler Kreis**

*Wir gehen wieder von $M$ und $B$ wie in Definition 1.3-4 aus. Fügen wir eine Sehne $s$ zu $B$ hinzu, so erhalten wir in dem neuen Teilgraphen genau einen Kreis, denn in $B$ gibt es zwischen den Ecken einer Sehne genau einen Weg, der mit der Sehne zusammen genau einen Kreis bildet. Dieser Kreis heißt* ***fundamentaler Kreis*** *der Sehne $s$. Die $\mu(M)$ vielen fundamentalen Kreise, die auf diese Weise entstehen, bilden eine* ***fundamentale Menge von Kreisen****.*

**fundamentaler Kreis**

**fundamentale Menge von Kreisen**

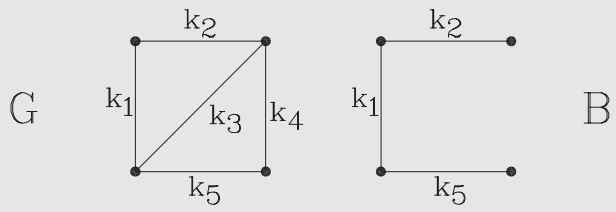
**Definition 1.3-6: Fundermentaler Schnitt**

*Dual führt jeder Zweig $z$ (von $M$ bezüglich $B$) eindeutig zu einem* ***fundamentalen Schnitt*** *des Zweiges $z$: die $z$ enthaltende Komponente $B_i$ von $B$ zerfällt bei Wegnahme der Brücke $z$ in Blätter $E'$ und $E''$. (Die Ecken der anderen Komponenten können o.B.d.A. $E'$ zugeschlagen werden.) Die $\varrho(M)$ vielen fundamentalen Schnitte, die so entstehen, bilden eine* ***fundamentale Menge von Schnitten****.*

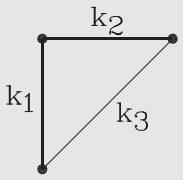
**fundamentaler Schnitt**

**fundamentale Menge von Schnitten**

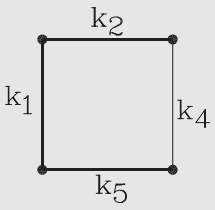
**Beispiel 1.3-4:**
Es wird wieder der Graph $G$ aus Beispiel 1.3-3 betrachtet, zunächst mit dem dort angegebenen Gerüst $B$:




Fundamentale Kreise:

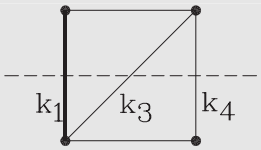



(der Sehne $k_3$)

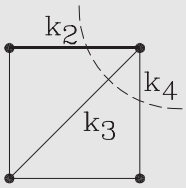



(der Sehne $k_4$)

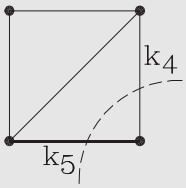
Fundamentale Schnitte:




(des Zweiges $k_1$)

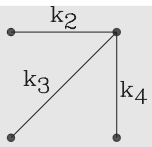



(des Zweiges $k_2$)

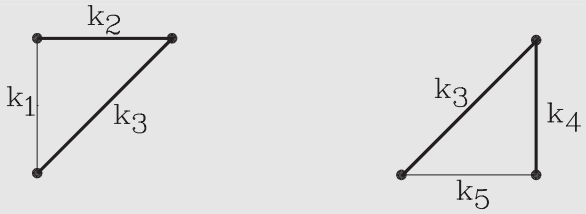



(des Zweiges $k_5$)

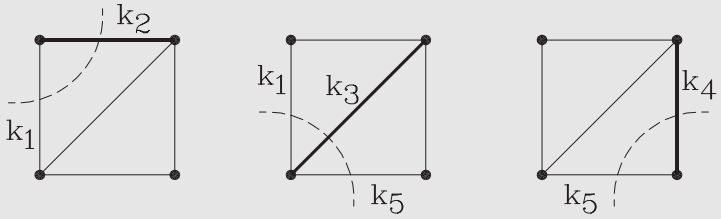
Ein anderes Gerüst ist z. B. folgendes $C$:




Dazu gehören diese beiden fundamentalen Kreise:




Entsprechend ergibt sich auch eine andere fundamentale Menge von Schnitten:




Eine wie sich bald zeigen wird wichtige Beobachtung über Kreise und Schnitte ist schließlich der folgende

**Satz 1.3-1:** *In einem Multigraphen hat jeder Schnitt mit jedem Kreis eine gerade Anzahl von Kanten gemeinsam.*

**Beweis:** Ein Kreis $C$ und ein Schnitt $S$ mit zugehöriger Einteilung in disjunkte Eckenmengen $E'$ und $E''$ seien gegeben. Gehört der Kreis $C$ ganz zu einem der von $E'$ bzw. von $E''$ aufgespannten Untergraphen, so hat $C$ mit $S$ offenbar keine Kanten gemeinsam, und die Behauptung stimmt.
Wir setzen nun also voraus, $C$ durchlaufe Ecken von $E'$ und von $E''$; $e \in E'$ sei eine zu $C$ gehörende Ecke. Durchläuft man nun, von $e$ ausgehend, einmal den Kreis $C$, wobei man wieder bei $e \in E'$ endet, so muss man für jeden Übergang von einer Ecke aus $E'$ zu einer Ecke aus $E''$ (d. h. die benutzte Kante liegt in $S$) auch einen solchen Übergang in umgekehrter Richtung haben, andernfalls könnte man nicht bei $e \in E'$ enden. Dies zeigt aber genau die Behauptung.

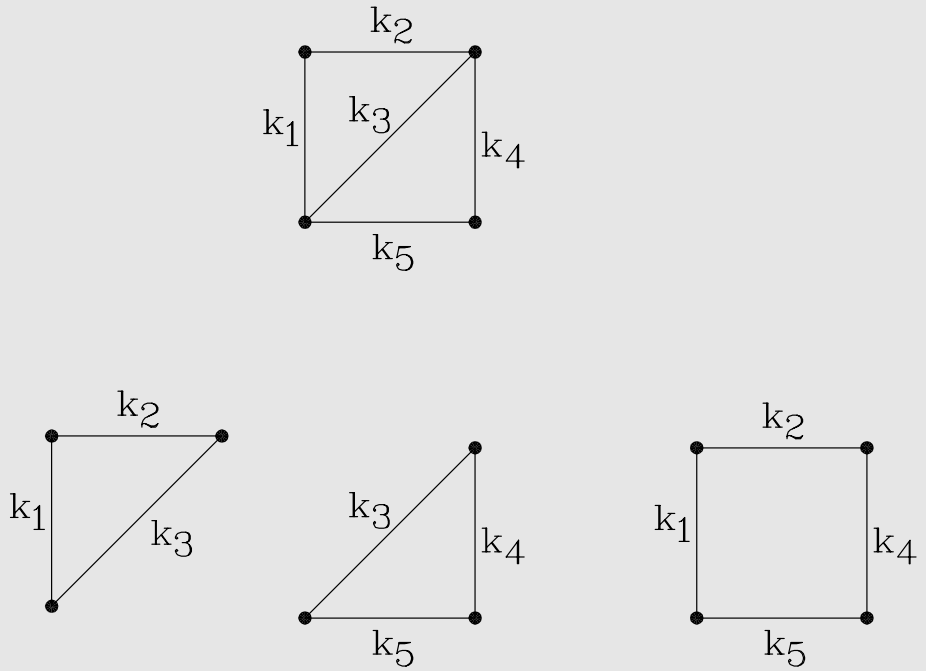
In den folgenden Abschnitten wird sich zeigen, dass sich die Struktur eines Multigraphen in hohem Maße mit den Mitteln der Linearen Algebra beschreiben lässt.

Dies ist besonders auch für die Anwendungen der Graphentheorie von Bedeutung, denn die Lineare Algebra ist ein gut untersuchtes mathematisches Gebiet und bietet für viele Probleme bewährte Rechenverfahren.
Der ganze weitere Verlauf des Kurses wird durch die Sprache und die Denkweisen der Linearen Algebra stark geprägt sein. Um die Eigenschaften von Kreisen besser darstellen zu können, wird es dabei günstig sein, einen Kreis in einem Pseudographen einerseits als Teilgraphen aufzufassen, ihn aber andererseits einfach mit der Menge seiner Kanten zu identifizieren.

**Potenzmenge der Kantenmenge**

Fasst man die **Potenzmenge** $P(K)$ **der Kantenmenge** $K$ als Vektorraum über dem Körper $\mathsf{IF}_2$ auf, so ist die Summe zweier Kreise, die genau eine Kante gemeinsam haben, wieder ein Kreis.

**Beispiel 1.3-5:**




Kreis $C_1$ $= \{k_1, k_2, k_3\}$ | Kreis $C_2$ $= \{k_3, k_4, k_5\}$ | Kreis $C_1 + C_2$ $= \{k_1, k_2, k_4, k_5\}$

Die soeben gewonnene Erkenntnis soll nun verallgemeinert werden.
Wir setzen (zu gegebenem Multigraphen $M$)

$$\mathbf{C} := \{C \subseteq K \mid C \text{ ist eine Vereinigung kantendisjunkter Kreise}\}$$

und $\mathbf{C}^* := \mathbf{C} \cup \{\emptyset\}$.
Man beachte, dass kantendisjunkte Kreise nicht notwendig auch knotendisjunkt sein müssen.

Es soll nun gezeigt werden, dass $\mathbf{C}^*$ ein Untervektorraum von $\mathbf{P}(K)$ ist. Das Entscheidende dazu steckt in dem folgenden

**Hilfssatz 1.3-1:** $\mathbf{C}^*$ *besteht aus allen Teilgraphen von* $M$*, in denen jede Ecke einen geraden Grad von mindestens zwei hat.*

**Beweis:** Zunächst folgt aus der Definition unmittelbar, dass jede Ecke eines $C \in \mathbf{C}^*$ einen geraden Grad hat. Umgekehrt sei nun ein Teilgraph $T \neq \emptyset$ mit dieser Eigenschaft gegeben; es ist $T \in \mathbf{C}$ zu zeigen. Dies folgt leicht mit Induktion: da $T$ kein Baum ist (dann müsste es eine Endecke geben), gibt es in $T$ einen Kreis. Nun sind mit jeder Ecke eines Kreises genau zwei seiner Kanten inzident. Entfernen der Kanten dieses Kreises aus $T$ führt zu einem Teilgraphen $T'$ von $T$ und damit auch von $M$, in dem wieder jede Ecke einen geraden Grad hat. Dabei werden isolierte Ecken (vom Grad Null) weggelassen, d. h. wenn $T'$ nicht der leere Graph ist, so hat jede Ecke von $T'$ wieder einen geraden Grad von mindestens zwei. Die analoge Argumentation für $T'$ führt zu $T''$ etc. -auf diese Weise kommt man durch dauerndes Entfernen von Kreisen und isolierten Ecken zum leeren Graphen, womit die Behauptung folgt.

**Satz 1.3-2:** $\mathbf{C}^*$ *ist ein Untervektorraum der Potenzmenge* $\mathbf{P}(K)$.

**Beweis:** Der Beweis dieses Satzes ergibt sich jetzt mit Hilfssatz 1.3-1 aus der Beobachtung, dass die Vektorraumsumme zweier Kantenmengen, in deren zugehörigen Teilgraphen jede Ecke einen geraden Grad besitzt, auch wieder diese Eigenschaft hat.

In dem Multigraphen $M$ sei nun wieder ein spannender Wald $B$ gegeben. Es gilt der

**Satz 1.3-3:** *Die* $\mu(M)$ *vielen fundamentalen Kreise von* $M$ *(bezüglich* $B$*) bilden eine Basis des Vektorraums* $\mathbf{C}^*$.

**Beweis:** $s_1, \ldots, s_{m-n+k}$ seien die Sehnen und $C_1, \ldots, C_{m-n+k}$ die zugehörigen fundamentalen Kreise. Die lineare Unabhängigkeit folgt nun daraus, dass die Sehne $s_i$ nur in $C_i$ und keinem der anderen $C_j$ enthalten ist, folglich kann $C_i$ nicht als Summe anderer fundamentaler Kreise dargestellt werden.

Es bleibt zu zeigen, dass jedes Element von $\mathbf{C}^*$ Summe von fundamentalen Kreisen ist. Sei $C \in \mathbf{C}^*$, $C \neq \emptyset$. Es seien $s_{i1}, \ldots, s_{ir}$ die in $C$ enthaltenen Sehnen. Wir behaupten, dass $C = C_{i1} + \ldots + C_{ir}$ ist. Da $C$ und $C_{i1} + \ldots + C_{ir}$ genau die gleichen Sehnen enthalten (nämlich $s_{i1}, \ldots, s_{ir}$), enthält $C' = C + (C_{i1} + \ldots + C_{ir})$ überhaupt keine Sehnen. Andererseits ist aber $C' \in \mathbf{C}^*$, womit $C' = \emptyset$ und die Behauptung folgt.

**Folgerung 1.3-1:** *Für jeden Multigraphen* $M$ *gilt*

$$|\mathbf{C}| = 2^{\mu(M)} - 1.$$

Wir wenden uns nun wieder den Schnitten zu und setzen (zu gegebenem Multigraphen $M$)

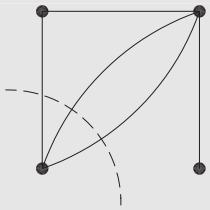
$$\mathbf{S} := \{\mathbf{S} \subseteq \mathbf{K} | \text{S ist ein Schnitt}\}$$

und

$$\mathbf{S}^* := \mathbf{S} \cup \{\emptyset\}.$$

Auch $\mathbf{S}^*$ ist gegenüber der Summenbildung abgeschlossen.

**Beispiel 1.3-6:**
Für die Schnitte $S_1$ und $S_2$ aus Beispiel 1.3-2 ist auch $S_3 = S_1 + S_2 = \{k_1, k_3, k_4\}$ ein Schnitt, wie folgendes Bild zeigt:



Es werden nun ohne Beweise einige Sätze und Folgerungen über die Menge $\mathbf{S}^*$ aufgezählt, wobei die Ähnlichkeit der Aussagen zu denen über $\mathbf{C}^*$ auffällt:

**Satz 1.3-4:**
- $\mathbf{S}^*$ *ist ein Untervektorraum von* $\mathbf{P}(K)$.
- *Ist irgendein spannender Wald* $B$ *in* $M$ *gegeben, so bilden die* $\rho(M)$ *vielen fundamentalen Schnitte eine Basis von* $\mathbf{S}^*$.
- $|\mathbf{S}| = \mathbf{2}^{\rho(\mathbf{M})} - \mathbf{1}$.

Es soll noch einmal darauf hingewiesen werden, dass die Aussagen über die Vektorräume $\mathbf{C}^*$ und $\mathbf{S}^*$ und deren Elementeanzahlen nur von $M$ abhängen; die Basis der fundamentalen Kreise bzw. Schnitte hängt allerdings von der Wahl eines spannenden Waldes $B$ ab.

Eine interessante Beobachtung, die an späterer Stelle Verwendung finden wird, ist es nun, dass die Untervektorräume $\mathbf{C}^*$ und $\mathbf{S}^*$ von $\mathbf{P}(K)$ orthogonal sind:

**Satz 1.3-5:**

$$\mathbf{C}^* \perp \mathbf{S}^*$$

**Beweis:** Mit Hilfe von Satz 1.3-1 folgt, dass ein beliebiges Element $D \in \mathbf{C}^*$ mit einem beliebigen $T \in \mathbf{S}^*$ eine gerade Anzahl von Kanten gemeinsam hat. Dies hat $D \cdot T = 0$ zur Folge.

Die bisher beschriebene Dualität zwischen Kreisen und Schnitten spiegelt sich auch in deren Verhältnis zu spannenden Bäumen bzw. Ko-Bäumen. (Der Einfachtheit halber sei $M$ als zusammenhängend vorausgesetzt).
Ohne Beweis merken wir an:

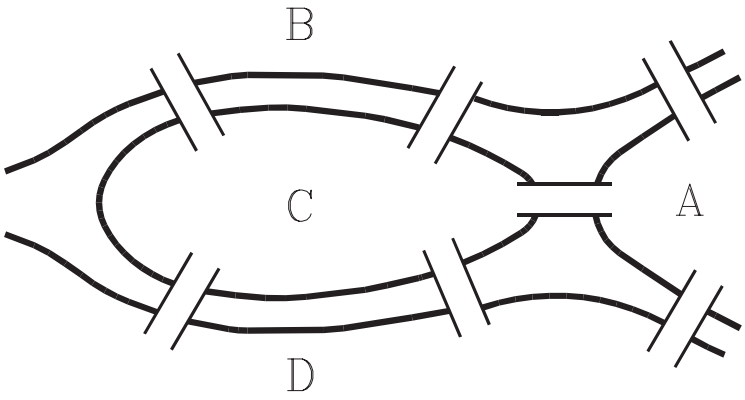
Eine Kantenmenge $S \subseteq K$ ist genau dann ein Schnitt, wenn $S$ eine minimale Teilmenge von $K$ ist mit der Eigenschaft, von jedem spannenden Baum mindestens einen Zweig zu enthalten.

Eine Kantenmenge $C \subseteq K$ ist genau dann ein Kreis, wenn $C$ eine minimale Teilmenge von $K$ ist mit der Eigenschaft, von jedem spannenden Ko-Baum mindestens eine Sehne zu enthalten.

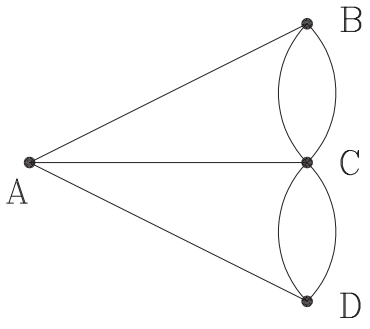
Zum Schluss dieses Abschnittes soll Hilfssatz 1.3-1 zum Anlass für einige kurze Bemerkungen zu Eulerschen und Hamiltonschen Graphen und den Ursprüngen der Graphentheorie genommen werden.

**Königsberger Brückenproblem**

Das berühmte **Königsberger Brückenproblem** lautet folgendermaßen. In dem durch Königsberg fließenden Pregel gab es zwei Inseln, die durch sieben Brücken miteinander und mit den Ufern (wie im untenstehendem Bild angedeutet) verbunden waren. Es stellte sich die Frage, ob es möglich sei, an irgendeinem Punkt in einem der Gebiete $A, B, C, D$ loszugehen, jede Brücke genau einmal zu passieren und zum Ausgangspunkt zurückzukehren.




Der Mathematiker Euler fand im Jahre 1736 heraus, dass das nicht möglich ist. Dies wird allgemein als Ursprung der Graphentheorie angesehen, denn Eulers Begründung war, dass der im folgendem Diagramm dargestellte Multigraph keinen geschlossenen Kantenzug enthält, der jede Kante genau einmal benutzt.



Aus diesen Gründen nennt man einen geschlossenen Kantenzug in einem Pseudographen, der jede Kante enthält, einen **Eulerschen Kantenzug**; ein **Eulerscher Pseudograph** ist ein solcher, der einen Eulerschen Kantenzug besitzt, der also aus einem geschlossenem Kantenzug besteht.

**Eulerscher Kantenzug**

**Eulerscher Pseudograph**

Es gilt nun folgender

**Satz 1.3-6:** *Für einen zusammenhängenden Pseudographen $P = (E, K, v)$ sind die folgenden Aussagen äquivalent:*

*i.* $P$ *ist Eulersch.*

*ii.* $P$ *(bzw.* $K$*) ist eine Vereinigung kantendisjunkter Kreise*

*iii. Jede Ecke von* $P$ *hat einen geraden Grad.*

**Beweis:** Am leichtesten ist es, die Implikationen i) $\rightarrow$ iii), iii) $\rightarrow$ ii) und ii) $\rightarrow$ i) nachzuweisen.
i) $\rightarrow$ iii): Es sei $C$ ein Eulerscher Kantenzug von $P$. Bei einem Durchlaufen des Kantenzugs $C$ trägt jedes Vorkommen einer Ecke $e$ den Summanden 2 zum Grad von $e$ bei. Da jede Kante von $P$ genau einmal durchlaufen wird, muss jede Ecke $e$ geraden Grad haben.
iii) $\rightarrow$ ii): Dies ist eine direkte Folgerung aus Hilfssatz 1.3-1.
ii) $\rightarrow$ i): Es sei $C$ ein Kreis in einer Zerlegung der Kantenmenge $K$ in kantendisjunkte Kreise. Ist $C$ ein Eulerscher Kantenzug, so ist nichts zu beweisen. Andernfalls gibt es, da $P$ zusammenhängend ist, einen weiteren Kreis $C'$, der mit $C$ eine Ecke $e$ gemeinsam hat. Wir können nun o.B.d.A. $e$ als Anfangs- und Endpunkt beider Kreise wählen und so, von $e$ ausgehend, zuerst $C$ und anschließend $C'$ durchlaufen. Indem die Argumentation ähnlich fortgeführt wird, erhalten wir schließlich einen Eulerschen Kantenzug.

Der Vollständigkeit halber soll noch angemerkt werden, dass ein Pseudograph, in dem es einen alle Ecken enthaltenden Kreis gibt, **Hamiltonsch** genannt wird. Eine einfache Charakterisierung **Hamiltonscher Pseudographen**(etwa ähnlich Satz 1.3-4 für Eulersche Pseudographen) wurde bisher nicht gefunden.

**Hamiltonscher Pseudograph**

**Selbsttestaufgabe 1.3-1:**

*Man überlege sich, dass* $\mu(M) = 0$ *für jeden Wald* $M$ *(und keine anderen Multigraphen) gilt.*

# 2 Darstellung von Graphen

## 2.1 Diagramme und Planarität

Schon im ersten Abschnitt war von Diagrammen - also zeichnerischen Darstellungen von Pseudographen - die Rede. Besonders übersichtlich ist ein solches Diagramm, wenn sich die gezeichneten Linien nicht schneiden. Für Graphen, die durch ein solches Diagramm in der Zeichenebene darstellbar sind, wurde folgender Begriff geprägt:

**Definition 2.1-1: Planare Darstellung**

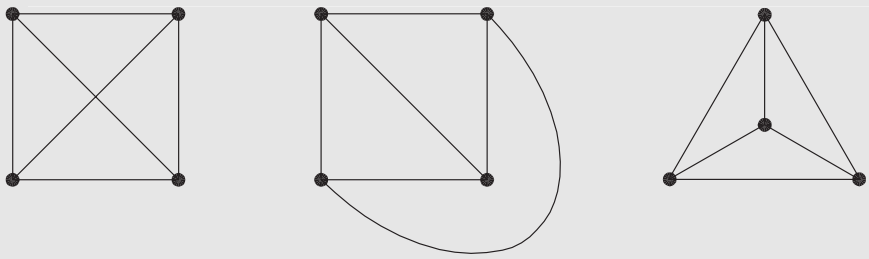
**planarer Graph**

*Ein Graph $G = (E, K, v)$ heißt **planar**, wenn er so durch Punkte und Linien der Zeichenebene dargestellt werden kann, dass sich die Linien nicht überschneiden;*

**planare Darstellung**

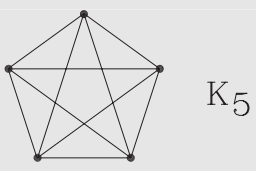
*ein solches Diagramm nennt man eine **planare Darstellung** von $G$.*

**Beispiel 2.1-1:**
Die folgenden Diagramme zeigen den vollständigen Graphen $K_4$ mit $4$ Ecken; das zweite und dritte bieten planare Darstellungen und zeigen, dass $K_4$ planar ist.



Man beachte: nicht jedes Diagramm eines planaren Graphen ist eine planare Darstellung.

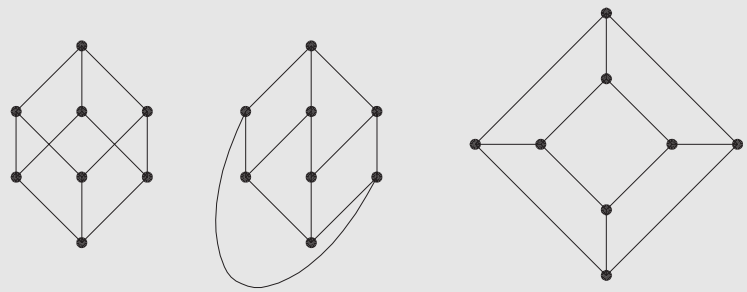
**Beispiel 2.1-2:**
Der Graph $K_5$ ist nicht planar. Dies zu beweisen, ist allerdings nicht ganz einfach.



Das dritte Diagramm in Beispiel 2.1-1 ist eine planare Darstellung, in der jede Kante als gerade Strecke gezeichnet ist. Man kann beweisen (doch dies würde hier zu weit führen), dass es ein solches Diagramm für einen planaren Graphen immer gibt.

Es leuchtet ein, dass es für einen gegebenen Graphen durchaus schwierig festzustellen sein kann, ob er planar ist oder nicht - die Frage ist, ob es unter den im Prinzip unendlich vielen möglichen Diagrammen eine planare Darstellung gibt. Nun kann man sich zwar überlegen, dass es nur endlich viele "wesentlich verschiedene" Diagramme gibt, jedoch bleibt nach wie vor das Problem, wie man die Frage nach der Planarität am schnellsten beantworten kann. Wir werden auf dieses Problem an späterer Stelle zurückkommen, wenn von der Bewertung von Algorithmen die Rede ist.

**Beispiel 2.1-3:**
Der Leser versuche zunächst selbst, eine planare Darstellung des Graphen aus dem ersten Diagramm unten zu finden. Anschließend überzeuge er sich, dass es sich im zweiten und dritten Bild um solche Darstellungen handelt.

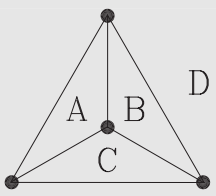


**Definition 2.1-2: Gebiete**

**Gebiete**

*Eine planare Darstellung teilt die Zeichenebene in **Gebiete** ein, wenn die gezeichneten Kanten als Grenzlinien aufgefasst werden.*

**Beispiel 2.1-4:**
In der folgenden planaren Darstellung des $K_4$ sind die Gebiete mit $A, B, C, D$ bezeichnet. Zu beachten ist, dass auch der außerhalb des Graphen liegende "Rest der Zeichenebene" als Gebiet gezählt wird.



**Eulersche Polyederformel**

Für die Anzahl der Gebiete in einer planaren Darstellung gilt der folgende Zusammenhang, die sogenannte **Eulersche Polyederformel**.

**Satz 2.1-1:** *Für eine planare Darstellung eines zusammenhängenden Graphen mit $n$ Ecken, $m$ Kanten und $r$ Gebieten gilt stets*

$$n - m + r = 2.$$

**Beweis:** Die $r - 1$ vielen Kreise, die die inneren Gebiete in dem Diagramm umschliessen, werden die Maschen der Darstellung genannt. Es wird nun gezeigt, dass die Maschen eine Basis von $\mathbf{C}^*$ bilden. Mit Satz 1.3-3 folgt dann $r - 1 = m - n + 1$ und damit die Behauptung.

Es seien $C_1, \ldots, C_{r-1}$ die Maschen, $G_1, \ldots, G_{r-1}$ die einzelnen von diesen Maschen eingeschlossenen Gebiete. Eine (im Vektorraum $\mathbf{C}^*$ gebildete) Summe irgendwelcher dieser Maschen umschließt stets genau die zu den Summanden gehörenden Gebiete, z. B. umschließt $C_1 + C_2 + C_3$ die Gebiete $G_1$, $G_2$ und $G_3$. Dies bedeutet auch, dass keine Masche als Linearkombination anderer Maschen darstellbar ist, $C_1, \ldots, C_{r-1}$ sind folglich linear unabhängig. Nun sei $C$ ein beliebiges Element von $\mathbf{C}^*$. Die von $C$ umschlossenen Gebiete seien $G_{i_1}, \ldots, G_{i_s}$. Es ist dann $C = C_{i_1} + \ldots, +C_{i_S}$. Damit ist $\{C_1, \ldots, C_{r-1}\}$ als Basis von $\mathbf{C}^*$ nachgewiesen, und der Beweis ist erbracht.

Der hier angedeutete algebraische Beweis der Eulerschen Polyederformel entspricht natürlich nicht der zuerst gefundenen Argumentation. Vielmehr verknüpft die Formel ursprünglich die Zahlen der Ecken, Kanten und Seitenflächen eines konvexen Polyeders im Raum und kann auch zur Bestimmung der $5$ regulären Polyeder genutzt werden.

Die Eulersche Polyederformel kann ausgenutzt werden, um die folgende notwendige Bedingung für die Planarität eines Graphen abzuleiten:

**Satz 2.1-2:** *Ein zusammenhängender planarer Graph $G$ mit $n$ Ecken $(n \geq 3)$ hat höchstens $3n - 6$ viele Kanten.*

**Beweis:** Der Beweis wird durch Induktion über die Anzahl der in $G$ enthaltenen Brücken geführt. Zunächst enthalte $G$ keine Brücke. $G_1, \ldots, G_r$ seien die Gebiete einer gegebenen planaren Darstellung, für $1 \leq i \leq r$ sei $k_{G_i}$ die Anzahl der das Gebiet $G_i$ begrenzenden Kanten. Da jede Kante von $G$ zu genau zwei Gebieten gehört, gilt für die Anzahl $m$ der Kanten

$$2m = k_{G_1} + \ldots + k_{G_r} \geq 3r,$$

denn es ist stets $k_{G_i} \geq 3$. Mit der Eulerschen Polyederformel folgt daraus

$$\begin{aligned} & 3(m - n + 2) && \leq 2m \\ \text{oder} \quad & m && \leq 3n - 6, \end{aligned}$$

was zu beweisen war.
Nun enthalte $Gs + 1$ viele Brücken, und die Richtigkeit des Satzes werde für

alle Graphen mit höchstens $s$ vielen Brücken angenommen. Wir wählen eine Brücke $k_0$ in $G$. Durch Entfernen von $k_0$ zerfällt $G$ in zwei planare Teile $G_1$ und $G_2$ mit $n_1$ bzw. $n_2$ vielen Ecken und $m_1$ bzw. $m_2$ vielen Kanten.

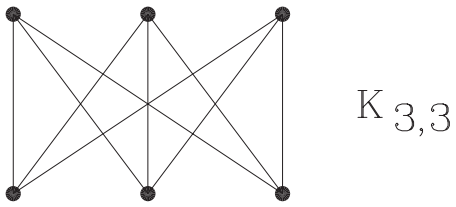
$$\begin{array}{lll} \text{Es ist} & n & = n_1 + n_2 \\ \text{und} & m & = m_1 + m_2 + 1. \end{array}$$

Da $G_1$ und $G_2$ höchstens $s$ viele Brücken enthalten, gilt nach

$$\begin{array}{lll} \text{Induktionsannahme} & m_1 & \leq 3n_1 - 6 \\ \text{und} & m_2 & \leq 3n_2 - 6. \end{array}$$

Mit den oberen Gleichungen folgt daraus $m \leq 3n - 11$ und somit erst recht $m \leq 3n - 6$. Damit ist der Satz insgesamt bewiesen.

Mit Satz 2.1-2 ist sofort klar, dass $K_5$ nicht planar ist, denn der Graph hat $10$ Kanten, dürfte aber höchstens $3 \cdot 5 - 6 = 9$ Kanten haben.




Für den hier dargestellten Graphen $K_{3,3}$, der ebenfalls nicht planar ist, greift Satz 2.1-2 allerdings nicht. Hier müssten kompliziertere Sätze herangezogen werden, auf die nicht eingegangen werden soll.

Ohne Beweis soll nun der **Satz von Kuratowski** (aus dem Jahre 1930) angegeben werden, welcher besagt, dass es im wesentlichen immer an einem der Graphen $K_5$ oder $K_{3,3}$ "liegt", wenn irgendein Graph nicht planar ist. Ein Beweis kann z. B. in dem zitierten Buch von R. Gould nachgelesen werden. Zur genauen Formulierung des Satzes muss allerdings der Begriff der Homöomorphie eingeführt werden:

**Satz von Kuratowski**

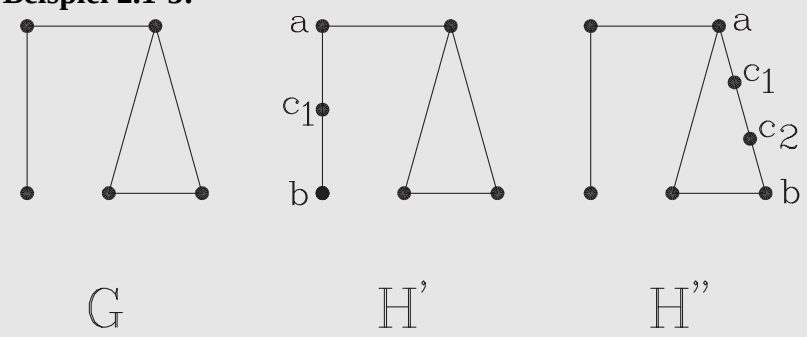
**Definition 2.1-3: Unterteilung**

*Eine **Unterteilung** eines Graphen $G$ ist ein Graph $H$, der durch ein- oder mehrfache Anwendung der folgenden Operation aus $G$ erhalten werden kann: eine Kante $k = \{a, b\}$ wird durch einen Weg $(a, c_1, \ldots c_n, b)$ ersetzt, wobei alle $c_i$ neue Ecken sind. Zwei Graphen $H'$ und $H''$ heißen **homöomorph**, wenn sie Unterteilungen desselben Graphen $G$ sind.*

**Unterteilung**

**homöomorph**

**Beispiel 2.1-5:**




$H'$ und $H''$ sind Unterteilungen von $G$ und somit homöomorph.

**Satz 2.1-3:** ***von Kuratowski***

*Ein Graph $G$ ist genau dann planar, wenn er keinen zu $K_5$ oder $K_{3,3}$ homöomorphen Teilgraphen enthält.*

Der Satz von Kuratowski liefert ein praktisches Verfahren für einen Planaritätstest von Graphen: man muss alle Teilgraphen betrachten und auf Homöomorphie zu $K_5$ oder $K_{3,3}$ hin überprüfen. Dieses Verfahren ist jedoch sehr aufwendig. Auf das Problem wird an späterer Stelle noch einmal eingegangen.

Der Abschnitt über Planarität soll nicht abgeschlossen werden ohne einen Hinweis darauf, dass beim VLSI-Layout Problemstellungen auftreten, bei denen z. T. die Kenntnisse über planare Graphen Anwendung finden.

**Selbsttestaufgabe 2.1-1:**

*Erläutern Sie den Unterschied zwischen den Begriffen "planarer Graph" und "planare Darstellung eines Graphen".*

## 2.2 Matrizen

Die Darstellung von Pseudographen durch Matrizen hat zwei große Vorteile:

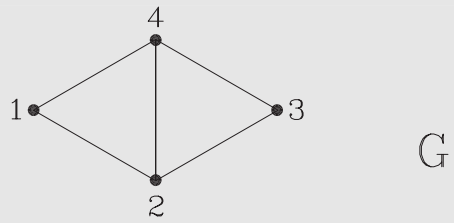
- sie ist "rechnerfreundlich", d. h. bietet sich an, wenn ein Algorithmus konkret für einen Graphen auf einem Rechner durchlaufen werden soll;
- sie bietet die Möglichkeit, mathematische Eigenschaften von Matrizen für die Graphentheorie zu nutzen.

Ein Nachteil gegenüber der Darstellung durch ein Diagramm liegt allerdings auf der Hand: beim Nachdenken über Graphen sind Diagramme sehr anregend für die menschliche Intuition, was sich von Matrizen nicht sagen lässt.

**Definition 2.2-1: Darstellung durch Listen**
*Eine Vorform der Matrizendarstellung besteht darin, die Ecken und Kanten eines Pseudographen einfach aufzuzählen. Man spricht dann von einer* ***Darstellung durch Listen****, die hier nur für Graphen erklärt werden soll. Der Einfachheit halber werden die Ecken des Graphen im folgenden Beispiel mit natürlichen Zahlen* $1, 2, \ldots, n$ *bezeichnet.*

**Darstellung durch Listen**

**Beispiel 2.2-1:**




Die **Kantenliste** eines Graphen $G$ wird spezifiziert durch

**Kantenliste**

- die Angabe von $n$;
- die Angabe der Liste der Kanten, etwa als Folge von Eckenpaaren.

Der Graph $G$ aus Beispiel 2.2-1 ist demnach beschrieben durch $n = 4$ und die Liste $(1,2), (1,4), (2,3), (2,4), (3,4)$.

Eine andere Möglichkeit ist die Darstellung durch eine **Inzidenzliste**, zu der außer der Angabe von $n$ noch die Aufzählung von $n$ Listen $A_1, \ldots, A_n$ gehört, wobei $A_i$ aus den Nachbarn des Punktes $i$ besteht.

**Inzidenzliste**

Der Graph $G$ aus Beispiel 2.2-1 wird wie folgt durch eine Inzidenzliste dargestellt:
$n = 4$; $A_1 : 2, 4$; $A_2 : 1, 3, 4$; $A_3 : 2, 4$; $A_4 : 1, 2, 3$.
Welche Art von Liste (oder Matrix) die günstigste ist, hängt natürlich von der konkreten Anwendung ab. Z.B. benötigen Kantenlisten zwar wenig Speicherplatz, sind aber sehr unhandlich, wenn man etwa die Nachbarn einer Ecke heraussuchen will.

Als nächstes wird die Adjazenzmatrix eingeführt, die nur für einen Graphen (und nicht allgemein für einen Pseudographen) gebildet werden soll.

**Definition 2.2-2:** *Ein Graph* $G = (E, K)$ *sei gegeben mit* $E = \{e_1, \ldots, e_n\}$ *und* $k = \{e_i, e_j\}$ *(mit geeigneten* $i, j$*) für* $k \in K$*. Die* ***Adjazenzmatrix*** $A(G) = (a_{ij})$ *von* $G$ *ist die* $n \times n$*-Matrix mit folgenden Einträgen:*

**Adjazenzmatrix**

$$a_{ij} = \begin{cases} 1, & \textit{falls } \{e_i, e_j\} \in K \\ 0, & \textit{sonst.} \end{cases}$$

*Statt* $A(G)$ *wird im folgenden oft nur* $A$ *geschrieben. Wie man sofort sieht, ist die Adjazenzmatrix eines Graphen stets symmetrisch und enthält insofern eine gewisse Redundanz, die man sich bei der Eingabe eines Graphen in einen Rechner durch eine Adjazenzmatrix zunutze machen kann.*

Für den Graphen $G$ aus Beispiel 2.2-1 ergibt sich die Adjazenzmatrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

Der folgende Satz liefert eine interessante Eigenschaft der Adjazenzmatrix.

**Satz 2.2-1:** *Ist $A$ die Adjazenzmatrix eines Graphen $G$, so liefert der $(i,j)$-Eintrag $a_{ij}^{(r)}$ der $r$-ten Potenz $A^r$ die Anzahl der Kantenfolgen von Länge $r$, die in $G$ die Ecken $e_i$ und $e_j$ verbinden.*

**Beweis:** Der Beweis wird durch Induktion über $r$ geführt. Für $r = 1$ ist die Aussage des Satzes offenbar richtig.
Nun sei die Behauptung als Induktionsannahme für $r-1$ (mit $r \geq 2$) richtig. Es gilt dann

$$a_{ij}^{(r)} = \sum_{k=1}^{n} a_{ik}^{(r-1)} \cdot a_{kj}$$

wobei $a_{ik}^{(r-1)}$ die Anzahl der Kantenfolgen von Länge $r-1$ zwischen $e_i$ und $e_k$ ist. Demnach ist für jedes $k$ das Produkt $a_{ik}^{(r-1)} \cdot a_{kj}$ die Anzahl der Kantenfolgen von Länge $r$ von $e_i$ nach $e_j$, deren letzte Kante $\{e_k, e_j\}$ ist. Da mit obiger Summe alle Alternativen für $e_k$ durchlaufen werden, folgt die Behauptung auch für $r$, d. h. der Satz ist insgesamt bewiesen.

**Beispiel 2.2-2:**
Für die Adjazenzmatrix $A$ des Graphen aus Beispiel 2.2-1 gilt

$$A^2 = \begin{pmatrix} 2 & 1 & 2 & 1 \\ 1 & 3 & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 1 & 2 & 1 & 3 \end{pmatrix}$$

Es ist z. B. $a_{11}^{(2)} = 2 : (1,4,1)$ und $(1,2,1)$ beschreiben die beiden möglichen Kantenfolgen von Länge 2, die die Ecke 1 mit sich selbst verbinden.

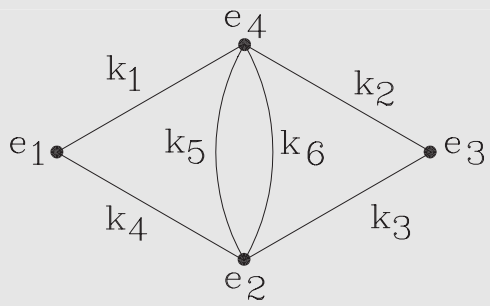
Als nächstes wird die Inzidenzmatrix eingeführt.

**Inzidenzmatrix**

**Definition 2.2-3:** *Es sei ein Multigraph $M = (E, K, v)$ gegeben. Die Mengen $E$ und $K$ seien duchnumeriert, $E = \{e_1, \ldots, e_n\}$ und $K = \{k_1, \ldots, k_m\}$. Die **Inzidenzmatrix** $I(M)$ (oder einfach $I$) des Multigraphen $M$ ist die $n \times m$-Matrix $(i_{rs})$ mit*

$$i_{rs} = \begin{cases} 1\,, & \textit{falls } e_r \textit{ mit } k_s \textit{ inzidiert} \\ 0\,, & \textit{sonst.} \end{cases}$$

**Beispiel 2.2-3:**




$$I(M) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Es gilt folgende Vereinbarung: Besteht $M$ aus $k$ vielen Komponenten $(k > 1)$, so nimmt man die Ecken und Kanten als in einer Weise durchnumeriert an, dass die einzelnen Komponenten nacheinander jeweils vollständig durchlaufen werden. Sind $M_1, \ldots, M_k$ die Komponenten, so hat $I(M)$ dann die Gestalt

$$I(M) = \begin{pmatrix} I(M_1) & 0 & \ldots & 0 \\ 0 & I(M_2) & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & I(M_k) \end{pmatrix}.$$

Wegen der Konstruktionsvorschrift laut Definition 2.2-3 gelten die folgenden Aussagen für jede Inzidenzmatrix $I(M)$ eines Multigraphen $M$:

- Jede Spalte von $I(M)$ enthält genau zwei Einsen.
- Die Anzahl der Einsen in der $i$-ten Zeile ist gleich dem Eckgrad $\gamma(e_i)$.
- Umnumerierung der Ecken- bzw. Kantenmenge führt zu Vertauschungen von Zeilen bzw. Spalten in der Inzidenzmatrix.

Dabei kann Aussage iii) noch verstärkt werden: Zwei Multigraphen sind genau dann isomorph, wenn deren Inzidenzmatrizen sich nur durch eine Permutation von Zeilen und Spalten unterscheiden. Man kann also sagen, dass die Struktur eines Multigraphen durch seine Inzidenzmatrix vollständig beschrieben ist.
Es wird sich nun herausstellen, dass auch die Begriffe Rang und Determinante (siehe Anhang C) einer Inzidenzmatrix bzw. ihrer Untermatrizen zur Beschreibung der Struktur eines Multigraphen genutzt werden können.

Für das folgende sei ein zusammenhängender Multigraph mit $(n \times m)$-Inzidenzmatrix $I$ gegeben.

**Hilfssatz 2.2-1:** *Es ist* $rg_2(I) \leq n-1$.

**Beweis:** Da jede Spalte von $I$ zwei Einsen enthält, ergibt die Summe aller Zeilen (modulo 2) den Nullvektor. Folglich sind die Zeilen über $\mathsf{IF}_2$ linear abhängig, somit $rg_2(I) \leq n-1$.

**Hilfssatz 2.2-2:** *Die Spalten von $I$, die zu den Kanten eines Kreises gehören, sind linear abhängig (über $\mathsf{IF}_2$).*

**Beweis:** Die Summe dieser Spalten ergibt den Nullvektor. Da nämlich eine Eins in einer Spalte eine mit der betreffenden Kante inzidente Ecke bezeichnet, gibt es zu jeder solchen Eins genau eine zweite Spalte mit einer Eins an derselben Stelle.

**reduzierte Inzidenzmatrix**
**Bezugsecke**

Für die nächsten Aussagen erweist es sich als nützlich, eine **reduzierte Inzidenzmatrix** $I_r$ zu betrachten, die sich ergibt, wenn die $r$-te Zeile von $I$ weggelassen wird. Die zu dieser Zeile gehörende Ecke $e_r$ wird auch als **Bezugsecke** der reduzierten Inzidenzmatrix $I_r$ bezeichnet.
Es muss hier darauf hingewiesen werden, dass in der Literatur häufig die Bezeichnung "Inzidenzmatrix" für das hier verwendete "reduzierte Inzidenzmatrix" genommen wird.

Ist $M$ ein Baum, so gilt $m = n-1$, eine reduzierte Inzidenzmatrix ist folglich quadratisch.

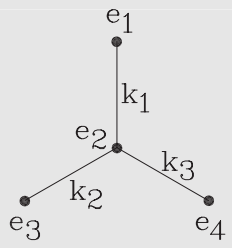
**Satz 2.2-2:** *Ist $M$ ein Baum und $I_r$ eine reduzierte Inzidenzmatrix von $M$, so gilt für die reelle Determinante $det(I_r) = \pm 1$.*

**Beweis:** Die Argumentation verläuft wieder induktiv.
Hat $M$ nur eine Kante und zwei Ecken, so ist $I_r$ die $(1 \times 1)$-Matrix mit Eintrag 1, mithin gilt $det(I_r) = 1$.
Als Induktionsannahme sei nun der Satz richtig für jeden Baum mit $k$ Ecken $(k \geq 2)$; ein Baum mit $k+1$ Ecken sei gegeben, $I_r$ sei eine reduzierte Inzidenzmatrix von $M$. Schließlich sei $e_s$ eine Endecke von $M$, die nicht die Bezugsecke ist. (Da nach Hilfssatz 1.2-2$M$ mindestens zwei Endecken hat, gibt es ein solches $e_s$.) Entwicklung von $det(I_r)$ nach Zeile $s$ führt zu

$$det(I_r) = (-1)^{s+t} det(I'),$$

wenn $k_t$ die eindeutige mit $e_s$ inzidente Kante ist; $I'$ ist dabei die Matrix, welche aus $I_r$ durch Streichen von Zeile $s$ und Spalte $t$ entsteht. Nun ist aber andererseits $I'$ eine reduzierte Inzidenzmatrix des Baumes, den man erhält, wenn aus $M$ die Ecke $e_s$ und die Kante $k_t$ entfernt werden. Da nach Induktionsannahme $det(I') = \pm 1$ gilt, folgt dies auch für $det(I_r)$, womit der Satz bewiesen ist.

**Beispiel 2.2-4:**




M

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Streichen der 4. Zeile (Bezugsecke $e_4$) führt zu

$$I_4 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \text{ mit } det(I_4) = -1.$$

Aus dem Satz und den vorangegangenen Hilfssätzen lassen sich nun zwei Folgerungen ableiten. Wieder sei ein zusammenhängender Multigraph $M$ mit $(n \times m)$-Inzidenzmatrix $I$ gegeben. Es ist noch zu beachten, dass Satz 2.2-2 bei Rechnung in $\mathsf{IF}_2$ ebenfalls die Aussage $det(I_r) = 1$ liefert.

**Folgerung 2.2-1:** *Es ist* $rg_2(I) = n - 1$.

**Beweis:** Die zu den Kanten eines Gerüsts von $M$ gehörenden Spalten einer reduzierten Inzidenzmatrix $I_r$ bilden nach Satz 2.2-2 über $\mathsf{IF}_2$ eine reguläre $((n-1) \times (n-1))$-Matrix, also muss auch $I$ den Rang $n-1$ haben.

**Folgerung 2.2-2:** *Eine quadratische* $((n-1)\times(n-1))$*-Untermatrix einer reduzierten Inzidenzmatrix* $I_r$ *ist genau dann regulär über* $\mathsf{IF}_2$*, wenn die ausgewählten Spalten zu einem Gerüst von* $M$ *gehören.*

Der Beweis folgt sofort mit Hilfssatz 2.2-2 und Folgerung 2.2-1.

Abschließend sei darauf hingewiesen, dass Folgerung 2.2-1 leicht verallgemeinert werden kann: besteht der Multigraph $M$ aus $k$ Komponenten, so gilt für eine Inzidenzmatrix $I$

$$rg_2(I) = n - k = \rho(M).$$

**Selbsttestaufgabe 2.2-1:**

*Berechnen Sie für den Baum $M$ aus Beispiel 2.2-4$det(I_1)$.*

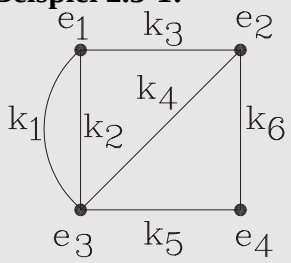
## 2.3 Weitere Matrizen und deren Eigenschaften

In diesem Abschnitt werden beliebigen Multigraphen zwei weitere Matrizen zugeordnet. In ihnen spiegelt sich die Zugehörigkeit von Kanten zu Kreisen bzw. Schnitten. Die Relevanz dieser beiden Arten von Matrizen liegt allerdings weniger in der Darstellung von Graphen als vielmehr in Anwendungsbereichen. Auf solche Anwendungen wird an späterer Stelle eingegangen, hier sollen zunächst nur einige grundlegende Eigenschaften dieser Matrizen dargestellt werden.
Für den ganzen Abschnitt sei ein beliebiger Multigraph $M$ vorgegeben, ebenso eine (von der Nummerierung der Ecken und Kanten abhängige) Inzidenzmatrix $I$. Ferner seien die Elemente von **C** (Vereinigungen kantendisjunkter Kreise) durchnumeriert ($C_i$ , $1 \leq i \leq 2^{\mu} - 1$) sowie die Elemente von **S** (Schnitte $S_j$ , $1 \leq j \leq 2^{\rho} - 1$).

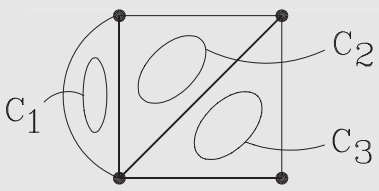
**Definition 2.3-1:** *Die **Kreis-Kanten-Matrix**$C(M) = (c_{ij})$ ist die in folgender Weise definierte $((2^{\mu} - 1) \times m)$-Matrix:*

$$c_{ij} = \begin{cases} 1\,, \textit{ falls } C_i \;\; k_j \textit{ enthält} \\ 0\,, \textit{ sonst.} \end{cases}$$

**Beispiel 2.3-1:**




Der Baum $B$ mit den Kanten $k_2, k_4, k_5$ führt zu den im folgendem Bild angedeuteten fundamentalen Kreisen $C_1, C_2, C_3$:



Die weiteren Elemente von $\mathbf{C}$ sind $C_4 = C_1 + C_2$, $C_5 = C_1 + C_3$, $C_6 = C_2 + C_3$ und $C_7 = C_1 + C_2 + C_3$. Es ergibt sich die Matrix

$$C(M) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Mit Hilfe von Hilfssatz 2.2-1 kann nun gezeigt werden:

**Folgerung 2.3-1:** *Für die Inzidenzmatrix $I$ und die Kreis-Kanten-Matrix $C$ gilt (über $\mathsf{IF}_2$) die Beziehung*

$$I \cdot C^t = 0.$$

**Beweis:** Die $i$-te Spalte von $I \cdot C^t$ erhält man, indem man die Spalten von $I$ aufsummiert, die zu den Kanten gehören, die in dem Element $\mathbf{C_i} \in \mathbf{C}$ enthalten sind. Nach Hilfssatz 2.2-1 muss dies den Nullvektor ergeben.

Fasst man wieder - wie in Abschnitt 1.3 - $\mathbf{C}$ als Untervektorraum der Potenzmenge $\mathbf{P}(K)$ auf, so besteht die Kreis-Kanten-Matrix aus einer zeilenweisen Auflistung der Elemente von $\mathbf{C}$, wobei - wie üblich - ein Element $L$ von $\mathbf{P}(K)$, also eine Menge von Kanten, durch das $m - tupel(l_1, \ldots, l_m)$ mit

$$l_j = \begin{cases} 1\,, \text{ falls } k_j \in L \\ 0\,, \text{ sonst} \end{cases}$$

beschrieben wird.

In diesem Lichte folgt unmittelbar aus Hilfssatz 1.3-1 der folgende

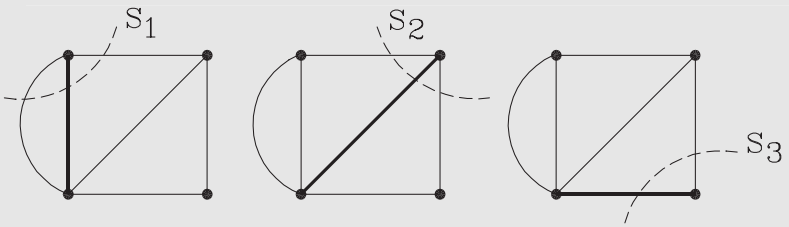
**Satz 2.3-1:** *Es ist $rg_2(C) = \mu(M)$.*

Analog zur Kreis-Kanten-Matrix ist die Schnitt-Kanten-Matrix definiert:

**Definition 2.3-2:** *Die **Schnitt-Kanten-Matrix** $S(M) = (s_{ij})$ ist die in folgender Weise definierte $((2^\rho - 1) \times m)$-Matrix:* **Schnitt-Kanten-Matrix**

$$s_{ij} = \begin{cases} 1\,, \textit{ falls } S_i \;\; k_j \textit{ enthält} \\ 0\,, \textit{ sonst.} \end{cases}$$

**Beispiel 2.3-2:**
Wir nehmen wieder den Multigraphen $M$ aus Beispiel 2.3-1, ebenso denselben Baum $B$. Es ergeben sich zunächst die in den folgenden Bildern angedeuteten fundamentalen Schnitte $S_1, S_2, S_3$:




Die weiteren Elemente von **S** sind $S_4 = S_1 + S_2$, $S_5 = S_1 + S_3$, $S_6 = S_2 + S_3$ und $S_7 = S_1 + S_2 + S_3$.

Es ergibt sich die Matrix

$$S(M) = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Man beachte: da die mit einer festen Ecke inzidenten Kanten immer einen Schnitt bilden, ist die Matrix $I$ stets in der Matrix $S$ enthalten, genauer: einige Zeilen von $S$ bilden $I$. So gesehen ist der folgende Satz eine Verstärkung von Folgerung 2.3-1:

**Satz 2.3-2:** *Es gilt (über* $\mathsf{IF}_2$*) die Beziehung*

$$S \cdot C^t = 0.$$

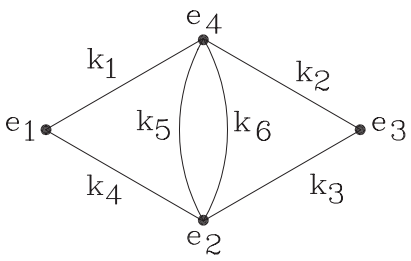
**Beweis:** Dies ist eine Konsequenz von Satz 1.3-1, dass jeder Schnitt mit jedem Kreis eine gerade Anzahl von Kanten gemeinsam hat.

Analog zu Satz 2.3-1 lässt sich nun auch leicht die folgende Aussage über den Rang von $S(M)$beweisen:

**Satz 2.3-3:** *Es ist* $rg_2(S) = \rho(M)$.

**Selbsttestaufgabe 2.3-1:**

*Stellen Sie für den folgenden Multigraphen $M$ eine Kreis-Kanten-Matrix auf:*



# 3 Algorithmen

## 3.1 Das Erkennen und Suchen von Bäumen

Einen Baum erkennt man daran, dass er ein Graph ohne Kreise ist. Ist ein Diagramm eines Graphen vorgelegt, so wird man dies in der Regel durch wenige Blicke nachprüfen können. Es stellt sich nun aber die Frage, wie man einen Graphen am schnellsten auf seine Kreisfreiheit untersucht, der in einem Rechner (z. B. durch eine Inzidenzmatrix) abgespeichert ist. Es wäre sicher keine gute Methode, den Rechner ein Diagramm zeichnen zu lassen und dieses dann optisch zu inspizieren. Es geht also darum, ein günstiges Rechenverfahren, einen sogenannten **Algorithmus**, zur Lösung dieses Problems zu finden.

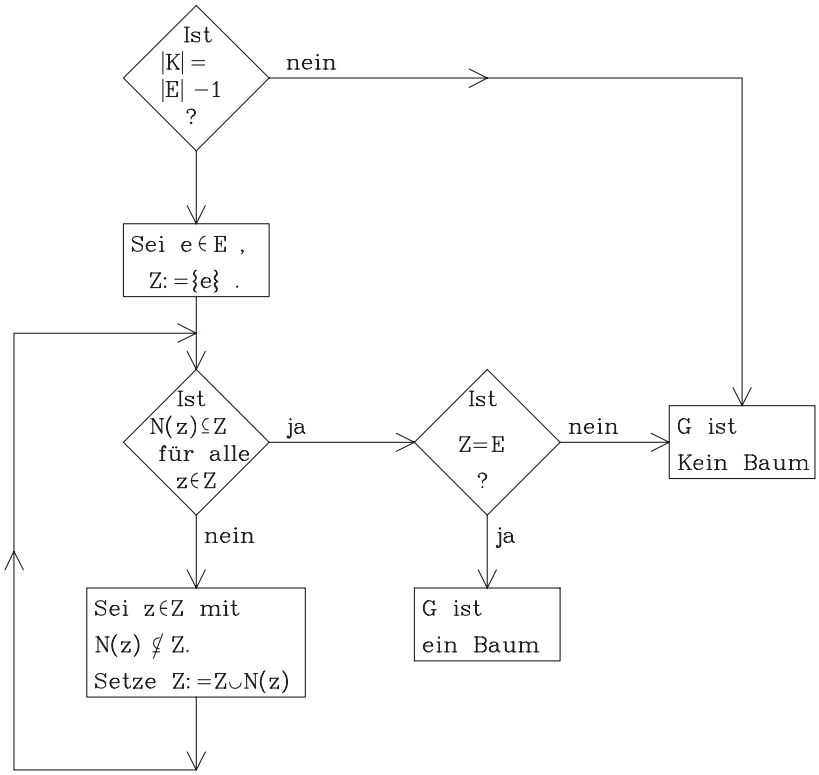
**Algorithmus**

Nach Satz 1.2-2 ist ein Graph $G = (E, K)$ genau dann ein Baum, wenn $G$ zusammenhängend ist und $|K| = |E| - 1$ gilt. Ein Algorithmus, der testet, ob $G$ ein Baum ist, kann also aus der Abfrage "Ist $|K| = |E| - 1$ ?" und einem Zusammenhangstest bestehen . In dem folgenden **Flussdiagramm** ist ein solcher Algorithmus beschrieben.

**Flussdiagramm**

Die Grundidee des in dem Diagramm beschriebenen Tests auf Zusammenhang ist es, dass von einer beliebig gewählten Ecke $e$ ausgehend die $e$ enthaltende Zusammenhangskomponente $Z$ bestimmt wird und am Schluss noch die Abfrage "$Z = E$?" steht. $Z$ wird iterativ dadurch aufgebaut, dass für ein $z \in Z$, welches noch Nachbarn außerhalb $Z$ hat, diese Nachbarn zu $Z$ hinzugenommen werden.

Gegeben sei ein Graph $G = (E, K)$, für jedes $e \in E$ sei $N(e)$ die Menge der Nachbarecken von $e$.
Der Algorithmus stoppt, wenn eine der beiden Boxen "$G$ ist ein Baum" bzw. "$G$ ist kein Baum" erreicht wird.




Flussdiagramme bieten eine von mehreren Möglichkeiten, die Funktionsweise von Algorithmen zu beschreiben. Eine andere Möglichkeit, die in der Literatur sehr gebräuchlich ist und die im folgenden auch hier verwendet werden soll, ist die Beschreibung mit Hilfe einiger weniger Sprachelemente, die in dieser oder ähnlicher Form Teil aller Computer- Hochsprachen sind. Es handelt sich m. a. W. um eine Schreibweise, die zwischen einem Flussdiagramm und Hochsprachen wie PASCAL oder ALGOL angesiedelt ist. Dabei ist es dann kein großer Aufwand mehr, von dort ausgehend zu einem **Programm** in einer dieser Hochsprachen zu kommen. Ein solches Programm ist also die konkrete Formulierung eines Algorithmus, wie sie zur Durchführung auf einem Computer benötigt wird, man spricht von einer **Implementierung** des Algorithmus.

Wir wollen den oben betrachteten Algorithmus in der hier verwendeten **Quasi-Hochsprache** beschreiben:

**Algorithmus "Baumtest"** Gegeben sei ein Graph $G = (E, K)$, $N(e) \subseteq E$ sei zu $e \in E$ die Menge der Nachbarecken. Es wird überprüft, ob $G$ ein Baum ist.

**Algorithmus**: Baumtest

```
begin
if |K| ≠ |E| − 1
   then G ist kein Baum,  goto ende
   else
   begin
      sei e ∈ E, Z := { e };
      while N(z) ⊆ /Z für ein z ∈ Z do
         wähle z ∈ Z mit N(z) ⊆/ Z,
         setze Z := Z ∪ N(z);
```

```
      if Z = E
         then G ist ein Baum
         else G ist kein Baum;
   end
ende: end
```

Diese Beschreibung des Algorithmus dürfte selbst für solche Leser verständlich sein, die bisher mit Programmiersprachen nichts oder nur wenig zu tun hatten. Bevor im nächsten Abschnitt noch einmal detailliert auf die hier verwendeten Sprachelemente eingegangen wird, soll ein weiteres Beispiel zur Illustration vorgestellt werden.

In Satz Satz 1.2-2 wurde gezeigt, dass jeder zusammenhängende Graph ein Gerüst enthält. Es wird nun ein Algorithmus zur Konstruktion eines Gerüsts angegeben. Dieser Algorithmus ist natürlich "besser" als die ebenfalls denkbare Vorgehensweise, einfach alle Teilgraphen des vorgegebenen Graphen einzeln darauf zu untersuchen, ob es sich um einen Baum handelt, der alle Ecken enthält. Die Funktionsweise des Algorithmus beruht auf der folgenden Tatsache, deren Beweis sich erübrigt.

**Hilfssatz 3.1-1:** *Sei $G = (E, K)$ ein Graph, der Teilgraph $T' = (E', K')$ sei ein Baum. Ist $e \in E'$ und $f \in E \backslash E'$ und $\{e, f\} \in K$, so ist auch der Teilgraph $(E' \cup \{f\},\ K' \cup \{\{e, f\}\})$ ein Baum in $G$.*

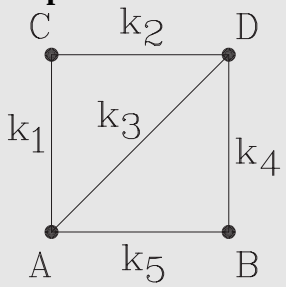
**Algorithmus "Konstruktion eines Gerüsts"** Gegeben sei ein zusammenhängender Graph $G = (E, K)$, $N(e) \subseteq E$ sei zu $e \in E$ die Menge der Nachbarecken. Es wird ein Gerüst $T = (E, L)$ konstruiert.

**Algorithmus**: Konstruktion eines Gerüsts

```
begin
sei  e ∈ E,  F := { e },  L := ∅;
   while N(e) ⊆/ F für
                ein e ∈ F do
  begin
      sei e ∈ F mit N(e) ⊆/ F,
      sei f ∈ N(e) \ F,
      F := F ∪ { f },  L := L ∪ {{e, f}};
   end
end
```

Der Algorithmus geht vor, wie im Hilfssatz vorgezeichnet ist, d. h. es wird in jedem Schritt eine Ecke des bereits konstruierten Baumes mit einer Nachbarecke, die noch nicht zu dem Baum gehörte, verbunden und so ein größerer Baum erhalten. Im folgenden Beispiel ist die Funktionsweise des Algorithmus illustriert.

**Beispiel 3.1-1:**




Zu Anfang sei $e = A$, also $F = \{A\}, L = \emptyset$.

$$
\begin{aligned}
1.Schritt: \quad & N(A) \not\subseteq F,\ C \in N(A) \setminus F, \\
& F = \{A, C\},\ L = \{k_1\}; \\
2.Schritt: \quad & N(A) \not\subseteq F,\ D \in N(A) \setminus F, \\
& F = \{A, C, D\},\ L = \{k_1, k_3\}; \\
3.Schritt: \quad & N(D) \not\subseteq F,\ B \in N(D) \setminus F, \\
& F = \{A, B, C, D\},\ L = \{k_1, k_3, k_4\}.
\end{aligned}
$$

Das ermittelte Gerüst ist $T = (E, \{k_1, k_3, k_4\})$.

## 3.2 Algorithmen und deren Komplexität

Zunächst soll der Begriff des Algorithmus etwas ausführlicher erläutert werden.

**Algorithmus**
**Problem**

Ein **Algorithmus** ist ein Verfahren zur Lösung eines Problems. Dabei muss unter einem **Problem** eine (im Prinzip unendlich große) Ansammlung gleichartiger einzelner Problemstellungen angesehen werden. Z. B. löst der zuletzt beschriebene Algorithmus allgemein das Problem der Konstruktion eines Gerüsts, wobei aber für eine Anwendung ein konkreter zusammenhängender Graph vorgegeben werden muss; für jeden dieser **Einzelfälle** arbeitet der Algorithmus erfolgreich.

**Einzelfall**

**Korrektheit**

Eine selbstverständliche Forderung an einen Algorithmus ist natürlich die **Korrektheit**, d. h. er sollte das Problem für jeden Einzelfall korrekt lösen. Der Nachweis, dass ein für ein Problem vorgeschlagener Algorithmus auch korrekt ist, kann sehr schwierig zu führen sein. Oft hilft ein mathematischer Satz, den der Algorithmus per Iteration ausnutzt. In anderen Fällen hat die Lösung nach dem Durchlaufen des Algorithmus stets gewisse Eigenschaften, die sich in einem allgemeinen mathematischen Satz zusammenfassen lassen, wobei die Korrektheit des Algorithmus als Konsequenz ebenfalls mit herauskommt - so verhält es sich beim "max-flow - min cut" - Theorem, das an späterer Stelle behandelt wird.
Die Korrektheitsbegriffe für Algorithmus und Programm sind (entsprechend den Begriffsbestimmungen des vorigen Abschnitts) im wesentlichen identisch - vorausgesetzt, dass bei der Umsetzung des Algorithmus in ein Programm für einen konkreten Rechner keine Fehler gemacht werden. Es gibt auch Ansätze, die Korrektheit

von Programmen mit formalen Methoden zu untersuchen, man spricht hier von der **Verifikation** von Programmen.

**Verifikation**

Als weitere notwendige Eigenschaften eines Algorithmus werden angesehen:

- Endliche Beschreibbarkeit: Das Verfahren kann mit einem endlichen Text (Sprache , Formeln) beschrieben werden.
- Effektivität: Jeder einzelne Schritt des Verfahrens muss "mechanisch" durchführbar sein (z. B. durch einen Computer).
- Determinismus: Die Reihenfolge aller Rechenschritte muss durch das Verfahren für jeden Einzelfall eindeutig festgelegt sein.
- Endlichkeit: Das Verfahren muss für jeden Einzelfall nach endlich vielen Schritten abbrechen.

Im vorigen Abschnitt wurde anhand von Beispielen bereits die "Sprache" festgelegt, in der im weiteren Verlauf des Kurses Algorithmen beschrieben werden sollen - wir haben dort den Begriff einer Quasi-Hochsprache eingeführt. Die Elemente dieser Sprache werden im folgenden kurz erläutert.

**Verzweigung**

**Verzweigung**: **if** A **then**$B_1, B_2, \ldots, B_k$**else**$C_1, C_2, \ldots, C_l$; steht für:
Wenn Bedingung A erfüllt ist, werden die Operationen $B_1, B_2, \ldots, B_k$ ausgeführt, andernfalls $C_1, C_2, \ldots, C_l$. Die mit "else" eingeleitete Alternative kann dabei auch fehlen.

**Schleife**

**Schleife**: **while** A **do**$B_1, \ldots, B_k$; steht für:
Wenn Bedingung A erfüllt ist, werden $B_1, \ldots, B_k$ ausgeführt; dies wird solange wiederholt, wie A erfüllt ist.

**Iteration**

**Iterationen**: **for** L **do**$B_1, \ldots, B_k$; steht für:
Für jedes Element der Liste L werden die Operationen $B_1, \ldots, B_k$ je einmal ausgeführt. Dabei kann die Liste L geordnet sein (z. B. **for**$i = 1, \ldots, 6$**do**$B_1, B_2$;) oder ungeordnet (z. B. **for**$s \in S$**do**$B_1, B_2$;). Im zweiten Fall ist die Reihenfolge nicht spezifiziert.

**repeat**$A_1, \ldots, A_k$**until**$B$; steht für:
Die Operationen $A_1, \ldots, A_k$ werden durchgeführt. Ist danach die Bedingung $B$ nicht erfüllt, wird dies solange wiederholt, bis $B$ erfüllt ist.

Zur Erhöhung der Übersichtlichkeit werden zusammenhängende Blöcke oft durch ein **begin-end-Paar** eingeschlossen, unter anderem auch der ganze Algorithmus (s. z. B. die "Konstruktion eines Gerüsts"). Außerdem wird die Struktur des Algorithmus durch Einrücken von Blöcken (beim Aufschreiben des Quasi-Programms) zusätzlich optisch verdeutlicht.

**Sprungbefehl**

Ein weiteres Element der Sprache ist der **Sprungbefehl**:**goto** Name
bedeutet, dass als nächstes das auszuführen ist, was in der Zeile steht, die mit "Name:" gekennzeichnet ist (s. z. B. in "Baumtest").

**Zuweisungen**

**Zuweisungen** werden mit $:=$ vorgenommen, z. B. steht "$Z := Z \cup \{e\}$" für: der

Mengenvariablen wird als Wert die Menge zugewiesen, die aus der "bisherigen Menge $Z$" plus dem Element $e$ besteht.

Unsere kurze Erläuterung der verwendeten Sprache schließen wir mit drei Bemerkungen ab:

**Datenstrukturen**

1. Bezüglich der **Datenstrukturen**, die verwendet werden dürfen, werden im folgenden keine Einschränkungen gemacht. Es sind einfache und indizierte Variablen sowie Variablen für Mengen erlaubt. Die durch die Variablen repräsentierten Größen können beliebiger Art sein (z. B. auch Ecken eines Graphen).

2. Als Bedingungen A oder Operationen $B_i$ (wie z. B. in **while** A **do** $B_1, \ldots, B_k$) lassen wir alle sinnvollen und unzweideutigen mathematischen Aussagen zu, z. B.:
   **while** $S \neq \emptyset$ **do** wähle $s \in S$, suche einen Kreis $C$ durch $s$, setze $K := K \cup \{C\}$, $S := S \backslash \{s\}$;
   Dabei müssen die aufgeführten Operationen stets endlich ausführbar sein.

3. Die unter 1. und auch schon bei der Erklärung einer Iteration gewährte Freizügigkeit, die z. B. dazu führt, dass beim Durchlaufen einer Menge die Reihenfolge der Elemente nicht festgelegt wird, bedeutet strenggenommen ein Abrücken von der Forderung des Determinismus. Dieser ist jedoch dadurch gegeben, dass eine konkret gewählte Hochsprache entweder selbst sich in solchen Punkten festlegen muss oder aber -spätestens- ihr "Compiler".

   Für den Rest dieses Abschnitts beschäftigen wir uns mit einer Eigenschaft, die nicht notwendig zu jedem Algorithmus gehört, jedoch wünschenswert ist und die eigentliche Herausforderung beim Suchen nach einem Algorithmus bildet: seine möglichst hohe **Effizienz**. Damit ist gemeint , dass ein Algorithmus möglichst schnell und ökonomisch arbeiten soll.

**Effizienz**

**Beispiel 3.2-1:**
Zur Illustration dieses Punktes wird zunächst ein Beispiel vorangestellt. Wir gehen von $n$ reellen Zahlen aus, die der Größe nach sortiert sind: $a_1 < a_2 < \ldots < a_n$. Eine weitere Zahl $a$ soll in diese Liste an der richtigen Stelle eingefügt werden (vgl. Beispiel 1.2-7). Als Ausgabe des Algorithmus wird der Index $i$ mit $a_i < a < a_{i+1}$ erwartet, wobei in den Grenzfällen $a < a_1$ bzw. $a_n < a$ jeweils eine der Zahlen keine Bedeutung hat.
Ein möglicher Algorithmus ist der folgende:

**Algorithmus**: A1

```
begin
   for k = 1,...,n do
      if a < a then i := k − 1, goto ende;
   i := n;
ende: end
```

Für die Formulierung des zweiten Algorithmus setzen wir voraus, dass $n$ eine Zahl der Form $n = 2^m - 1$ ist (also z. B. $1, 3, 7, 15, 31, \ldots$). Dadurch ist es gewährleistet, dass die auftretenden Brüche immer zu ganzen Zahlen führen - sonst müsste das Verfahren etwas komplizierter beschrieben werden. Das Vorgehen entspricht genau dem Intervallhalbierungsverfahren beim binären Suchen (vgl. Beispiel 1.2-7).

**Algorithmus**: A2

```
begin
   k := (n+1)/2;
   while n ≥ 3 do
   begin
      if a < a_k

         then k := k − (n+1)/4
         else k := k + (n+1)/4;
      n := (n−1)/2;
   end
   if a < a_k
      then i := k − 1
      else i := k;
end
```

Die Arbeitsweise dieser beiden Algorithmen soll nun an einem konkreten Fall verdeutlicht werden.
Es sei $n = 7$ mit der gegebenen aufsteigenden Folge

$$\begin{array}{ccccccccccccc} 2 & < & 6 & < & 7 & < & 7,2 & < & 9,1 & < & 11 & < & 20 \\ a_1 & & a_2 & & a_3 & & a_4 & & a_5 & & a_6 & & a_7, \end{array}$$

$a = 10$ soll eingefügt werden.

Ablauf von A1:

$$\begin{array}{lcll} k = 1 & \rightarrow & Vergleich\ a < a_1 & \rightarrow \quad Ergebnis : nein; \\ k = 2 & \rightarrow & Vergleich\ a < a_2 & \rightarrow \quad Ergebnis : nein; \\ k = 3 & \rightarrow & Vergleich\ a < a_3 & \rightarrow \quad Ergebnis : nein; \\ k = 4 & \rightarrow & Vergleich\ a < a_4 & \rightarrow \quad Ergebnis : nein; \\ k = 5 & \rightarrow & Vergleich\ a < a_5 & \rightarrow \quad Ergebnis : nein; \\ k = 6 & \rightarrow & Vergleich\ a < a_6 & \rightarrow \quad Ergebnis : ja, \end{array}$$

also $i = 5$, Sprung nach "ende".

Ablauf von A2:

$k = (7+1)/2 = 4$

$$
\begin{array}{lll}
\rightarrow\ Vergleich\ a < a_4 & \rightarrow & Ergebnis : nein, \\
 & & also\ k = 4+2 = 6, n = (7-1)/2 = 3; \\
\rightarrow\ Vergleich\ a < a_6 & \rightarrow & Ergebnis : ja, \\
 & & also\ k = 6-1 = 5, n = (3-1)/2 = 1; \\
 & & Ausstieg\ aus\ der\ while-Schleife; \\
\rightarrow\ Vergleich\ a < a_5 & \rightarrow & Ergebnis : nein, \\
 & & also\ i = 5, end.
\end{array}
$$

Im Ergebnis waren bei A1 sechs, bei A2 nur drei Vergleiche nötig.
Eine wichtige Einsicht ist es nun, dass der Algorithmus A2 besser (weil effizienter) ist als A1: beim zweiten Algorithmus wird die Zahl $a$ mit $log_2(n+1)$-vielen der $a_i$ verglichen, bis seine richtige Position gefunden ist (für $n = 2^m - 1$ bedeutet dies m Vergleiche), bei A1 muss $a$ im schlechtesten Fall mit allen $a_i$ verglichen werden (falls nämlich $a > a_n$ ist). Im Mittel braucht A1 $n/2$ viele Vergleiche, was immer noch schlechter ist als $log_2(n+1)$ - umso mehr für große $n$.

Allgemein geht man davon aus, dass einem Problem stets eine natürliche Zahl $n$ als **Problemgröße** zugeordnet werden kann, relativ zu welcher die Güte eines Algorithmus zu beurteilen ist. Ist nun $A$ ein Algorithmus, der das gegebene Problem löst, so ist die **Komplexität** $T_A(n)$ die Anzahl der im schlechtesten Fall auszuführenden elementaren Rechenoperationen bei einer Anwendung von $A$ auf einen Einzelfall von Größe $n$.

**Problemgröße**

**Komplexität**

Als Maßstab wird das "Verhalten im schlechtesten Fall" genommen, weil zur Bestimmung des "Verhaltens im Mittel" bereits bei einfachen Algorithmen sehr komplizierte wahrscheinlichkeitstheoretische Untersuchungen nötig wären und man sich schon bald mit empirischen Verfahren behelfen müsste.

Man muss sich nun festlegen, was zu den "elementaren Rechenoperationen" gezählt werden soll.
Wir stellen uns einen Rechner mit unbegrenztem Speicher vor, in dem zu Beginn der Rechnung die Eingabedaten und am Ende die Ausgabedaten stehen. Der technische Aufwand der Ein- und Ausgabe sowie der Speicherung wird also außer acht gelassen. Der Rechner kann arithmetische Operationen $(+, -, *, /, exp)$, Vergleichsoperationen $(=, \neq, <, \leq, >, \geq)$, Speicheroperationen, Sprungbefehle $(GOTO, IF\ \ldots\ THEN)$ usw. ausführen.
Bei der Bestimmung von $T_A(n)$ wollen wir nur die arithmetischen und die Vergleichsoperationen zählen und Sprungbefehle, Zugriffe zu Feldern usw. außer acht lassen. Dies macht im Grunde nichts aus, da man im allgemeinen $T_A(n)$ ohnehin nicht präzise bestimmen kann und nur nach der **Größenordnung** bzw. **Wachstumsrate** dieser Funktion fragt. Dazu wird die folgende Notation verwendet:

**Größenordnung**

**Wachstumsrate**

$f(n)$ und $g(n)$ seien Funktionen mit natürlichen Zahlen als Argumenten und Werten in $\mathbb{R}^+$, der Menge der positiven reellen Zahlen.
Man schreibt

$$f(n) = O(g(n))$$

und sagt, $f$ habe höchstens Wachstumsrate $g$, wenn es eine Konstante $c > 0$ gibt mit $f(n) \leq c \cdot g(n)$ für alle $n$ ab einem gewissen $n_0$;

$$f(n) = \Omega(g(n))$$

($f$ hat mindestens Wachstumsrate $g$), wenn es eine Konstante $c > 0$ gibt mit $f(n) \geq c \cdot g(n)$ für alle $n$ ab einem gewissen $n_1$;

$$f(n) = \Theta(g(n))$$

($f$ hat Wachstumsrate $g$), wenn $f(n) = O(g(n))$ und $f(n) = \Omega(g(n))$ gelten.

Gilt für einen Algorithmus $A$

$$T_A(n) = O(g(n)),$$

so sagen wir, $A$ habe die Komplexität $O(g(n))$.

Nach dieser Definition hat $A1$ (in Beispiel 3.2-1) die Komplexität $O(n)$ und $A2$ die Komplexität $O(logn)$.

**Beispiel 3.2-2:**
Ist $f(n)$ ein Polynom vom Grad $k$, d. h. gilt

$$f(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0,$$

so ist $f(n) = O(n^k)$, denn mit $c = 2|a_k|$ ist für genügend großes $n$ auf jeden Fall $f(n) \leq cn^k$.

**polynomialer Algorithmus**

**effizienter Algorithmus**

Man sucht stets nach **polynomialen Algorithmen**, also Algorithmen der Komplexität $O(n^k)$ für ein geeignetes $k$. Aus gutem Grund nennt man diese Algorithmen auch **effizient**: Polynome wachsen wesentlich langsamer als etwa Exponentialfunktionen. In einer Formel kann dies so ausgedrückt werden: ist $p(n)$ irgendein Polynom und $a > 1$ eine reelle Zahl, so gilt

$$\lim_{n \to \infty} \frac{p(n)}{a^n} = 0.$$

("Schlimmer" als die Funktion $a^n$ ist noch die Funktion $n!$.)

Die folgende überschlagsweise Rechnung dient der Verdeutlichung des Unterschieds zwischen polynomialer und exponentieller Wachstumsrate: Angenommen, es stehe ein Rechner zur Verfügung, der pro Sekunde 1 Milliarde Schritte ausführen kann. Für ein Problem $P$ habe man einen Algorithmus $A1$ von Komplexität $O(2^n)$ und einen Algorithmus $A2$ von Komplexität $O(n^3)$. In einem Einzelfall mit $n = 60$ dauert die Rechnung dann, wenn wir direkt $T_{A1}(n) = 2^n$ und $T_{A2}(n) = n^3$ annehmen, bei $A1$ ca. 37 Jahre, bei $A2 \frac{1}{4630}$ Sekunden.

Hier sind einige weitere Begriffe:
Probleme, die einen polynomialen Algorithmus zulassen, heißen **leicht**. Probleme, für die es keinen polynomialen Algorithmus geben kann, werden **unzugänglich** oder **hart** genannt. Über eines muss man sich im klaren sein: es kann sehr schwierig sein herauszufinden, ob ein Problem leicht ist!

**leichtes Problem**

**unzugängliches Problem**
**hartes Problem**

**Beispiel 3.2-3:**
Wir betrachten das folgende Problem. Den $2^m$ vielen Teilmengen einer $m$-elementigen Menge $X$ seien beliebige reelle Zahlen (Bewertungen) zugeordnet, man hat m. a. W. eine Abbildung

$$f : \mathbf{P}(X) \to \mathbb{R}.$$

Die Abbildung sei auf irgendeine Weise in einem Rechner abgespeichert, und es sei nun das Ziel, eine Menge $Y \in P(X)$ mit möglichst kleiner Bewertung $f(Y)$ zu finden. Dieses Problem ist, wenn die Zahl $m$ als Problemgröße angenommen wird, unzugänglich, denn jeder Algorithmus muss die Bewertung jeder diese Mengen mindestens einmal mit irgendeiner anderen vergleichen, was zu mindestens $2^m$ vielen Rechenoperationen führt. Sieht man allerdings die Anzahl der Teilmengen $2^m$ als Problemgröße $n$ an, so ist das Problem leicht.

Es sei an dieser Stelle auch auf die für die Theorie der Algorithmen wichtigen **NP-vollständigen** Probleme hingewiesen. Es handelt sich um eine (durch neue Erkenntnisse immer weiter wachsende) Klasse von Problemen mit der Eigenschaft, dass

**NP-vollständig**

1. von keinem der Probleme bekannt ist, ob es leicht oder hart ist und
2. **alle** diese Probleme als leicht nachgewiesen sind, wenn man dies nur für **eines** von ihnen herausfindet.

Auf eine präzise mathematische Definition der NP-Vollständigkeit können wir hier nicht eingehen. Man müsste hierzu zunächst die Definition eines Algorithmus noch stärker formalisieren. Üblicherweise wird dann der Begriff der **Turing-Maschine** benutzt. Wer in dieses Gebiet im Bereich Mathematische Logik / Theoretische Informatik tiefer einsteigen möchte, sei auf das Buch von Garey und Johnson (s. Literaturverzeichnis) verwiesen.

**Turing-Maschine**

Den Schluss dieses Abschnitts bilden folgende kurze Anmerkungen:

1. Bei den graphentheoretischen Problemen, die im weiteren Verlauf des Kurses betrachtet werden, wird in der Regel die Eckenanzahl $n$ als Problemgröße angenommen. Stützt sich ein Algorithmus auf ein Absuchen der Kanten, so wird man zu einer Bewertung des Algorithmus eher die Kantenanzahl $m$ als Problemgröße nehmen. Handelt es sich um einen Graphen (weder Pseudo- noch Multi-), so gilt $m = 0(n^2)$; insofern können sich in diesem Fall bei der Wahl von $n$ oder $m$ keine qualitativen Unterschiede (z. B. bei der Frage, ob der Algorithmus polynomial ist) ergeben.

2. Man müsste eigentlich die Problemgröße sorgfältiger definieren, wenn in einem Problem große numerische Zahlenwerte auftreten. In Beispiel 3.2-1 ist z. B. nicht nur die Anzahl $n$ der betrachteten Zahlen von Bedeutung, sondern auch die **Größe** der $a_i$ hat bei einem realen Computer Einfluss auf die Laufzeit. Ebenso verhält es sich in der Graphentheorie, wenn man Ecken- und Kantenbewertungen zulässt (s. nächstes Kapitel). Wir werden diesen Einfluss der Zahlengrößen jedoch im weiteren vernachlässigen.

**Selbsttestaufgabe 3.2-1:**

*Erläutern Sie den Begriff der Komplexität eines Algorithmus.*

## 3.3 Weitere Algorithmen und Begriffe

Zur Illustration der bisherigen Aussagen über Algorithmen werden in diesem Abschnitt weitere Algorithmen zur Lösung graphentheoretischer Probleme vorgestellt. Unter anderem beschäftigen wir uns mit einem NP-vollständigen Problem. Ferner werden die Begriffe Heuristik und approximativer Algorithmus eingeführt.

Ein Graph $G = (E, K)$ ist nach Definition 1.2-8 bipartit, wenn seine Eckenmenge so in Teilmengen $E'$ und $E''$ unterteilt werden kann, dass jede Kante eine Ecke aus $E'$ mit einer aus $E''$ verbindet. Nach Satz 1.2.4 ist dies gleichbedeutend damit, dass $G$ keinen Kreis ungerader Länge enthält. Der folgende Algorithmus prüft dies nach.

**Algorithmus "Bipartit"**

**Algorithmus "Bipartit"** Gegeben seien ein zusammenhängender Graph $G = (E, K)$ und für jedes $e \in E$ die Menge $N(e)$ der Nachbarn von $e$. Es wird überprüft, ob $G$ bipartit ist.

**Algorithmus**: Bipartit

```
begin
   sei e ∈ E, f(e) := 0;
   while es gibt noch unmarkierte Ecken do
      begin
         for x ∈ E do
         if x ist bereits markiert
            then
            if es gibt ein markiertes y ∈ N(x)
               mit f(y) = f(x)
                 then G ist nicht bipartit
            else markiere jedes nicht-
               markierte y ∈ N(x)
               mit f(y) := f(x) + 1
               (in IF₂);
      end
end
```

Der Algorithmus hat die Komplexität $0(n^3)(n = |E|)$. Dies sieht man am leichtesten so ein: Die while-Schleife wird höchstens $(n-1)$-mal durchlaufen. Innerhalb der Schleife werden für jede bereits markierte Ecke deren noch nicht markierte Nachbarn markiert (oder ein ungerader Kreis entdeckt).
Sind nach dem Durchlauf des Algorithmus alle Ecken markiert worden, so bilden $E' := f^{-1}(0)$ und $E'' := f^{-1}(1)$ eine disjunkte Aufteilung der Eckenmenge derart, dass Kanten nur zwischen Elementen von $E'$ und $E''$ verlaufen.

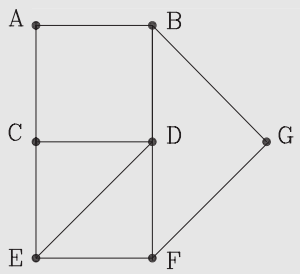
**Algorithmus "Planar"**

Auch die Frage, wie man Graphen effektiv auf Planarität testen kann, wollen wir hier noch einmal zur Sprache bringen. Es sind mehrere polynomiale Algorithmen bekannt, jedoch sind bei jedem dieser Algorithmen für das genaue Verständnis weitere Begriffsbildungen nötig, auf die hier nicht eingegangen werden soll. Wir wollen nur kurz die Grundidee des Algorithmus nach G. Demoucron, Y. Malgrange und R. Pertuiset skizzieren - genauer kann dies z. B. in dem zitierten Buch von R. Gould nachgelesen werden.
In dem gegebenen Graphen $G$ sucht man zunächst einen Kreis $C$, den man in die Zeichenebene einbettet. Der Rest des Graphen (also $G$ ohne $C$) zerfällt in sog. "Segmente". Diese werden nun schrittweise zu einer planaren Einbettung eines immer größeren Teils von $G$ hinzugenommen, wobei nach jeder Teileinbettung die übriggebliebenenen Segmente neu bestimmt werden müssen. Das Hauptproblem beim Korrektheitsbeweis dieses Algorithmus steckt -wie zu erwarten- in dem Nachweis, dass der Graph nicht planar sein kann, wenn der Algorithmus bei der Einbettung eines Teilgraphen stockt, d. h. diese nicht erweitert werden kann.

**minimale Knotenüberdeckung**

Zum Abschluss dieses Kapitels wollen wir das Problem der **minimalen Knotenüberdeckung** betrachten. Es geht darum, in einem gegebenen Graphen $G = (E, K)$ eine Menge $U \subseteq E$ mit möglichst wenigen Knoten zu finden derart, dass von jeder Kante mindestens eine der beiden Ecken zu $U$ gehört, formal: ist $k = \{e, f\}$, so gilt $e \in U$ oder $f \in U$.

**Beispiel 3.3-1:**
Der im folgenden Diagramm gezeigte Graph wurde bereits in 1.2.10 verwendet - er soll ein Datennetz mit Netzknoten $A, B, \ldots, G$ darstellen:



Sollen nun die einzelnen Leitungen überwacht werden (z. B., um Verkehrsmessungen zu machen), so genügt es, in den Knoten einer Knotenüberdeckung Beobachtungsstationen einzurichten. Es sind $\{A, D, E, G\}$ und $\{B, C, D, F\}$ minimale Knotenüberdeckungen.

Fragt man nach einem gutem Algorithmus zur Suche einer minimalen Knotenüberdeckung, so bietet sich, da man möglichst wenige Knoten auswählen will, folgende Strategie an: man wählt zunächst einen Knoten mit möglichst vielen Nachbarn aus und erklärt ihn zum Element der gesuchten Menge, entfernt ihn (mit allen mit ihm inzidenten Kanten) aus dem Graphen, verfährt mit dem Rest genauso, usw. . . .
Dies führt zu dem folgenden Algorithmus:

**Algorithmus "Knotenüberdeckung" Nr. 1**

### Algorithmus "Knotenüberdeckung" Nr. 1

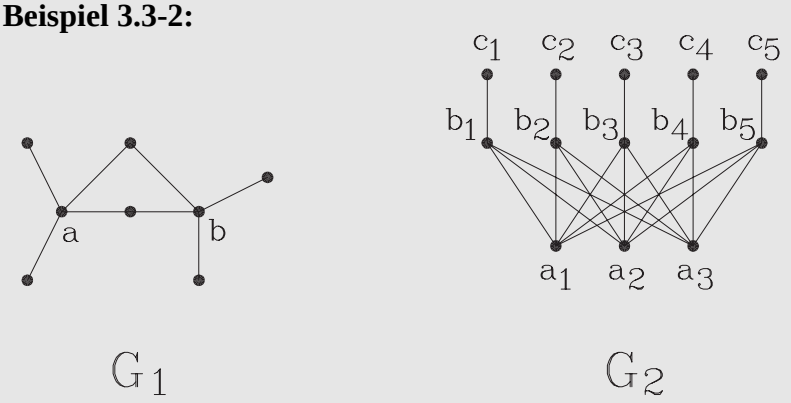
Gegeben sei ein Graph $G = (E, K)$.
Es wird eine Knotenüberdeckung $\ddot{U}$ konstruiert.

**Algorithmus**: Knotenüberdeckung Nr. 1

```
begin
   Ü := ∅;
   while K ≠ ∅ do
      wähle eine Ecke e ∈ E mit größtem
      Grad,
      entferne sie aus E und die e
      enthaltenden Kanten aus K,
      setze Ü := Ü ∪ {e};
end
```

Wendet man diesen Algorithmus auf den Graphen aus Beispiel 3.3-1 an, so ergeben sich u. a. die beiden dort aufgeführten Überdeckungen - jedoch nur dann, wenn nach der Wahl von $D$ in dem dann übrigbleibenden Kreis "zufällig" auf günstige Weise weitere Ecken ausgewählt werden. Wird nämlich nach der Wahl von $D$ etwa $A$ und dann $F$ gewählt, so ergibt sich am Ende eine Überdeckung mit fünf Knoten. Dies zeigt, dass obiger Algorithmus das gestellte Problem nicht löst.

**Beispiel 3.3-2:**




Bei $G_1$ führt die Anwendung von Algorithmus Nr. 1 zu der minimalen Knotenüberdeckung $\{a, b\}$. Bei Anwendung auf $G_2$ werden auf jeden Fall zunächst $a_1, a_2, a_3$ und dann fünf weitere Knoten ausgewählt, was stets zu einer Menge mit acht Knoten führt. Die (in diesem Fall eindeutige) minimale Knotenüberdeckung $\{b_1, b_2, b_3, b_4, b_5\}$ wird nicht gefunden.

Einen Algorithmus wie den obigen, der auf einer plausiblen Strategie beruht, aber nicht unbedingt immer zum Ziel führt, nennt man ein **heuristisches Verfahren** (oder einfach **Heuristik**). Von einer guten Heuristik erwartet man, dass sie -wenn sie schon nicht das eigentliche Problem löst- zumindest akzeptable **Nährungslösungen** zum Ergebnis hat. Man spricht dann von einem **approximativen Algorithmus**.

**heuristisches Verfahren**
**Heuristik**
**Nährungslösung**
**approximativer Algorithmus**

Dass wir bei "Knotenüberdeckung Nr.1" trotzdem von einem ”Algorithmus” gesprochen haben, bedeutet strenggenommen eine Abkehr von der im vorigen Abschnitt aufgestellten Forderung, dass ein Algorithmus das vorgelegte Problem korrekt lösen soll. Wir werden auch im folgenden so verfahren und den etwas gelockerten Begriff des Algorithmus verwenden.

Das Problem der minimalen Knotenüberdeckung ist NP-vollständig, d. h. es ist kein guter Algorithmus zur Lösung bekannt, und es ist sogar unwahrscheinlich, dass überhaupt ein polynomialer Algorithmus existiert. In einem solchen Fall ist man auf approximative Algorithmen angewiesen. Leider ist es hier so, dass der obige Algorithmus Nr. 1 reichlich schlechtes Verhalten zeigt: man kann das Beispiel $G_2$ aus Beispiel 3.3-2 zu einer Serie von Beispielen ausbauen, mit der sich dann demonstrieren lässt, dass die von dem Algorithmus produzierte Lösung prozentual beliebig weit vom Optimum abweichen kann.

Besser verhält sich eine andere Heuristik, die auf folgender Idee beruht: Man betrachte irgendeine Kante und entferne die beiden Endpunkte sowie alle mit diesen inzidenten Kanten aus dem Graphen. Dies führt zu:

**Algorithmus "Knotenüberdeckung" Nr. 2**

**Algorithmus "Knotenüberdeckung" Nr. 2**

Gegeben sei ein Graph $G = (E, K)$.
Es wird eine Knotenüberdeckung $\ddot{U}$ konstruiert.

**Algorithmus**: Knotenüberdeckung Nr. 2

```
begin
   Ü := ∅;
   while K ≠ ∅ do
      wähle ein k = {e, f} ∈ K,
      entferne e und f aus E,
      entferne die e oder f enthaltenden
      Kanten aus K,
      setze Ü := Ü ∪ {e, f};
end
```

Wendet man Algorithmus "Knotenüberdeckung Nr.2" auf die Graphen aus Beispiel 3.3-2 an, so erhält man ( unabhängig von den getroffenen Wahlentscheidungen) bei $G_1$ eine vierelementige, bei $G_2$ eine zehnelementige Knotenüberdeckung. Dies ist in beiden Fällen eine Abweichung von $100\%$ vom Optimum. Im Gegensatz zu der Situation bei Algorithmus Nr. 1 kann man jedoch beweisen, dass es schlimmer nicht kommen kann:

**Satz 3.3-1:** *Eine von dem Algorithmus "Knotenüberdeckung" Nr.2 gefundene Lösung enthält höchstens doppelt so viele Knoten wie eine minimale Knotenüberdeckung.*

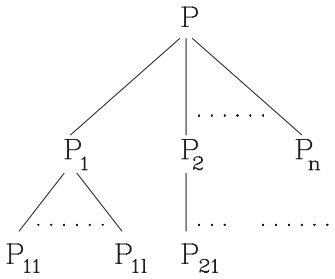
**Beweis:** Die in einem Durchlauf des Algorithmus ausgewählten Kanten haben keine Knoten gemeinsam. Jede Knotenüberdeckung (auch eine minimale) muss also von jeder **dieser** Kanten mindestens einen Knoten enthalten, muss also insgesamt mindestens halb so viele Knoten enthalten wie die vom Algorithmus gelieferte Knotenmenge.

Wir wollen das Kapitel mit folgendem Hinweis abschließen: Auch bei unzugänglichen Problemen hat man gewisse Techniken entwickelt, nicht-polynomiale Algorithmen so weit zu verbessern (z. B. hinsichtlich der Absuchreihenfolge der zu betrachtenden Möglichkeiten), dass sie in einer Vielzahl von Fällen akzeptables Verhalten zeigen. Eine dieser Techniken ist die **branch-and-bound-Methode** (manchmal auch ”Verzweigungsmethode” genannt).

**branch-and-bound-Methode**

Ihr liegt die folgende Idee zugrunde.

Gegeben sei ein Minimierungsproblem, wo für ein Problem $P$ eine optimale Lösung (hier also ein Minimum) gesucht werden soll. Man versucht nun, $P$ in Teilprobleme $P_1, \ldots, P_n$ zu zerlegen, deren Lösungen insgesamt zu einer Lösung von $P$ führen (”branch”). Erkennt man dabei, dass die Lösung eines $P_i$ eine obere Schranke für die Lösung von $P_j (j \neq i)$ darstellt, dann braucht $P_i$ nicht mehr weiter betrachtet zu werden (”bound”). Nach diesem Prinzip können auch die einzelnen Teilprobleme behandelt werden, man kommt also zu einem Baum von Teilproblemen:




Wir wollen die Methode an einem Beispiel illustrieren. An weiteren Beispielen Interessierte seien auf die Literatur verwiesen, z. B. das Buch von Papadimitriou und Steiglitz (s. Literaturverzeichnis).

**Beispiel 3.3-3:**
Die Entfernungen zwischen den Städten $A$, $B$, $C$ und $D$ seien (in km) in der folgenden Matrix $(d_{ij})$ festgehalten:

| | A | B | C | D |
|---|---|---|---|---|
| A | $\infty$ | 4 | 2 | 1 |
| B | 4 | $\infty$ | 1 | 2 |
| C | 2 | 1 | $\infty$ | 3 |
| D | 1 | 2 | 3 | $\infty$ |

(Für dieses Problem ist in der Diagonalen der Eintrag ”$\infty$” statt ”0” sinnvoll.) Es ist eine Tour durch alle Städte und zurück zum Ausgangsort gesucht, so dass insgesamt möglichst wenige km zurückzulegen sind. Es handelt sich hier um einen Spezialfall des bekannten ”Problems des Handlungsreisenden”.

Eine untere Schranke für die Länge des gesuchten Weges ist offenbar

$$s = \sum_{i=1}^{4} \min_j \{d_{ij}\},$$

hier also $s = 4$.

Wir verzweigen nun in
$P_1$: Annahme, die Strecke $A - D$ wird genutzt, und
$P_2$: Annahme, die Strecke $A - D$ wird nicht genutzt.
Zu den ”Zweigen” $P_1$ und $P_2$ gehören die Matrizen:

$\mathbf{P_1}$ :

| | A | B | C | D | |
|---|---|---|---|---|---|
| A | $\infty$ | 4 | 2 | 1 | |
| B | 4 | $\infty$ | 1 | 2 | $(\mathbf{s = 4})$ |
| C | 2 | 1 | $\infty$ | 3 | |
| D | 1 | 2 | 3 | $\infty$ | |

$\mathbf{P_2}$ :

| | A | B | C | D |
|---|---|---|---|---|
| A | $\infty$ | 4 | 2 | 1 |
| B | 4 | $\infty$ | 1 | 2 |
| C | 2 | 1 | $\infty$ | 3 |
| D | 1 | 2 | 3 | $\infty$ |

$(\mathbf{s = 6})$

Der Zweig $P_2$ braucht wegen $s = 6$ nicht weiter untersucht zu werden. $P_1$ lässt sich nun weiter aufspalten mit der Annahme, dass $D - B$ benutzt wird $(P_{11})$ bzw. nicht benutzt wird $(P_{12})$. Man erhält für $P_{11}$ (die Tour $A-D-B-C-A$) die Länge $6$ und für $P_{12}$ (Tour $A-D-C-B-A$) Länge $9$, $A-D-B-C-A$ ist folglich optimal.

**Selbsttestaufgabe 3.3-1:**

*Begründen Sie, weshalb man sich bei der Lösung gewisser Probleme mit approximativen Algorithmen zufrieden gibt.*

# 4 Pseudodigraphen

## 4.1 Grundbegriffe

**Definition 4.1-1:**

**Pseudodigraph**

Ein **Pseudodigraph** (gerichteter Pseudograph) ist ein Tripel $\vec{P} = (E, B, v)$ bestehend aus einer Eckenmenge $E$, einer Bogenmenge $B$ und einer (Inzidenz-) Abbildung

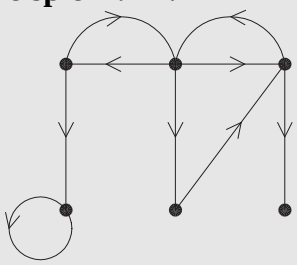
$$v : B \to E \times E.$$

**Ecken**
**Bögen**

Die Elemente von E heißen **Ecken**, die von B **Bögen** (oder gerichtete Kanten).

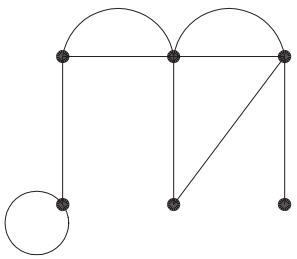
Im Unterschied zu einem Pseudographen sind die Kanten eines Pseudodigraphen gerichtet: $v(b) = (x, y)$ wird so gelesen, dass der Bogen $b$ "von x nach y" gerichtet ist; im Diagramm wird dies durch einen Pfeil angedeutet.
Ein Pseudodigraph trägt als Teil seiner Bezeichnung stets einen Pfeil (z. B. $\vec{P}$).

**Beispiel 4.1-1:**



$\vec{\mathrm{P}} = (\mathrm{E}, \mathrm{B}, \mathrm{v})$

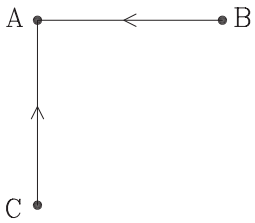
Zu jedem Pseudodigraphen $\vec{P}$ gehört ein Pseudograph $P$, der dadurch entsteht, dass statt des geordneten Paares $(x, y)$ das ungeordnete Paar $\{x, y\}$ betrachtet wird, anders gesagt: die Richtung der Kanten wird "vergessen". So entsteht z. B. aus dem in Beispiel 4.1-1 dargestellten Pseudodigraphen $\vec{P}$ der folgende Pseudograph $P$:



$\mathrm{P} = (\mathrm{E}, \mathrm{K}, \mathrm{v})$

Es liegt auf der Hand, dass sich immer dann ein Pseudodigraph als mathematisches Modell anbietet, wenn zweistellige Beziehungen mit einer Richtung zwischen

irgendwelchen Objekten untersucht werden sollen.
Wir betrachten etwa den folgenden gerichteten Graphen




Folgende reale Hintergründe wären z. B. denkbar:

1. $A, B$ und $C$ sind Telefonteilnehmer, und $B$ und $C$ haben beide $A$ angerufen.
2. $A, B$ und $C$ sind Straßenkreuzungen, und nach $A$ führen von $B$ und $C$ aus Einbahnstraßen.
3. $A, B$ und $C$ sind Menschen, und $B$ und $C$ haben $A$ als Vorfahr.

Als nächstes sollen nun einige Begriffe für Pseudodigraphen aufgeführt werden, die sich teilweise wegen der Analogie zum ungerichteten Fall von selbst verstehen. Deshalb werden wir nicht auf jeden dieser Begriffe näher eingehen.

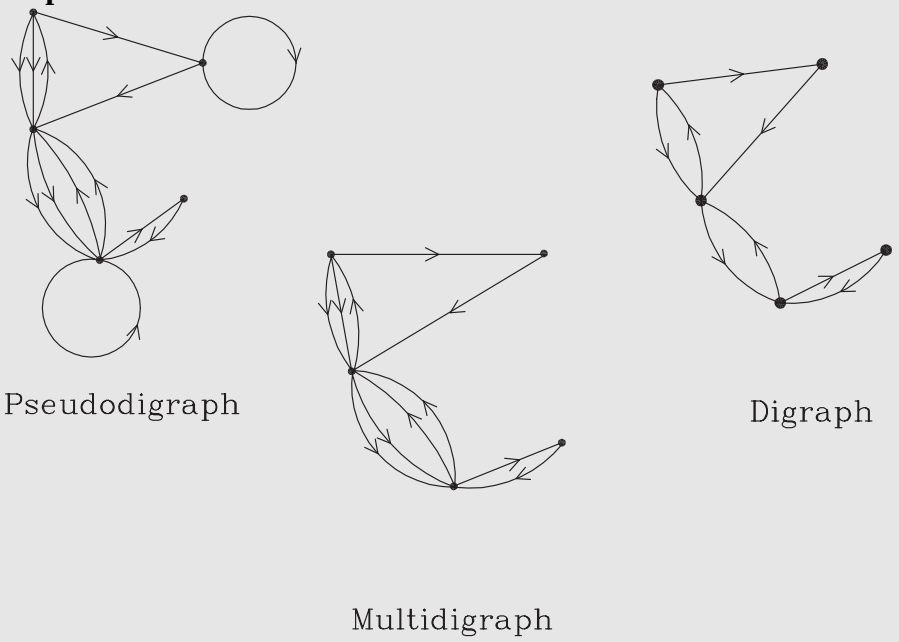
**Anfangsecke, Endecke**
**Nachfolger**
**Vorgänger**

**Definition 4.1-2:** *Ein Pseudodigraph $\vec{P} = (E, B, v)$ sei gegeben. Ist $b \in B$ mit $v(b) = (x, y)$, so heißt $x$ die **Anfangs**- und $y$ die **Endecke** des Bogens $b$. Man sagt auch: $y$ ist ein **Nachfolger** (oder eine Nachfolgerecke) von $x$, $x$ ein **Vorgänger** von $y$.*

**Inzidenz**
**Multidigraph**
**Digraph**

*Weitere Sprechweisen sind: $x$ und $y$ inzidieren mit $b$, $b$ inzidiert mit $x$ und $y$, $b$ ist ein Bogen von $x$ nach $y$. Ein Pseudodigraph ohne Schleifen heißt **Multidigraph**. Ein Multidigraph ohne Parallelbögen heißt ein **Digraph** (oder gerichteter Graph).*

**Beispiel 4.1-2:**




**Definition 4.1-3:** *$\vec{P} = (E, B, v)$ sei ein Pseudodigraph und $x \in E$ eine Ecke von $\vec{P}$. Dann ist der*

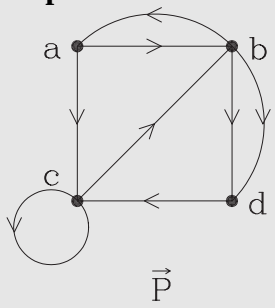
***Innengrad*** *von $x$ in $\vec{P}$, in Zeichen $\gamma^+(x, \vec{P})$, die Anzahl der Bögen von $\vec{P}$ mit $x$ als Endecke;* **Innengrad**

***Außengrad*** *von $x$ in $\vec{P}$, in Zeichen $\gamma^-(x, \vec{P})$, die Anzahl der Bögen von $\vec{P}$ mit $x$ als Anfangsecke;* **Außengrad**

***Grad*** *von $x$ in $\vec{P}$, in Zeichen $\gamma(x, \vec{P}) := \gamma^-(x, \vec{P}) + \gamma^+(x, \vec{P})$, die Anzahl der mit $x$ inzidenten Bögen. (Schleifen werden dabei doppelt gezählt.)* **Grad**

Es werden oft die Abkürzungen $\gamma^-(x)$ für $\gamma^-(x, \vec{P})$ und $\gamma^+(x)$ für $\gamma^+(x, \vec{P})$ verwendet. Zum ungerichteten Fall hat man den Zusammenhang $\gamma(x, \vec{P}) = \gamma(x, P) = \gamma(x)$.

**Beispiel 4.1-3:**



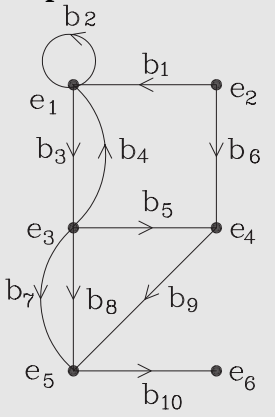

$\gamma^+(a) = 1, \gamma^-(a) = 2, \gamma(a) = 3$
$\gamma^+(b) = 2, \gamma^-(b) = 3, \gamma(b) = 5$
$\gamma^+(c) = 3, \gamma^-(c) = 2, \gamma(c) = 5$
$\gamma^+(d) = 2, \gamma^-(d) = 1, \gamma(d) = 3$

Die Begriffe Teilgraph und Untergraph werden in demselben Sinne wie im ungerichteten Fall verwendet. Als nächstes werden für Pseudodigraphen die den Kantenfolgen, Wegen, Kreisen usw. entsprechenden Begriffe eingeführt. Zur besseren Unterscheidung werden hier andere Wörter verwendet, so dass sich die obengenannten Begriffe nur auf den ungerichteten Fall beziehen.

**Bogenfolge**

**Definition 4.1-4:** *Eine Folge* $(e_1, b_1, e_2, b_2, \ldots, b_n, e_{n+1})$ *mit* $e_i \in E$ *und* $b_j \in B$ *heißt* ***Bogenfolge*** *(gerichtete Kantefolge), wenn für alle* $i$ *(*$1 \leq i \leq n$*)* $e_i$ *die Anfangsecke des Bogens* $b_i$ *und* $e_{i+1}$ *dessen Endecke ist.*

**Beispiel 4.1-4:**




Bogenfolge: $(e_2, b_1, e_1, b_2, e_1, b_3, e_3, b_8, e_5)$
Kantenfolge: $(e_2, b_1, e_1, b_2, e_1, b_4, e_3, b_3, e_1, b_4, e_3, b_8, e_5)$
Man beachte: In der angegebenen Kantenfolge wird für die Bögen $b_i$, die hier als ungerichtete Kanten aufgefasst werden, die gleiche Bezeichnung verwendet.

**geschlossene Bogenfolge**
**offene Bogenfolge**
**Länge einer Bogenfolge**
**Bogenzug**

**Definition 4.1-5:** *Eine Bogenfolge ist* ***geschlossen*** *, falls ihre Anfangsecke gleich ihrer Endecke ist, andernfalls ist sie* ***offen****. Die Anzahl der Bögen in einer Bogenfolge wird als* ***Länge*** *der Bogenfolge bezeichnet. Ein* ***Bogenzug*** *ist eine Bogenfolge,*

*die keinen Bogen zweimal enthält. ( Ein Bogenzug wird auch als gerichteter Kantenzug bezeichnet.) Eine **Bahn** (oder gerichteter Weg) ist ein Bogenzug, der keine Ecke zweimal enthält.*

**Bahn**

*Schließlich ist ein **Zyklus** (oder gerichteter Kreis) ein Bogenzug, in dem alle Ecken verschieden sind, bis auf die Anfangs- und die damit identische Endecke.*

**Zyklus**

**Beispiel 4.1-5:**
Wir betrachten wieder den Pseudodigraphen aus Beispiel 4.1-4.
$(e_2, b_1, e_1, b_2, e_1, b_3, e_3, b_8, e_5)$ ist ein Bogenzug, und z. B. ist
$(e_2, b_1, e_1, b_2, e_1, b_4, e_3, b_8, e_5)$ ein Kantenzug. Eine Bahn ist
$(e_2, b_1, e_1, b_3, e_3, b_5, e_4, b_9, e_5, b_{10}, e_6)$; ersetzt man in dieser Bahn $b_3$ durch $b_4$, so ist das Ergebnis ein Weg. Ein Zyklus ist $(e_1, b_3, e_3, b_4, e_1)$, ein Kreis ist z. B. $(e_3, b_8, e_5, b_9, e_4, b_5, e_3)$. Auch der Pseudodigraph aus Beispiel 4.1-3 enthält Zyklen, z. B. $(a, c, b, a)$.

Wie im ungerichteten Falle üblich, so können auch bei den Pseudodigraphen Bogenzüge, Bahnen und Zyklen nur durch die Folgen der beteiligten Ecken oder Bögen beschrieben werden, sofern dies nicht zu Missverständnissen führen kann. Die von diesen Bogenzügen herkommenden Teildigraphen werden in der Regel mit denselben Begriffen bezeichnet.

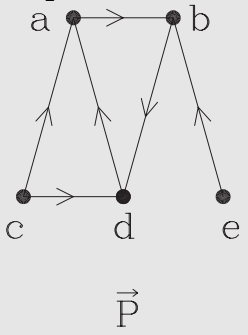
**Definition 4.1-6:** *Zwei Ecken $a$ und $b$ eines Pseudodigraphen $\vec{P}$ heißen **verbindbar** in $\vec{P}$, wenn es einen Weg in $P$ von $a$ nach $b$ gibt oder wenn $a = b$ gilt. Die Ecke $a$ heißt **einseitig verbindbar** mit der Ecke $b$, wenn eine Bahn von $a$ nach $b$ in $\vec{P}$ existiert oder $a = b$ gilt. Schließlich heißen $a$ und $b$ **stark verbindbar** , wenn je eine Bahn von $a$ nach $b$ und von $b$ nach $a$ existiert oder $a = b$ gilt.*

**verbindbare Ecken**

**einseitig verbindbare Ecke**

**stark verbindbare Ecken**

**Beispiel 4.1-6:**




$c$ und $e$ sind verbindbar in $P$.
$c$ ist in $\vec{P}$ einseitig verbindbar mit $b$, jedoch $b$ nicht mit $c$.
$a$ und $b$ sind in $\vec{P}$ stark verbindbar.

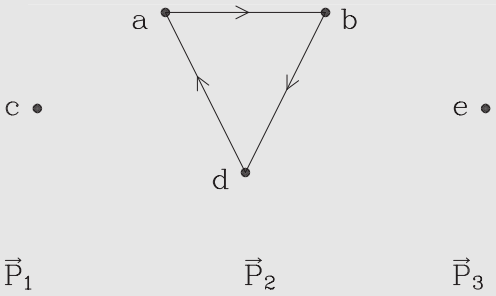
Der Begriff der starken Verbindbarkeit für Pseudodigraphen führt nun auch zu einem eigenen Zusammenhangsbegriff:

**Definition 4.1-7:** *Ein Pseudodigraph $\vec{P}$ heißt **stark zusammenhängend**, wenn je zwei Ecken in $\vec{P}$ stark verbindbar sind.*

Wie bei ungerichteten Pseudographen ist bei Pseudodigraphen die Verbindbarkeit bzw. die starke Verbindbarkeit eine Äquivalenzrelation. Die von den Äquivalenzklassen aufgespannten Untergraphen werden die **Komponenten** bzw. die **starken Komponenten** des Pseudodigraphen genannt.

Die Äquivalenzklassen sind bei der Verbindbarkeitsrelation die Äquivalenzklassen des zugrundeliegenden Pseudographen, und zwischen den Ecken aus zwei verschiedenen Komponenten existieren keine Bögen. Anders verhält es sich bei den starken Komponenten: hier kann es Bögen geben, jedoch haben dann alle Bögen zwischen zwei Komponenten jeweils die gleiche Richtung.

**Beispiel 4.1-7:**
$\vec{P}$ aus Beispiel 4.1-6 hat nur eine Komponente, jedoch drei starke Komnponenten:

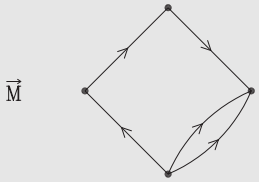



Wie bei Pseudographen sind auch bei Pseudodigraphen diejenigen von speziellem Interesse, die kreisfrei (im gerichteten Sinne) sind:

**azyklisch**

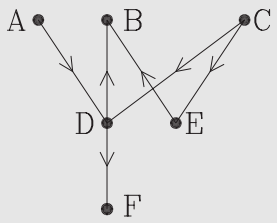
**Definition 4.1-8:** *$\vec{P}$ heißt **azyklisch**, wenn $\vec{P}$ keinen Zyklus enthält. Ein azyklischer Pseudodigraph $\vec{P}$ kann keine Schleifen enthalten und ist somit stets ein Multigraph.*

**Beispiel 4.1-8:**



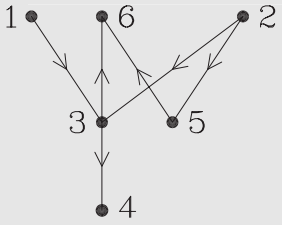

$\vec{M}$ ist azyklisch, aber
$M$ enthält Kreise.

**Beispiel 4.1-9:**
Wir stellen uns vor, jemand will ein wissenschaftliches Buch mit sechs Kapiteln $A, B, \ldots, F$ schreiben, wobei manche Kapitel inhaltlich auf anderen aufbauen. Die Abhängigkeiten mögen in dem folgenden Digraphen dargestellt sein:




(Z.B. baut $D$ auf $A$ auf,
$F$ auf $D$ usw..)

Um das Buch zu erstellen, müssen die sechs Kapitel in eine sinnvolle Reihenfolge gebracht werden: baut $Y$ auf $X$ auf, so muss im Buch $X$ vor $Y$ stehen. Eine sinnvolle Numerierung der Kapitel ist in folgendem Diagramm eingetragen:




Eine solche Numerierung wäre natürlich nicht möglich, wenn der Digraph einen Zyklus enthielte. (Das Buch könnte in diesem Fall nicht geschrieben werden.)

Das letzte Beispiel motiviert die folgende Definition.

**Definition 4.1-9:** *Ein Multidigraph $\vec{M} = (E, B, v)$ sei gegeben mit $n$ Ecken, also $|E| = n$. Eine Nummerierung der Ecken, also eine bijektive Abbildung*

$$\lambda : E \to \{1, 2, \ldots, n\},$$

*heißt eine **topologische Anordnung**, wenn für jeden Bogen $(e, f) \in B$ gilt $\lambda(e) < \lambda(f)$, d. h. die Anfangsecke hat stets eine kleinere Nummer als die Endecke.*

**topologische Anordnung**

Ohne Beweis ist klar, dass es für $\vec{M}$ nur dann eine topologische Anordnung geben kann, wenn $\vec{M}$ keinen Zyklus enthält. Interessanterweise ist diese notwendige Bedingung auch hinreichend:

**Satz 4.1-1:** *Für einen Multidigraphen $\vec{M}$ sind folgende Aussagen äquivalent:*

- *i. $\vec{M}$ ist azyklisch.*

- *ii. Jeder nichtleere Untergraph $\vec{U}$ von $\vec{M}$ enthält eine Ecke x mit $\gamma^-(x, \vec{U}) = 0$.*
- *iii. $\vec{M}$ besitzt eine topologische Anordnung.*

**Beweis:** $i) \rightarrow ii)$ :$\vec{U}$ sei ein nichtleerer Untergraph, $u_1, u_2, \ldots, u_p$ seien die Ecken einer maximalen Bahn in $\vec{U}$ (d. h. es gibt in $\vec{U}$ keine Bahn mit mehr als $p$ vielen Ecken). Wir machen nun die Annahme $\gamma^-(u_p, \vec{U}) > 0$ und leiten daraus einen Widerspruch her. Es muss nämlich nun eine Ecke $u$ in $\vec{U}$ mit einem Bogen $(u_p, u)$ geben. Jetzt sind zwei Fälle möglich:
Fall 1: $u = u_i$ für ein $i$ mit $1 \leq i \leq p$. In diesem Fall folgt, dass $u_i, u_{i+1}, \ldots, u_p$ die Ecken eines Zyklus in $\vec{U}$ und damit auch in $\vec{M}$ sind, man hat einen Widerspruch.
Fall 2: $u$ stimmt mit keinem der $u_i$ überein. Dann ist $u_1, u_2, \ldots, u_p, u$ eine Bahn in $\vec{U}$ von Länge $p$ (mit $p + 1$ Ecken), womit sich ebenfalls ein Widerspruch ergibt.
Damit ist in jedem Fall ein Widerspruch hergeleitet.
$ii) \rightarrow iii)$ : Eine topologische Anordnung $\gamma : E \rightarrow \{1, 2, \ldots, n\}$ wird induktiv auf folgende Weise konstruiert. Es sei $x$ eine Ecke von $\vec{M}$ mit $\gamma^-(x, \vec{M}) = 0$. $x$ wird mit $n$ numeriert, d. h. wir setzen $\lambda(x) = n$. Hat man die Nummern $i+1, i+2, \ldots, n$ bereits vergeben (für ein $i$ mit $1 \leq i < n$), so sei $\vec{U_i}$ der von den noch nicht numerierten Ecken aufgespannte Untergraph. Man wählt eine Ecke $y$ von $\vec{U_i}$ mit $\gamma^-(y, \vec{U_i}) = 0$ und setzt $\lambda(y) = i$.
Es muss nun gezeigt werden, dass das so konstruierte $\lambda$ eine topologische Anordnung ist. Es sei $(e, f)$ ein Bogen von $\vec{M}$. Offenbar muss bei der Konstruktion von $\lambda$ die Ecke $f$ früher als die Ecke $e$ numeriert worden sein (und somit $\lambda(f) > \lambda(e)$ gelten), denn bei der Numerierung von $e$ hat $e$ im "Restgraphen" den Außengrad $0$, d. h. $f$ kann nicht zu den noch nicht numerierten Ecken gehören.
$iii) \rightarrow i)$ : Diese Implikation ist offensichtlich.

Der Beweis der Impliaktion ii) $\rightarrow$ iii) in obigem Satz liefert auch einen Algorithmus, der einen vorgegebenen Multidigraphen auf dessen Freiheit von Zyklen testet, indem er eine topologische Anordnung für den Multidigraphen zu konstruieren versucht. Gelingt diese Konstruktion nicht, so folgt aus dem Satz auch, dass $\vec{M}$ nicht azyklisch ist.

**Algorithmus "Azyklisch"** Gegeben sei ein Multidigraph $\vec{M} = (E, B, v)$ mit $|E| = n$. Es wird eine topologische Anordnung für $\vec{M}$ konstruiert oder die Information geliefert, dass $\vec{M}$ nicht azyklisch ist.

**Algorithmus**: Azyklisch

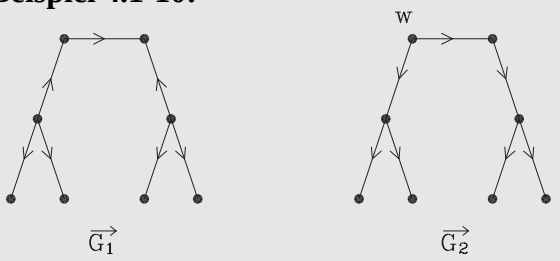
```
begin
   setze i:=n;
   while i > 0 do
   begin
      if es gibt kein x ∈ E in M mit γ^-(x, M) = 0
         then M ist nicht azyklisch, goto ende;
         else wähle x in M mit γ^-(x, M) = 0,
```

```
                setze λ(x) := i, M⃗ := M⃗ \ {x },
                i :=  i − 1;
        end
    ende: end
```

Im ungerichteten Fall ist ein zusammenhängender Pseudograph ohne Kreis stets ein spezieller Graph von sehr einfacher Struktur, nämlich ein Baum. Bei Pseudodigraphen ist die Situation komplexer. Ein azyklischer Pseudodigraph ist stets ein Multidigraph, kann jedoch (ungerichtete) Kreise und auch Parallelkanten enthalten - in dem Beispiel 4.1-8 ist beides der Fall. Enthält ein Multidigraph auch keinen ungerichteten Kreis, so handelt es sich um einen Digraphen $\vec{G}$, dessen zugrundeliegender (ungerichteter) Graph ein Baum ist. Einen solchen Digraphen $\vec{G}$ nennt man einen **gerichteten Baum**, wenn er eine Ecke $w$ enthält, die mit jeder anderen Ecke einseitig verbindbar ist. Eine solche Ecke heißt **Wurzel**.

**gerichteter Baum**

**Beispiel 4.1-10:**




$\vec{G}_2$ ist ein gerichteter Baum, $\vec{G}_1$ nicht, obwohl $G_1$ ein Baum ist.

Auch in vielen Anwendungen gerichteter Graphen stehen die Begriffe im Mittelpunkt, die sich auf die zugrundeliegenden ungerichteten Graphen beziehen. So werden z. B. oft Kreise (statt Zyklen) betrachtet. Den Kreisen und Schnitten wird jedoch zusätzlich eine Orientierung zugeordnet - darauf wird im nächsten Abschnitt eingegangen. Die im Abschnitt 1.3 eingeführten Begriffe wie spannender Wald, Rang, Nullität usw. behalten selbstverständlich für Multidigraphen ihre Gültigkeit, wenn sie jeweils auf den zugrundeliegenden Multigraphen bezogen werden. Gleiches gilt für die dort bewiesenen Sätze, insbesondere die über die Vektorräume $\mathbf{C}^*$ und $\mathbf{S}^*$.

**Selbsttestaufgabe 4.1-1:**

*Erläutern Sie die Begriffe "kreisfrei" und "azyklisch" für Pseudodigraphen, und illustrieren Sie den Unterschied anhand eines Beispiels.*

## 4.2 Multigraphen und Matrizen

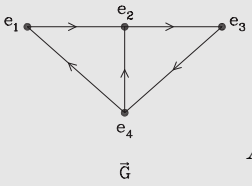
**Adjazenzmatrix**

Analog zum ungerichteten Fall wird die **Adjazenzmatrix** nur für Digraphen (ohne Parallelbögen) erklärt. Ein Digraph $\vec{G} = (E, B)$ sei gegeben mit $E = \{e_1, \ldots, e_n\}$ und $b = (e_i, e_j)$ (mit geeigneten $i, j$) für $b \in B$. Die Adjazenzmatrix $A(\vec{G}) = (a_{ij})$ von $\vec{G}$ ist die $n \times n$- Matrix mit folgenden Einträgen:

$$a_{ij} = \begin{cases} 1, & \text{falls } (e_i, e_j) \in B \\ 0, & \text{sonst.} \end{cases}$$

Statt $A(\vec{G})$ wird auch hier oft nur $A$ geschrieben.

**Beispiel 4.2-1:**

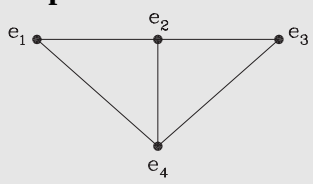



$\vec{G}$

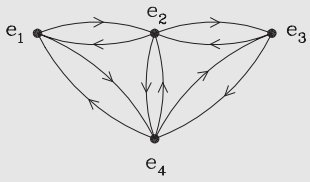
$$A(\vec{G}) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Die Adjazenzmatrix eines Digraphen ist i.a. nicht symmetrisch - sie ist es genau dann, wenn $\vec{G}$ mit jedem Bogen $(e_i, e_j)$ auch den umgekehrt gerichteten Bogen $(e_j, e_i)$ enthält. Interpretiert man einen ungerichteten Graphen $H$ als den Digraphen $\vec{H}$, der für jede Kante von $H$ zwei entgegengesetzt gerichtete Bögen enthält, so berühren sich die beiden Definitionen der Adjazenzmatrix wieder.

**Beispiel 4.2-2:**




H




$\vec{H}$

$H$ und $\vec{H}$ haben die gleiche Adjazenzmatrix:

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Es gilt nun auch der folgende Satz.

**Satz 4.2-1:** *Ist $A$ die Adjazenzmatrix eines Digraphen $\vec{G}$, so liefert der $(i,j)$-Eintrag $a_{ij}^{(r)}$ der $r$-ten Potenz $A^r$ die Anzahl der Bogenfolgen von Länge $r$, die in $\vec{G}$ von $e_i$ nach $e_j$ verlaufen.*

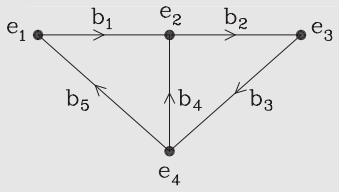
**Beweis:** Der Beweis verläuft völlig analog zu dem der ungerichteten Variante (vgl. Satz 2.2-1) und wird hier nicht wiederholt.

Wir kommen nun zum Begriff der **Inzidenzmatrix**. **Inzidenzmatrix**

**Definition 4.2-1:** *Es sei ein Multidigraph $\vec{M} = (E, B, v)$ gegeben. Die Mengen $E$ und $B$ seien durchnumeriert, $E = \{e_1, \ldots, e_n\}$ und $B = \{b_1, \ldots, b_m\}$. Die Inzidenzmatrix $I(\vec{M})$ (oder einfach $I$) des Multidigraphen $\vec{M}$ ist die $n \times m$-Matrix $(i_{rs})$ mit*

$$i_{rs} = \begin{cases} 1, & \text{falls die Ecke } e_r \text{ die Endecke des Bogens } b_s \text{ ist} \\ -1, & \text{falls die Ecke } e_r \text{ die Anfangsecke des Bogens } b_s \text{ ist} \\ 0, & \text{sonst.} \end{cases}$$

**Beispiel 4.2-3:**
Wir betrachten den Digraphen aus Beispiel 4.2-1 mit den folgenden Numerierungen:




Als Inzidenzmatrix ergibt sich:

$$I(\vec{M}) = \begin{pmatrix} -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 \end{pmatrix}$$

Für jede Inzidenzmatrix $I(\vec{M})$ eines Multidigraphen $\vec{M}$ gelten folgende Aussagen, wie sich unmittelbar aus der Definition ergibt:

- Ersetzt man jede -1 durch +1, so ergibt sich die Inzidenzmatrix $I(M)$ des Multigraphen $M$.
- Jede Spalte von $I(\vec{M})$ enthält einmal den Eintrag +1 und einmal -1.
- In der zur Ecke $e_r$ gehörenden Zeile von $I(\vec{M})$ stehen $\gamma^+(e_r, \vec{M})$ viele Einträge +1, die -1 kommt $\gamma^-(e_r, \vec{M})$ mal vor.

- Umnummerierung der Ecken- bzw. Bogenmenge führt zu Vertauschungen von Zeilen bzw. Spalten in $I(\vec{M})$. Zwei Multidigraphen, deren Inzidenzmatrizen bis auf solche Vertauschungen identisch sind, sind isomorph.

Wie sich nun zeigen wird, können auch bei Inzidenzmatrizen von Multidigraphen die Begriffe Rang und Determinante zur Untersuchung der Graphenstruktur genutzt werden. Für das folgende sei ein zusammenhängender Mulidigraph $\vec{M}$ mit $(n \times m)$-Inzidenzmatrix $I$ gegeben. Rang und Determinante werden immer im Körper $\mathbb{R}$ der reellen Zahlen gebildet.

**Hilfssatz 4.2-1:** *Die Summe irgendwelcher $t$ Zeilen von $I$ (mit $t < n$) enthält mindestens ein von Null verschiedenes Element.*

**Beweis:** Es sei $E_t$ die Menge der Ecken, die den $t$ ausgewählten Zeilen entsprechen. Da $\vec{M}$ zusammenhängend ist, können wir einen Bogen $b$ in $\vec{M}$ wählen, der eine Ecke von $E_t$ mit einer nicht zu $E_t$ gehörenden Ecke verbindet. Da die $b$ entsprechende Spalte von $I(\vec{M})$ einmal die 1 und einmal die -1 enthält und nur genau eine der beiden dadurch angesprochenen Zeilen zu den $t$ aufsummierten Zeilen gehört, bleibt in der Summe in der betreffenden Komponente entweder 1 oder -1 stehen.

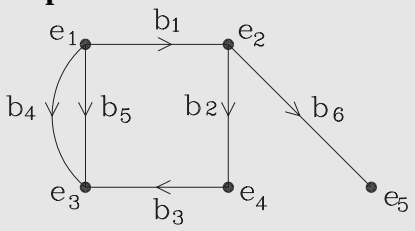
**Hilfssatz 4.2-2:** *Es ist $rg(I) \leq n - 1$.*

**Beweis:** Da die Summe aller Zeilen von $I$ den Nullvektor ergibt, sind die Zeilen linear abhängig, der Rang kann folglich höchstens $n - 1$ sein.

**Hilfssatz 4.2-3:** *Die Spalten von $I$, die zu den Bögen eines Kreises gehören, sind linear abhängig.*

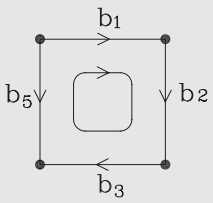
**Beweis:** Wir setzen zunächst voraus, dass es sich bei dem Kreis um einen Zyklus handelt. Dann ergibt die Summe dieser Spalten den Nullvektor. Da nämlich eine Eins in einer Spalte eine Anfangsecke des betreffenden Bogens bezeichnet, gibt es zu jeder solchen Eins innerhalb der betrachtenden Spaltenmenge genau eine zweite Spalte mit -1 an dieser Stelle. Nun handele es sich bei dem betrachteten Kreis nicht um einen Zyklus. Wählt man nun eine "Durchlaufrichtung" für diesen Kreis, so werden einige Kanten in diese Richtung, andere entgegen dieser Richtung orientiert sein. Da das Umdrehen der Kanten, die entgegen dieser Richtung orientiert sind, zu einem Zyklus führen würde, folgt: die Summe der zu dem Kreis gehörenden Spalten ergibt den Nullvektor, wenn die zu den umgedrehten Kanten gehörenden Spalten mit -1 multipliziert werden.

**Beispiel 4.2-4:**




$$I = \begin{pmatrix} -1 & 0 & 0 & -1 & -1 & 0 \\ 1 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Der Kreis $(b_1, b_2, b_3, b_5)$ ist kein Zyklus:




Bei Wahl der im Diagramm angedeuteten Orientierung im Uhrzeigersinn ist $b_5$ entgegengesetzt orientiert. Somit ergibt die Summe der Spalten 1 bis 3 und des Negativen der Spalte 5 den Nullvektor.

Auch bei Multidigraphen erweist es sich als nützlich, eine **reduzierte Inzidenzmatrix** $I_r$ zu betrachten, die sich ergibt, wenn die $r$-te Zeile von $I$ weggelassen wird. Die zu dieser Zeile gehörende Ecke $e_r$ wird auch als **Bezugsecke** der reduzierten Inzidenzmatrix $I_r$ bezeichnet.

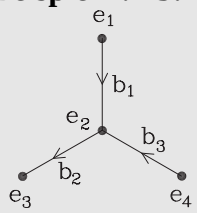
**reduzierte Inzidenzmatrix**
**Bezugsecke**

Der folgende Satz bezieht sich auf den Fall, dass der Multigraph $M$, der dem betrachteten Multidigraphen $\vec{M}$ zugrundeliegt, ein Baum ist.

**Satz 4.2-2:** *Ist $M$ ein Baum und $I_r$ eine reduzierte Inzidenzmatrix von $\vec{M}$, so gilt für die reelle Determinante $det(I_r) = \pm 1$.*

**Beweis:** Der Beweis verläuft bis auf den Induktionsanfang völlig analog dem zu Satz 2.2.15: hat $\vec{M}$ nur einen Bogen und zwei Ecken, so kann hier auch $det(I_r) = -1$ sein.

**Beispiel 4.2-5:**




$$I = \begin{pmatrix} -1 & 0 & 0 \\ 1 & -1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Streichen der zweiten Zeile (Bezugsecke $e_2$) führt zu

$$I_2 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

mit $det(I_2) = +1$.

Wir kommen nun zu zwei interessanten Folgerungen. Nach wie vor sei ein zusammenhängender Multidigraph $\vec{M}$ mit $(n \times m)$-Inzidenzmatrix $I$ gegeben. Die Nachweise dieser Folgerungen ergeben sich - völlig analog den entsprechenden Folgerungen Folgerung 2.2-1 und Folgerung 2.2-2 - aus Hilfssatz 4.2-2, Hilfssatz 4.2-3 und Satz 4.2-2.

**Folgerung 4.2-1:** *Es ist* $rg(I) = n - 1$.

**Folgerung 4.2-2:** *Eine quadratische* $(n-1) \times (n-1)$*-Untermatrix einer reduzierten Inzidenzmatrix* $I_r$ *ist genau dann regulär, wenn die ausgewählten Spalten zu einem Gerüst von* $\vec{M}$ *(bzw.* $M$*) gehören.*

Abschließend sei auf die folgende Verallgemeinerung von Folgerung 4.2-1 hingewiesen: besteht $M$ aus $p$ vielen Komponenten, so gilt für eine Inzidenzmatrix $I$

$$rg(I) = n - p = \rho(M).$$

Um - analog zum ungerichteten Fall - eine Kreis-Bogen-Matrix und eine Schnitt-Bogen-Matrix einführen zu können, ordnen wir zunächst jedem Kreis und jedem Schnitt eine Orientierung zu. Dies ist, ohne es ausdrücklich zu sagen, im Beweis von Hilfssatz 4.2-3 bereits verwendet worden.

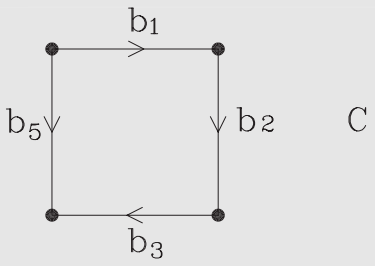
**Orientierung**

**Definition 4.2-2:** *Es sei* $C$ *ein Kreis in dem Multidigraphen* $\vec{M}$*. (*$C$ *muss kein Zyklus sein.) Unter einer* ***Orientierung*** *von* $C$ *versteht man die Festlegung einer "Durchlaufrichtung" des Kreises, ohne allerdings Anfangs- und Endecke festzulegen. Es gibt also stets zwei mögliche Orientierungen, man spricht von einem* ***orientierten Kreis****.*

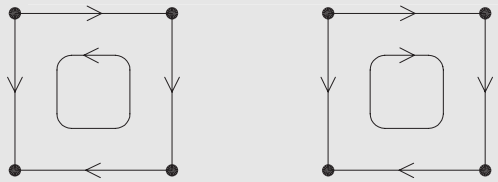
**orientierter Kreis**

Im Grunde bedeutet der Übergang zum orientierten Kreis ein Zurückgreifen auf die ursprüngliche Definition eines Kreises als Kantenzug, der als Folge von Kanten definiert war und ebenfalls eine Reihenfolge beinhaltete (siehe Kapitel 1).
Im Diagramm wird die Orientierung eines Kreises häufig durch eine zusätzliche geschlossene Linie angedeutet, die man mit einem Pfeil versieht.

**Beispiel 4.2-6:**
Der Multidigraph $\vec{M}$ aus Beispiel 4.2-4 enthält z. B. folgenden Kreis:



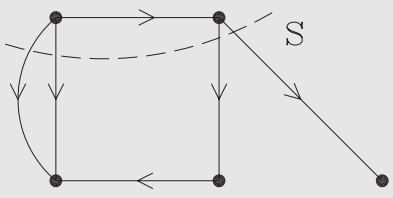

$C$ kann auf zwei Weisen zu einem orientierten Kreis gemacht werden:



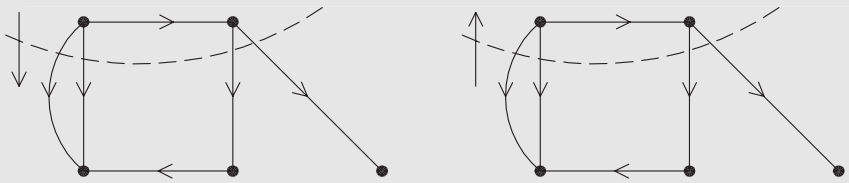
**Definition 4.2-3:** *Auch einem Schnitt $S$ eines Multidigraphen $\vec{M}$ können zwei mögliche Orientierungen zugeordnet werden, das Resultat ist jeweils ein* ***orientierter Schnitt****. Im Diagramm wird die Orientierung eines Schnitts häufig durch einen zusätzlichen Pfeil angegeben.*

**orientierter Schnitt**

**Beispiel 4.2-7:**
Es wird wieder der Multidigraph aus Beispiel 4.2-4 zugrundegelegt, und zwar mit folgendem Schnitt:



$S$ kann auf zwei Weisen zu einem orientierten Schnitt gemacht werden:



Im folgenden wird von orientierten Kreisen bzw. Schnitten die Rede sein, wobei jedoch die Bezeichnungen (z. B. $\mathbf{C}$ oder $\mathbf{S}$) beibehalten werden.

**Definition 4.2-4:** *Es sei nun wieder ein Multidigraph $\vec{M} = (E, B, v)$ gegeben, wobei die Mengen $E$ und $B$ durchnumeriert seien, $E = \{e_1, \ldots, e_n\}$ und $B = \{b_1, \ldots, b_m\}$. Weiter seien die - auf den Multigraphen $M$ bezogenen - Mengen $\mathbf{C}$ und $\mathbf{S}$ ebenfalls durchnumeriert:*
$\mathbf{C} = \{\mathbf{C_i}|\mathbf{1} \leq \mathbf{i} \leq \mathbf{2}^{\mu} - \mathbf{1}\}$, $\mathbf{S} = \{\mathbf{S_j}|\mathbf{1} \leq \mathbf{j} \leq \mathbf{2}^{\rho} - \mathbf{1}\}$.
*Außerdem sei jedem Kreis und jedem Schnitt eine beliebige feste Orientierung zugeordnet.*

**Kreis-Bogen-Matrix**

*Die **Kreis-Bogen-Matrix** $C(\vec{M}) = c_{ij}$ ist die in folgender Weise definierte $((2^{\mu} - 1) \times m)$-Matrix:*

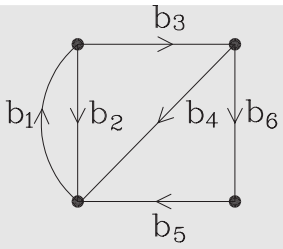
$$c_{ij} = \begin{cases} 1, & \textit{falls } C_i\ b_j \textit{ enthält und die Orientierung des } b_j \\ & \textit{enthaltenden Kreises von } C_i \textit{ mit der Richtung} \\ & \textit{von } b_j \textit{ übereinstimmt} \\ -1, & \textit{falls } C_i\ b_j \textit{ enthält und Orientierung und Richtung} \\ & \textit{(s.o.) nicht übereinstimmen} \\ 0, & \textit{sonst.} \end{cases}$$

**Schnitt-Bogen-Matrix**

*Die **Schnitt-Bogen-Matrix** $S(\vec{M}) = (s_{ij})$ ist die folgendermaßen erklärte $((2^{\rho} - 1) \times m)$-Matrix:*

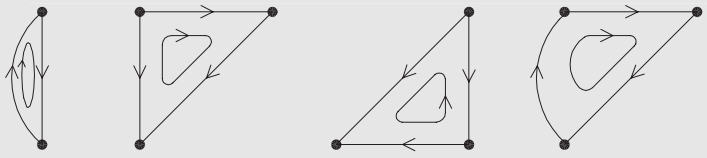
$$s_{ij} = \begin{cases} 1, & \textit{falls } S_i\ b_j \textit{ enthält und die Orientierung von } S_i \\ & \textit{mit der Richtung von } b_j \textit{ übereinstimmt} \\ -1, & \textit{falls } S_i\ b_j \textit{ enthält und Orientierung und Richtung} \\ & \textit{(s.o.) nicht übereinstimmen} \\ 0, & \textit{sonst.} \end{cases}$$

**Beispiel 4.2-8:**
Wir betrachten die folgende gerichtete Variante des in Beispiel 2.3-1 analysierten Multigraphen:

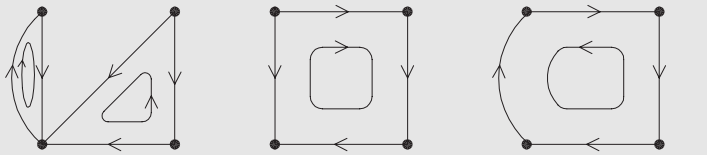



$\vec{\mathrm{M}}$

Die Elemente von $\mathbf{C}$ stellen sich im Diagramm so dar:



$\mathrm{C}_1$ $\mathrm{C}_2$ $\mathrm{C}_3$ $\mathrm{C}_4 = \mathrm{C}_1 + \mathrm{C}_2$

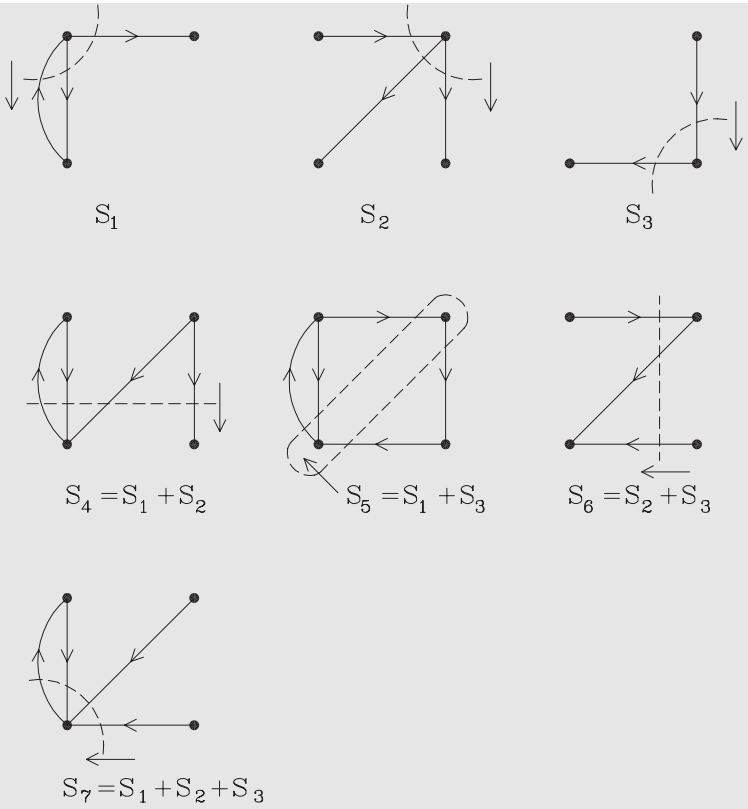


$\mathrm{C}_5 = \mathrm{C}_1 + \mathrm{C}_3$ $\mathrm{C}_6 = \mathrm{C}_2 + \mathrm{C}_3$ $\mathrm{C}_7 = \mathrm{C}_1 + \mathrm{C}_2 + \mathrm{C}_3$

Außer $C_5$ sind alle $C_i$ Kreise, ihre Orientierungen werden willkürlich und unabhängig voneinander ausgewählt. Da $C_5$ aus $C_1$ und $C_3$ disjunkt zusammengesetzt ist, ist hier keine Orientierung neu festzulegen. Es ergibt sich nun die folgende Kreis-Bogen-Matrix:

$$C(\vec{M}) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & -1 & -1 \\ 0 & -1 & 1 & 0 & 1 & 1 \\ -1 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}$$

Als nächstes führen wir Darstellungen der mit Orientierungen versehenen Schnitte $S_1, \ldots, S_7$ auf:




Als Schnitt-Bogen-Matrix ergibt sich:

$$S(\vec{M}) = \begin{pmatrix} -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ -1 & 1 & 0 & 1 & 0 & 1 \\ -1 & 1 & 1 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 & 1 & 0 \\ -1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Abschließend werden nun einige algebraische Eigenschaften und Zusammenhänge der Matrizen $I(\vec{M}), C(\vec{M})$ und $S(\vec{M})$ (kurz: $I, C$ und $S$) aufgezeigt. Wieder sei ein Multidigraph $\vec{M}$ vorgegeben. Aus dem Beweis von Hilfssatz 4.2-3 ergibt sich sofort, ähnlich wie im ungerichteten Fall (vgl. Folgerung 2.3-1):

**Folgerung 4.2-3:** *Für die Inzidenzmatrix $I$ und die Kreis-Bogen-Matrix $C$ gilt die Beziehung*

$$I \cdot C^t = 0.$$

Es sei noch einmal daran erinnert, dass die Matrizenrechnung in diesem Kapitel über dem Körper $\mathbb{R}$ der reellen Zahlen auszuführen ist. Der folgende Satz ist die Entsprechung zu Satz 2.3-1:

**Satz 4.2-3:** *Es ist* $rg(C) = \mu(\vec{M})$.

Da die mit einer festen Ecke inzidenten Bögen immer einen Schnitt bilden, ist es auch hier so, dass eine Auswahl bestimmter Zeilen von $S$ zusammen $I$ ergibt. (Dabei können, wegen verschiedener möglicher Orientierungen, einzelne Zeilen mit -1 multipliziert sein.) Insofern hat man hier die folgende Verstärkung von Folgerung 4.2-3:

**Satz 4.2-4:** *Es gilt die Beziehung* $S \cdot C^t = 0$.

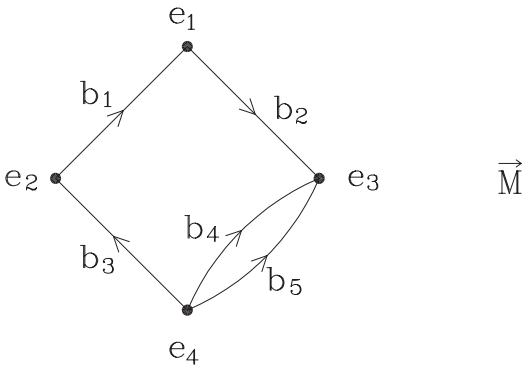
**Beweis:** Es seien $S_i$ und $C_j$ ein orientierter Schnitt bzw. Kreis von $\vec{M}$. Wir wissen schon, dass $S_i$ und $C_j$ eine gerade Anzahl von Kanten gemeinsam sind. Durchläuft man, ähnlich wie im Beweis von Satz 1.3-1, den Kreis $C_j$ in Richtung der gegebenen Orientierung, so ist ersichtlich, dass die dabei vorkommenden zu $S_i$ und $C_j$ gehörenden Kanten gleich oft parallel zur Orientierung des Schnitts wie gegenläufig dazu auftreten. Dies bedeutet also, dass beim Produkt der $i$-ten Zeile von $S$ mit der $j$-ten Zeile von $C$ ebenso oft zwei gleiche Einträge (1 oder -1) aufeinander treffen wie zwei ungleiche, das Produkt (als Summe dieser Einzelprodukte) ist folglich 0.

Zum schluss soll - ohne Beweis - auf die folgende Aussage über den Rang von $S(\vec{M})$ hingewiesen werden:

**Satz 4.2-5:** *Es ist* $rg(S) = \rho(\vec{M})$.

**Selbsttestaufgabe 4.2-1:**

*Gegeben sei der im folgenden Bild dargestellte Multidigraph* $\vec{M}$*:*




*Stellen Sie die Inzidenzmatrix* $I$ *und eine Kreis-Bogen-Matrix* $C$ *auf, und rechnen Sie die Beziehung* $I \cdot C^t = 0$ *nach.*

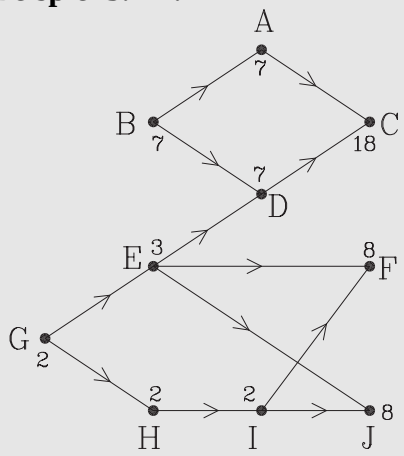
# 5 Bewertungen

## 5.1 Ecken-, Kanten- und Bogenbewertungen

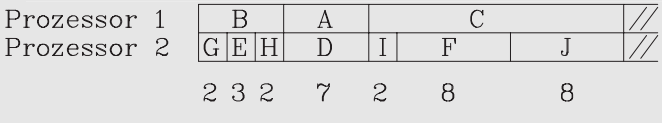
**Bewertung**

Die Struktur von Pseudographen bzw. Pseudodigraphen wird reichhaltiger, wenn den Ecken, Kanten oder Bögen zusätzlich noch irgendwelche Zahlen zugeordnet werden, man spricht dann von **Bewertungen** . In den meisten Anwendungen von Pseudo(di)graphen hat man es mit bewerteten Strukturen zu tun, wobei die zugeordneten Zahlenwerte z. B. Verarbeitungszeiten von Teilaufgaben, Benutzungskosten von Leitungen oder deren Ausfallwahrscheinlichkeiten sein können usw.
Einer formalen Definition von Bewertungen sollen einige Beispiele vorangestellt werden.

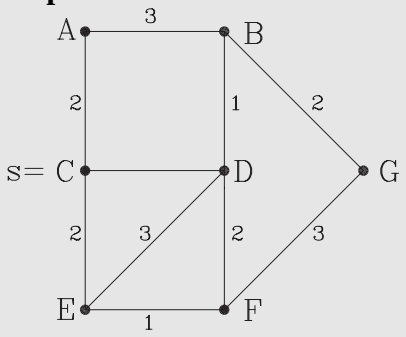
**Beispiel 5.1-1:**



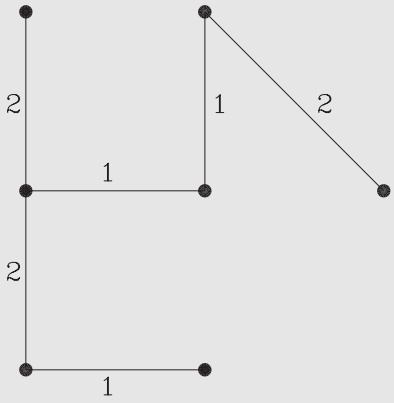

In dem hier dargestellten Digraphen mögen die Ecken Teilaufgaben einer von einem Rechner zu bearbeitenden Gesamtaufgabe darstellen, die Eckenbewertungen die Bearbeitungszeiten der Teilaufgaben. Ein Bogen von $X$ nach $Y$ soll bedeuten, dass $X$ fertig bearbeitet sein muss, bevor $Y$ begonnen werden kann. Besitzt der Rechner zwei Prozessoren, die parallel arbeiten können, wobei jedoch eine begonnene Aufgabe stets ohne Unterbrechung zuende geführt werden muss, so ist der im folgenden Balkendiagramm dargestellte Ablaufplan hinsichtlich der Gesamtablaufzeit optimal:



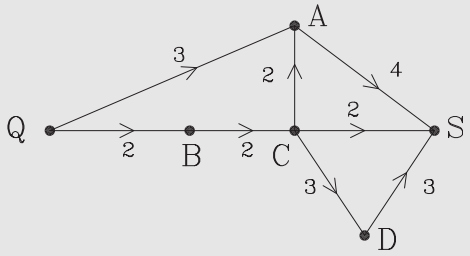
**Beispiel 5.1-2:**




Das Diagramm zeigt den mehrfach - zuletzt in Beispiel 3.3-1 - vorgekommenen Graphen, der hier die möglichen Leitungen in einem zu errichtenden Datennetzwerk darstellen soll. Die zusätzlich eingetragenen Kantenbewertungen mögen die Erstellungskosten der einzelnen Leitungen sein. Soll das Netz nun möglichst wirtschaftlich errichtet werden, so sucht man nach einem Gerüst mit der Eigenschaft, dass die Summe der Bewertungen der Gerüstkanten möglichst klein ist; es ist z. B. das folgende Gerüst optimal:

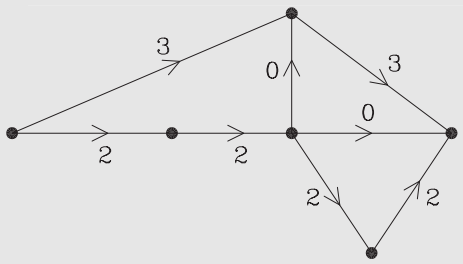



**Beispiel 5.1-3:**




Der hier dargestellte Digraph trägt eine Kantenbewertung. Wir stellen uns vor, es handele sich um ein Transportnetz, wo (entlang der Bögen) Güter von der "Quelle" $Q$ zur "Senke" $S$ transportiert werden sollen. Die Bewertung stellt eine

obere Kapazitätsbeschränkung dar; z. B. bedeutet die Zahl ”3” auf dem Bogen von $Q$ nach $A$, dass höchstens 3 Gütereinheiten entlang des Bogens transportiert werden können. Im folgenden Diagramm ist ein maximaler ”Fluss” von 5 Einheiten dargestellt, der die gegebene Beschränkung respektiert:




Um größeren Spielraum zu haben, lässt man bei der allgemeinen Definition von Bewertungen nicht nur die reellen Zahlen zu.

**Definition 5.1-1:** *Es sei $W$ ein beliebiger Körper. (Meist wird es sich um $\mathsf{IF}_2$ oder $\mathsf{IR}$ handeln). $P = (E, K, v)$ sei ein Pseudograph. Eine **Eckenbewertung** von $P$ ist eine Abbildung $\omega$ von $E$ nach $W$,*

**Eckenbewertung**

$$\omega : E \to W.$$

*Entsprechend ist eine **Kantenbewertung** als Abbildung von $K$ nach $W$ definiert. Handelt es sich um einen Pseudodigraphen $P = (E, B, v)$, so ist eine **Bogenbewertung** eine Abbildung von $B$ nach $W$.*

**Kantenbewertung**

**Bogenbewertung**

## 5.2 Die algebraische Struktur von Bewertungen

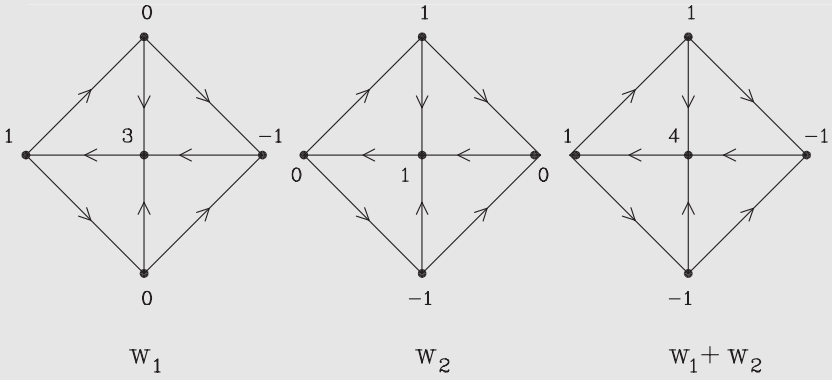
Die sehr allgemeine Definition einer Bewertung erlaubt eine große Breite von Anwendungen von bewerteten Pseudo(di)graphen. Oft sind jedoch spezielle Bewertungen von Interesse, die zu algebraischen Strukturen führen, welche zu den an früherer Stelle betrachteten Strukturen (z. B. $\mathbf{C}$ und $\mathbf{S}$) in enger Beziehung stehen. Von diesen Bewertungen und algebraischen Strukturen handelt der vorliegende Abschnitt. Der Einfachheit halber wird hier stets von einem Digraphen $\vec{G} = (E, B, v)$ ausgegangen, die Ergebnisse können leicht auf die anderen Fälle übertragen werden.

$\vec{G} = (E, B, v)$ sei gegeben mit $|E| = n$ und $|B| = m$, ebenso ein Körper $W$. Die Eckenbewertungen $\omega : E \to W$ bilden in der üblichen Weise einen Vektorraum über dem Körper $W$, der zu $W^n$ isomorph ist. Dabei ergibt sich z. B. die Summe $\omega_1 + \omega_2$ von Bewertungen als diejenige Bewertung, welche jeder Ecke die Summe der einzelnen Bewertungen zuordnet, formal: $(\omega_1 + \omega_2)(e) := \omega_1(e) + \omega_2(e)$ für $e \in E$. Den **Vektorraum der Eckenbewertungen** bezeichnen wir mit $V_e$. Analog hat man den (zu $W^m$ isomorphen) **Vektorraum**$V_b$ der **Bogenbewertungen**.

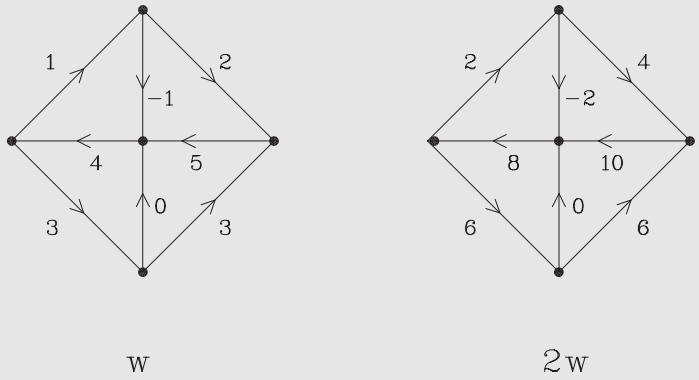
**Vektorraum der Eckenbewertungen**

**Vektorraum der Bogenbewertungen**

**Beispiel 5.2-1:**
Summe von Eckenbewertungen:




Multiplikation einer Bogenbewertung mit einem Skalar:




Die als nächstes eingeführten beiden Begriffe sind physikalisch motiviert. Es wird sich herausstellen, dass sie besonders gut algebraisch erfasst werden können.

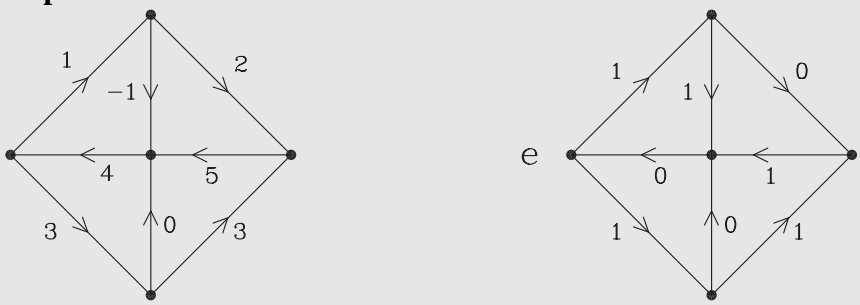
Es wird die folgende Notation verwendet: für einen Bogen $b$ bezeichnet $b^-$ die Anfangs- und $b^+$ die Endecke, d. h. es gilt $b = (b^-, b^+)$.

**Definition 5.2-1:** *Eine Bogenbewertung $f : B \to W$ heißt **Zirkulation**, wenn für jede Ecke $e \in E$ gilt* **Zirkulation**

$$\sum_{b^+=e} f(b) = \sum_{b^-=e} f(b).$$

*Stellt man sich $f(b)$ als den Umfang irgendwelcher entlang des Bogens $b$ transportierter oder fliessender Güter vor, so bedeutet obige Gleichung: alles, was in eine Ecke hineinfließt, kommt auch wieder heraus.*

**Beispiel 5.2-2:**




Im ersten Diagramm handelt es sich um eine Zirkulation, wobei die Bewertungen aus dem Körper $\mathbb{R}$ der reellen Zahlen genommen sind. fasst man auch im zweiten Bild die Bewertungen als Elemente von $\mathbb{R}$ auf, so hat man keine Zirkulation, weil z. B. in Ecke $e$ nichts hineinfließt, aber zwei Einheiten hinaus; nimmt man jedoch die Zahlen 0 und 1 als Elemente von $\mathbb{F}_2$ (als $W$), so handelt es sich hier um eine Zirkulation.

Die Zirkulationen bilden einen Untervektorraum von $V_b$. Um dies nachzuweisen, und auch um eine weitere Kategorie spezieller Bogenbewertungen einzuführen, werden als nächstes zwei lineare Abbildungen zwischen den Vektorräumen $V_e$ und $V_b$ untersucht.

Die Abbildung $\partial : V_b \to V_e$ ordnet jeder Bogenbewertung $f$ eine Eckenbewertung $\partial f$ zu, die definiert ist durch

$$(\partial f)(e) := \sum_{b \in B, b^+ = e} f(b) - \sum_{b \in B, b^- = e} f(b).$$

Sind die Mengen $E = \{e_1, \ldots, e_n\}$ und $B = \{b_1, \ldots, b_m\}$ durchnummeriert, und ist $I$ die zugehörige Ecken-Bogen-Inzidenzmatrix, so kann man auch schreiben

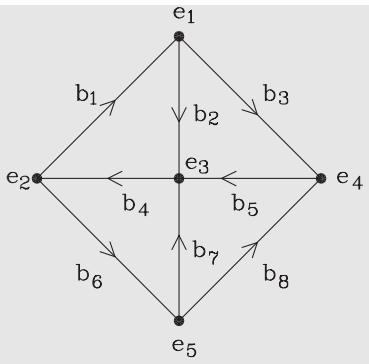
$$(\partial f)(e_r) = \sum_{s=1}^{m} i_{rs} \cdot f(b_s).$$

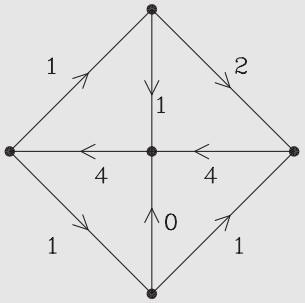
fasst man schließlich $f$ als Element von $W^m$ auf und entsprechend die Elemente von $V_e$ als solche von $W^n$, so lässt sich die Definition von $\partial$ auch kurz schreiben als

$$\partial f = I \cdot f.$$

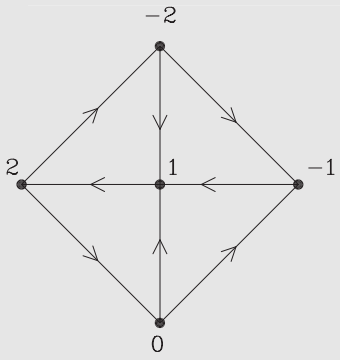
Aus dieser Beziehung wird klar, dass $\partial$ eine lineare Abbildung ist. Anschaulich bedeutet die Definition von $\partial$: einer Bogenbewertung wird als Eckenbewertung jeweils ”zufliessende minus abfliessende Menge” zugeordnet.

**Beispiel 5.2-3:**
Die Ecken und Bögen des betrachteten Digraphen seien wie im ersten Diagramm bezeichnet:




Im zweiten Bild ist eine Bogenbewertung $f$ dargestellt. Die Eckenbewertung $\partial f$ (zufliessende minus abfliesende Menge) ist dann die folgende:




Dies lässt sich mit der Inzidenzmatrix nachrechnen. Es ist

$$I = \begin{pmatrix} 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \end{pmatrix}$$

und $f = (1,1,2,4,4,1,0,1)$, und damit ergibt sich $\partial f = If = (-2,2,1,-1,0)$.

Offenbar gilt der

**Hilfssatz 5.2-1:** *$f \in V_b$ ist genau dann eine Zirkulation, wenn $\partial f = 0$ gilt.*

Die Menge aller Zirkulationen wird mit $\mathbf{C}(W)$ bezeichnet. Nach Hilfssatz 5.2-1 ist $\mathbf{C}(W)$ genau der Kern der linearen Abbildung $\partial$. Damit ergibt sich:

**Satz 5.2-1:** *$\mathbf{C}(W)$ ist ein Untervektorraum von $V_b$.*

Die Abbildung $\delta : V_e \to V_b$ ordnet jeder Eckenbewertung $g$ eine Bogenbewertung $\delta g$ zu, die definiert ist durch

$$(\delta g)(b) := g(b^+) - g(b^-).$$

Mithilfe der Inzidenzmatrix $I$ und der Isomorphien von $V_e$ und $W^n$ bzw. $V_b$ und $W^m$ kann man auch schreiben

$$\begin{aligned}(\delta g)(b_s) &= g(b_s^+) - g(b_s^-) \\ &= \sum_{r=1}^{n} i_{rs} \cdot g(e_r)\end{aligned}$$

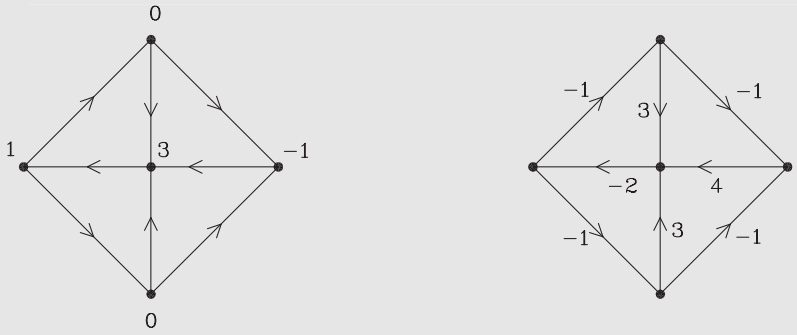
oder kurz

$$\delta g = I^t \cdot g.$$

Auch $\delta$ ist eine lineare Abbildung.
Eine anschauliche Vorstellung von $\delta$ kann man sich so machen: sieht man die gegebene Eckenbewertung als Potential an, so ergibt $\delta g$ für jeden Bogen die Potentialdifferenz der beiden Eckpunkte. Wegen dieser Interpretation nennt man eine Bogenbewertung $f$, für die es ein $g \in V_e$ mit $\delta g = f$ gibt, eine **Potentialdifferenz**.

**Potentialdifferenz**

**Beispiel 5.2-4:**
Die beiden Diagramme zeigen eine Eckenbewertung $g$ und die zugehörige Bogenbewertung $\delta g$:




$g$ $\delta g$

Man rechnet auch leicht nach, dass sich mit $g = (0, 1, 3, -1, 0)$ ergibt

$$\delta g = I^t g = (-1, 3, -1, -2, 4, -1, 3, -1).$$

**Definition 5.2-2:** *Die Menge aller Potentialdifferenzen wird mit* $\mathbf{S}(W)$ *bezeichnet.* $\mathbf{S}(W)$ *ist der Bildraum der linearen Abbildung* $\delta$ *und folglich ebenfalls ein Untervektorraum von* $V_b$*.*

Für die Elemente von $\mathbf{S}(W)$ fehlt noch eine direkte Charakterisierung. Dazu gehen wir, wie bei der Definition der Kreis-Bogen-Matrix, von einer gegebenen Orientierung für jeden Kreis von $\vec{G}$ aus.

**Satz 5.2-2:** *$f \in V_b$ ist genau dann Element von $\mathbf{S}(W)$, wenn für jeden Kreis $C$ von $\vec{G}$ gilt*

$$\sum_{b \in C} f(b) = 0.$$

*(Dabei werden in der Summe die Bögen, die entgegen der Orientierung gerichtet sind, negativ gezählt.)*

**Beweis:** Es sei $\mathbf{S}(W)$ und $g \in V_e$ mit $\delta g = f$, ferner sei $C$ ein Kreis mit den Ecken $x_1, \ldots, x_t$ und Bögen $y_1, \ldots, y_t$, wobei $y_1 x_1$ mit $x_2$, $y_2 x_2$ mit $x_3$ verbindet usw. und schließlich $y_t x_t$ mit $x_1$ verbindet. Die Numerierung stimme mit der Orientierung überein. Es gilt dann

$$\begin{aligned}\sum_{b \in C} f(b) &= f(y_1) + \ldots + f(y_t) \\ &= (g(x_2) - g(x_1)) + (g(x_3) - g(x_2)) + \ldots + (g(x_t) - g(x_{t-1})) \\ &\quad +(g(x_1) - g(x_t)) \\ &= 0.\end{aligned}$$

Umgekehrt sei für ein $f \in V_b$ die Beziehung $\sum_{b \in C} f(b) = 0$ für jeden Kreis $C$ erfüllt. Es muss ein $g \in V_e$ mit $\delta g = f$ konstruiert werden. Dazu wird zunächst einer beliebigen Ecke $e_0 \in E$ als $g(e_0)$ ein beliebiger Wert $w_0 \in W$ zugeordnet. (Ist $\vec{G}$ nicht zusammenhängend, so müssen in jeder Komponente solche Wahlen getroffen werden.) Nun hat man, da ein $g$ mit $\delta g = f$ herauskommen soll, nur eine Wahl, $g(b^-)$ zu erklären, wenn $f(b)$ und $g(b^+)$ gegeben sind, nämlich

$$g(b^-) := g(b^+) - f(b).$$

Entsprechend muss, wenn $g(b^-)$ schon definiert ist,

$$g(b^+) := g(b^-) + f(b)$$

gesetzt werden. Auf diese Weise kann induktiv für ganz $E$ die Eckenbewertung $g$ konstruiert werden, wobei die gegebene Voraussetzung für $f$ hinsichtlich Kreisen dafür sorgt, dass für Ecken, denen man sich aus verschiedenen Richtungen nähert, der gleiche Wert herauskommt.

Wir haben nun die Untervektorräume $\mathbf{C}(W)$ (Zirkulationen) und $\mathbf{S}(W)$ (Potentialdifferenzen) des Vektorraums $V_b$ der Bogenbewertungen durch Bedingungen beschrieben, die den **Kirchhoffschen Gesetzen** der Theorie elektrischer Netzwerke

**Kirchhoffschen Gesetzen**

entsprechen. Danach sind Zirkulationen Bogenbewertungen, die dem 1. Kirchhoffschen Gesetz (für elektrische Ströme) folgen, und Potentialdifferenzen solche Bogenbewertungen, auf die das 2. Kirchhoffsche Gesetz (für elektrische Spannungen) zutrifft. Diese Zusammenhänge werden bei der Analyse elektrischer Netzwerke angewandt.

$\mathbf{S}(W)$ ist der von den Zeilenvektoren von $I$ aufgespannte Untervektorraum von $V_b$ (bzw. $W^m$), während $\mathbf{C}(W)$ aus den Elementen von $W^m$ besteht, die zu jeder Zeile von $I$ (und damit zu jedem Element von $\mathbf{S}(W)$) orthogonal sind.
Damit kann man insgesamt den folgenden Satz zeigen.

**Satz 5.2-3:** *$f \in V_b$ ist genau dann eine Zirkulation (Potentialdifferenz), wenn $f$ zu allen Potentialdifferenzen (Zirkulationen) orthogonal ist.*

Es wurde bereits angekündigt, dass die Vektorräume $\mathbf{C}(W)$ und $\mathbf{S}(W)$ zu den in Abschnitt 1.3 behandelten $\mathbf{C}^*$ und $\mathbf{S}^*$ in enger Beziehung stehen. Um dies näher zu beleuchten, setzen wir zunächst einmal $W = \mathsf{IF}_2$ voraus.

Im Abschnitt 1.3 wurde $\mathbf{C}^*$ (für einen Multigraphen $M$) als $\mathsf{IF}_2$-Vektorraum der Kantenmengen eingeführt, die sich als Vereinigung kantendisjunkter Kreise darstellen lassen. In Hilfssatz 1.3-1 wurde die alternative Charakterisierung als Menge derjenigen Teilgraphen gegeben, in denen jede Ecke einen geraden Grad (von mindestens zwei) besitzt.

Wir gehen nun wieder von dem Digraphen $\vec{G} = (E, B, v)$ aus, $\mathbf{C}^*$ sei der zu dem ungerichteten Graphen $G$ gehörige $\mathsf{IF}_2$-Vektorraum. Andererseits sei $f : B \to \mathsf{IF}_2$ eine Bogenbewertung, $f^{-1}(1)$ die Menge der mit "1" bewerteten Kanten. Wir zeigen nun:

**Hilfssatz 5.2-2:** *Es ist genau dann $f \in \mathbf{C}(\mathsf{IF}_2)$, wenn $f^{-1}(1) \in \mathbf{C}^*$ gilt.*

**Beweis:** Es sei $f \in \mathbf{C}(\mathsf{IF}_2)$. Um $f^{-1}(1) \in \mathbf{C}^*$ nachzuweisen, überlegt man sich, dass in dem durch die Kantenmenge $f^{-1}(1)$ bestimmten Teilgraphen von $G$ jede Ecke einen geraden Grad hat. Dies folgt jedoch sofort daraus, dass aufgrund der Bedingung an eine $\mathsf{IF}_2$-Zirkulation auf $\vec{G}$ bei jeder Ecke Innengrad und Außengrad (bezogen auf den Teilgraphen) entweder beide ungerade oder beide gerade sind.
Umgekehrt, wenn $f^{-1}(1) \in \mathbf{C}^*$ vorausgesetzt wird, kann so geschlossen werden: in dem Teilgraphen $f^{-1}(1)$ hat jede Ecke einen geraden Grad, also müssen Innen- und Außengrad gleichzeitig gerade bzw. ungerade sein: letzteres bedeutet genau, dass die in $\mathsf{IF}_2$ summierten "zufliessenden bzw. abfliessenden Ströme" gleichzeitig 0 bzw. 1 sind. Folglich muss $f$ eine Zirkulation sein.

Es ergibt sich nun der folgende

**Satz 5.2-4:** *$\mathbf{C}(\mathsf{IF}_2)$ ist isomorph zu $\mathbf{C}^*$.*

**Beweis:** Den Beweis dieses Satzes wollen wir nicht in jeder Einzelheit formal nachvollziehen. Inhaltlich ist der Satz aufgrund der obigen Bemerkungen klar: die $\mathsf{IF}_2$-Zirkulation entsprechen eineindeutig den Kantenmengen, die Vereinigung disjunkter Kreise sind; die Isomorphie ergibt sich dann wei-

ter daraus, dass die Vektorraumaddition in $\mathbf{C}(\mathsf{IF}_2)$ und $\mathbf{C}^*$ völlig analog ausgeführt wird.

Der entsprechende Satz für Schnitte lautet:

**Satz 5.2-5:** $\mathbf{S}(\mathsf{IF}_2)$ *ist isomorph zu* $\mathbf{S}^*$.

Der Schlüssel zum Beweis diese Satzes, den wir auch hier nicht formal führen wollen, ist der folgende

**Hilfssatz 5.2-3:** *Für eine Bogenbewertung* $f : B \to \mathsf{IF}_2$ *gilt genau dann* $f \in \mathbf{S}(\mathsf{IF}_2)$, *wenn* $f^{-1}(1) \in \mathbf{S}^*$ *ist.*

**Beweis:** Es sei $f \in \mathbf{S}(\mathsf{IF}_2)$ und $g$ eine Eckenbewertung $E \to \mathsf{IF}_2$ mit $\delta g = f$. Wir setzen $E' = g^{-1}(0)$ und $E'' = g^{-1}(1)$, die Kantenmenge $S$ sei der zur disjunkten Einteilung von $E$ in $E'$ und $E''$ gehörende Schnitt. Offenbar bewertet die aus $g$ abgeleitete Potentialdifferenz $f$ genau die Kanten (bzw. Bögen) aus $S$ mit 1, d. h. $f^{-1}(1) = S \in \mathbf{S}^*$.
Umgekehrt sei nun $f^{-1}(1) \in \mathbf{S}^*$, d. h. die mit 1 bewerteten Bögen bilden einen Schnitt. Es folgt, dass in jedem Kreis von $\vec{G}$ (bzw. $G$) eine gerade Anzahl von Bögen die Bewertung 1 trägt, damit folgt $f \in \mathbf{S}(\mathsf{IF}_2)$.

Aufgrund der soeben abgeleiteten Isomorphien hat man mit den Ergebnissen über die Vektorräume $\mathbf{C}^*$ und $\mathbf{S}^*$ aus Abschnitt 1.3 nun auch Aussagen über Dimensionen und Basen von $\mathbf{C}(\mathsf{IF}_2)$ und $\mathbf{S}(\mathsf{IF}_2)$. Diese Aussagen gelten auch für $\mathbf{C}(W)$ und $\mathbf{S}(W)$; jedoch lässt sich im allgemeinen Fall natürlich nicht mehr die Verbindung zu $\mathbf{C}^*$ und $\mathbf{S}^*$ ziehen, und die Aussagen bedürfen erneuter Beweise. Wir wollen diese Beweise hier nicht führen. Die Aussagen sollen im folgenden aufgeführt werden.

Zur Vorbereitung müssen einige Bezeichnungen eingeführt werden. Es sei $A$ ein spannender Wald im gegebenen Digraphen $\vec{G}$, $W$ sei irgendein Körper. Die $\mu(G)$ vielen fundamentalen Kreise von $\vec{G}$ seien mit einer Orientierung versehen, ebenso die $\rho(G)$ vielen fundamentalen Schnitte. Für einen fundamentalen Kreis $C$ sei $f_c$ die Bogenbewertung mit

$$f_c(b) = \begin{cases} 1, & \text{falls } b \text{ zu } C \text{ gehört und die Richtung von } b \text{ mit der Orientierung von } C \text{ übereinstimmt} \\ -1, & \text{falls } b \text{ zu } C \text{ gehört und Richtung und Orientierung nicht übereinstimmen} \\ 0, & \text{sonst.} \end{cases}$$

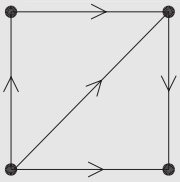
Für einen fundamentalen Schnitt $S$ sei $f_S$ die in ähnlichem Sinne auf den Schnitt $S$ bezogene Bogenbewertung.

**Satz 5.2-6:** *Mit den obigen Bezeichnungen gilt:*

1. *Die* $\mu(G)$ *vielen Bogenbewertungen der Form* $f_C$ *bilden eine Basis des* $W$*-Vektorraums* $\mathbf{C}(W)$ *aller Zirkulationen.*

2. *Die* $\rho$ *vielen Bogenbewertungen der Form* $f_S$ *bilden eine Basis des* $W$*-Vektorraums* $\mathbf{S}(W)$ *aller Potentialdifferenzen.*

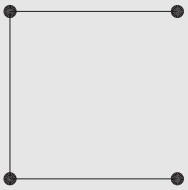
**Beispiel 5.2-5:**
Der folgende Digraph $\vec{G}$ ist eine gerichtete Version des in Beispiel 1.3-4 analysierten Graphen:



$\vec{\mathrm{G}}$

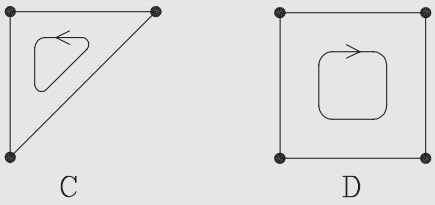
Es ist $\mu(G) = 2$ und $\rho(G) = 3$.

Wir wählen auch wieder das folgende Gerüst:

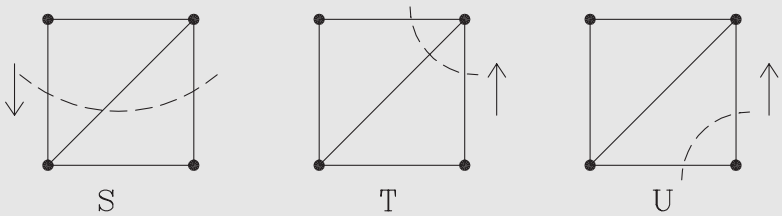


A

Dies führt zu den fundamentalen Kreisen $C$, $D$ und den fundamentalen Schnitten $S$, $T$, $U$, die jeweils mit einer Orientierung versehen werden:

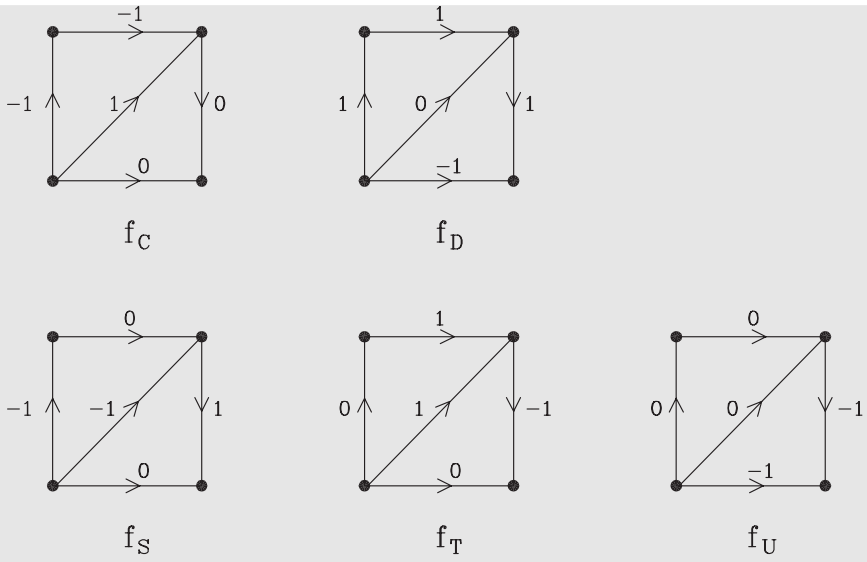


C D

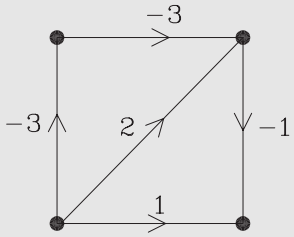


S T U

Im Falle $W = \mathsf{IR}$ bilden die im nächsten Bild dargestellten Bogenbewertungen Basen von $\mathbf{C}(\mathsf{IR})$ bzw. $\mathbf{S}(\mathsf{IR})$:




Jede Zirkulation ist nun als Linearkombination von $f_C$ und $f_D$ erhältlich. Das folgende Diagramm z. B. stellt eine Zirkulation $f$ dar:




Es ist $f = 2f_C - f_D$.

**Selbsttestaufgabe 5.2-1:**

*Es sei wieder der Multidigraph $\vec{M}$ aus Selbsttestaufgabe Selbsttestaufgabe 4.2-1 gegeben. Geben Sie eine Basis des Vektorraums $\mathbf{C}(\mathbb{R})$ der reellen Zirkulationen an.*

# 6 Kürsteste Wege und minimale Gerüste

## 6.1 Kürzeste Wege

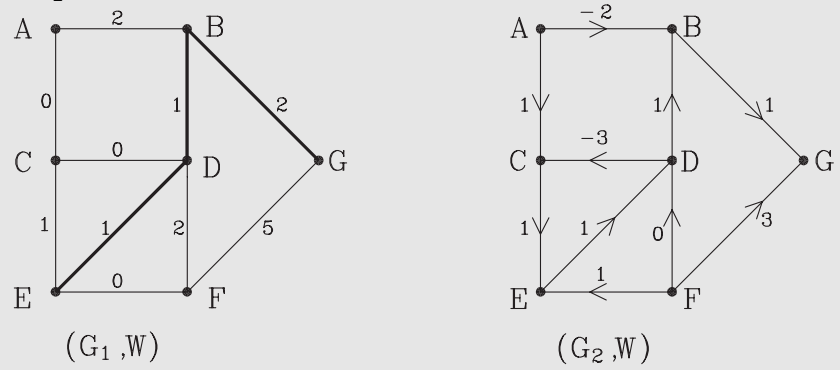
In vielen Anwendungen von Graphen fragt man nach besonders günstigen Wegen, oft handelt es sich dabei um möglichst kurze Wege. Die alltäglichsten Beispiele sind Straßennetze oder die Bus- und Bahnlinien eines Verkehrsverbundes, wo nach der kürzesten Möglichkeit gesucht wird, von einem Punkt zu einem anderen zu gelangen. Stehen auf einigen Teilstrecken verschiedene Verkehrsmittel zur Verfügung, so mag man auch nach der insgesamt billigsten Variante fragen. In einem Fernmeldenetz, wo die einzelnen Leitungen verschiedene Ausfallwahrscheinlichkeiten haben, ist die verlässlichste Route zwischen zwei Punkten von Interesse.

Zur Vereinfachung der Sprechweise führen wir den Begriff eines Netzes ein.

**Netz**
**Länge einer Kante**
**Länge einer Kantenfolge**
**kürzester Weg**

Gegeben sei ein Graph oder Digraph $G = (E, K)$[2], mit einer Bewertung $w : K \to \mathbb{R}$. Das Paar$(G, w)$ nennen wir ein **Netz**, die Zahl $w(k)$ heißt die **Länge der Kante** $k$. Unter der **Länge der Kantenfolge** $k_1, k_2, \ldots k_r$ wird die Zahl $w(k_1) + w(k_2) + \ldots + w(k_r)$ verstanden. Ist $G$ ein Digraph, so ist dabei selbstverständlich vorausgesetzt, dass es sich um eine gerichtete Kantenfolge handelt. Eine Kantenfolge möglichst kurzer Länge von einer Ecke $a$ zu einer Ecke $b$ ist, sofern eine solche existiert, immer ein Weg, man nennt ihn einen **kürzesten Weg** von $a$ nach $b$.

**Beispiel 6.1-1:**




In $(G_1, w)$ bilden die herausgehobenen Kanten einen kürzesten Weg (der Länge $4$) von $E$ nach $G$; Nutzung der Kanten von $E$ nach $C$und von dort nach $D$ ergäbe ebenfalls einen kürzesten Weg.
In $(G_2, w)$ gibt es beliebig kurze Kantenfolgen von $E$ nach $G$, die man erhält, indem man den negativen Kreis $E \to D \to C \to E$ immer wieder durch-

2 Ab hier werden der Einfachheit halber Graphen und Digraphen nicht mehr unterschiedlich bezeichnet, in $G = (E, K)$ können also die Elemente von $K$ ungerichtete Kanten wie auch gerichtete Kanten (Bögen) sein.

läuft. Von $E$ nach $F$ z. B. gibt es überhaupt keine Kantenfolge (somit auch keine kürzeste). Von $B$ nach $G$ kommt man nur über die direkte Kante, diese bildet folglich einen kürzesten Weg.

Wie oben an dem zweiten Beispiel zu sehen ist, gibt es bei der Frage der Existenz kürzester Kantenfolgen zwei mögliche Schwierigkeiten: es kann

- keine Kantenfolge von $a$ nach $b$ geben,
- beliebig kurze Kantenfolgen von $a$ nach $b$ geben.

Das zweite Problem hat seinen Ursprung in der Existenz von Kreisen negativer Länge. Diese Verkomplizierung ließe sich umgehen, wenn man nur positive Kantenbewertungen zuließe, jedoch wäre dies für die spätere allgemeine mathematische Behandlung von Netzen nicht sinnvoll und auch eine für mögliche Anwendungen zu starke Einschränkung: man mag z. B. an ein Transportnetz denken, wo positive Kantenbewertungen Profit und negative entstehende Kosten bedeuten.

**Definition 6.1-1:**

In enger Beziehung zu kürzesten Kantenfolgen bzw. Wegen steht der Begriff des Abstands, den wir für den Fall nicht-bewerteter Pseudographen schon in Kapitel 1 kennengelernt haben (s. Definition 1.2-7). Für Ecken $a, b$ eines Netzes $(G, w)$ ist der **Abstand** $| a, b |$ das Minimum der Längen aller Kantenzüge von $a$ nach $b$. Existiert kein solcher Kantenzug, so wird $| a, b | = \infty$ gesetzt. Gibt es negative Kreise in $(G, w)$, so ist $| a, b |$ nicht immer definiert, es sei denn, man beschränkt sich auf Wege von $a$ nach $b$.

**Abstand**

Ist $G$ als unbewerteter Graph oder Digraph gegeben, so kann man ein Netz bilden, indem

$$w(k) = 1$$

für jede Kante $k$ gesetzt wird. Damit wird obige Definition des Abstandes eine Erweiterung der in Kapitel 1 gegebenen. Auch alle weiteren Begriffe und Resultate sind auf unbewertete Graphen übertragbar, wenn diese als Netze mit konstanter Kantenbewertung aufgefasst werden.

Im weiteren Verlauf dieses Abschnitts geht es vorwiegend darum, Algorithmen zur Bestimmung von Abständen und kürzesten Wegen in Netzen kennenzulernen. Folgende Überlegung zeigt, dass es notwendig ist, nach günstigen Algorithmen Ausschau zu halten: In dem vollständigen Graphen $K_n$ mit $n$ Ecken gibt es zwischen zwei Ecken allein schon $(n-2)!$ viele Wege, die durch alle anderen Ecken laufen. Der "Algorithmus", einfach **alle** Wege zu bestimmen und deren Längen zu vergleichen, hat also eine exponentielle Komplexität.

Wegen seiner grundsätzlichen Bedeutung wollen wir zunächst einen Algorithmus behandeln, der auf unbewertete Graphen anzuwenden ist: **Algorithmus "Breadth first search"**

**Algorithmus "Breadth first search"**

Gegeben sei ein zusammenhängender Graph $G = (E, K)$, $N(e) \subseteq E$ sei zu $e \in E$

die Menge der Nachbar- bzw. Nachfolgerecken, $s \in E$ fest gewählt. Es wird jeder Ecke eine Markierung (Bewertung) $d(e)$ zugeordnet.
Der Algorithmus verwendet als Datenstruktur eine Liste $L$, von der vorn Elemente entfernt und hinten Elemente angefügt werden.

**Algorithmus**: Breadth first search

```
begin
   Setze L := {s}, d(s) := 0;
   while L ≠ ∅ do
   begin
   entferne den ersten Punkt v von L aus L;
   for w ∈ N(v) do
         begin
         if d(w) ist noch undefiniert
         then setze d(w) := d(v) + 1,
            füge w ans Ende von L an;
         end
   end
end
```

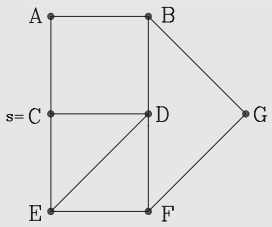
Wir werden gleich beweisen, dass nach dem Durchlauf des Algorithmus "breadth first search" $d(e)$ für jede Ecke $e$ den Abstand $\mid s, e \mid$ zum Punkt $s$ angibt. Jedoch ist an diesem Algorithmus noch folgendes interessant: die Ecken des Graphen werden - von $s$ ausgehend- in der Reihenfolge betrachtet und mit einem $d$-Wert markiert, in der sie als Nachbarn bereits markierter Ecken aufgetreten sind; dafür sorgt die nach dem "first in - first out" funktionierende Liste. Der Graph wird sozusagen "in die Breite" untersucht, d. h. es werden zunächst alle Nachbarn von $s$ markiert, bevor deren Nachbarn betrachtet werden usw. Diese Vorgehensweise ist als allgemeines Suchprinzip in Graphen auch in anderen Algorithmen enthalten. Das Gegenstück dazu, "Depth first search", wird im nächsten Abschnittbehandelt.

**Satz 6.1-1:** *Nach dem Durchlaufen des Algorithmus "Breadth first search" gilt $d(e) = |s, e|$ für jede Ecke $e$. Der Algorithmus hat die Komplexität $O(|K|)$.*

**Beweis:** Für eine beliebige Ecke $e$ gilt offenbar $|s, e| \leq d(e)$, da $e$ im Verlauf des Algorithmus bis zu seiner Markierung auf einem Weg der Länge $d(e)$ von $s$ aus erreicht wurde. Mit Induktion über $|s, e|$ wird nun gezeigt, dass sogar die Gleichheit $|s, e| = d(e)$ für alle $e$ gilt. Für $|s, e| = 0$ ist das klar, da in diesem Fall $s = e$ sein muss. Es sei $|s, e| = n + 1$, und $s, v_1, \ldots, v_n, e$ sei ein kürzester Weg von $s$ nach $e$. Es ist dann $s, v_1, \ldots, v_n$ ein kürzester Weg von $s$ nach $v_n$, und nach Induktionsannahme gilt folglich $|s, v_n| = n = d(v_n)$. Wegen $d(v_n) < n + 1 = |s, e| \leq d(e)$ ist $d(v_n) < d(e)$, d. h. im Algorithmus wurde $v_n$ vor $e$ erreicht und in der while-Schleife betrachtet. Da $G$ die Kante von $v_n$ nach $e$ enthält, wird andererseits $e$ spätestens bei der Untersuchung der Nachbarn von $v_n$ erreicht, weshalb $d(e) \leq n+1$ und somit $d(e) = n+1$ folgt. Die Aussage über die Komplexität ergibt sich daraus, dass jede Kante zweimal (im gerichteten Fall sogar nur einmal) betrachtet wird

und die ausgeführten Operationen nur aus dem Markieren und dem Pflegen der Liste $L$ bestehen.

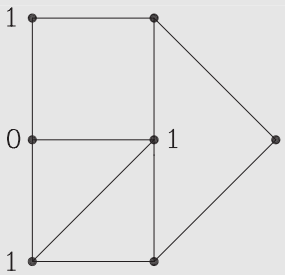
**Beispiel 6.1-2:**
In dem Graphen $G$ sei $C$ als die feste Ecke $s$ gewählt:



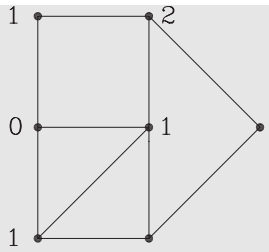

Beim Durchlauf des Algorithmus "Breath first search" ist zunächst $L = \{C\}$ und $C$ mit $0$ markiert. In der folgenden Darstellung sind die Durchläufe der while-Schleife durchnumeriert:

1. $C$ wird aus $L$ entfernt,
   $A$ wird mit $1$ markiert, $L = \{A\}$,
   $D$ wird mit $1$ markiert, $L = \{A, D\}$,
   $E$ wird mit $1$ markiert, $L = \{A, D, E\}$;

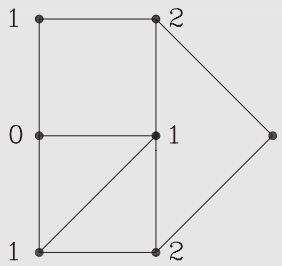
   Zwischenresultat:




2. $A$ wird aus $L$ entfernt,
   $B$ wird mit $1$ markiert, $L = \{D, E, B\}$,
   $C$ ist bereits markiert; Zwischenresultat:

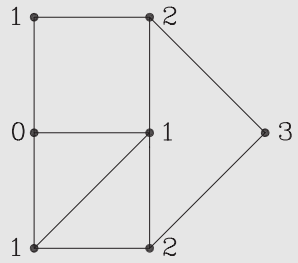



3. $D$ wird aus $L$ entfernt,
   $B$, $C$, $E$ sind bereits markiert,
   $F$ wird mit 2 markiert, $L = \{E, B, F\}$;
   Zwischenresultat:




4. $E$ wird aus $L$ entfernt,
   keine neue Ecke wird markiert;

5. $B$ wird aus $L$ entfernt,
   nur $G$ wird neu mit 3 markiert, $L = \{F, G\}$;
   Zwischenresultat:




   6. und 7. dienen nur noch dem Abbau der Liste $L$, das letzte Zwischenresultat ist auch das Endergebnis.

Wir betrachten nun das Problem der Bestimmung kürzester Wege in einem allgemeinen Netz $(G, w)$. Es wird vorausgesetzt, dass $G$ keine Kreise negativer Länge enthält, ferner wird $G$ als gerichtet angenommen.

**Definition 6.1-2:** *Zur leichteren Formulierung der nun folgenden Aussage sei die Eckenmenge von $G$ die Menge $\{1, 2, \ldots n\}$. Mit Hilfe der gegebenen Bewertung $w$ definieren wir eine Matrix $(w_{ij})$ durch*

$$w_{ij} := \begin{cases} w(ij), \textit{falls } G \textit{ die Kante } ij \textit{ enthält} \\ \infty, \textit{sonst.} \end{cases}$$

*Für jede Ecke $i$ bezeichne $u_i$ den Abstand $|1, i|$. Um unnötige Verkomplizierungen zu vermeiden, sei $u_i < \infty$ vorausgesetzt, d. h. jede Ecke $i$ sei von $1$ aus erreichbar.*

**Satz 6.1-2:** *Unter den obenstehenden Voraussetzungen gelten die **Bellmanschen Gleichungen*** **Bellmansche Gleichungen**

$$u_1 = 0\ ,\ u_i = min\ \{u_k + w_{ki} | k \in \{1, \ldots, n\}, k \neq i\}\ \ (i = 2, \ldots, n).$$

**Beweis:** Jeder kürzeste Weg von $1$ nach $i$ muss eine letzte Kante $ki$ enthalten, und durch Weglassen der Kante $ki$ muss sich ein kürzester Weg von $1$ nach $k$ ergeben. Daraus folgen die Gleichungen unmittelbar.

Die Bellmanschen Gleichungen können als notwendige Eigenschaften angesehen werden, die den Zahlen $u_i = |1, i|$ eigen sind. Es zeigt sich, dass diese Eigenschaften sogar charakteristisch für die Abstände $|1, i|$ sind:

**Satz 6.1-3:** *Enthält $G$ nur Kreise positiver Länge, so haben die Gleichungen*

$$x_1 = 0\ ,\ x_i = min\ \ \{x_k + w_{ki} | k \in \{1, \ldots, n\}, k \neq i\}\ \ (i = 2, \ldots, n)$$

*als eindeutige Lösung $x_i = u_i$ für $i = 1, \ldots, n$.*

Zur Erleichterung des Beweises dieses Satzes stellen wir die folgende Konstruktion voran.
Es seien (unter den Voraussetzungen des Satzes) $v_1, \ldots, v_n$ irgendeine Lösung der Gleichungen, $j$ sei eine beliebige Ecke von $G$. Da $v_j = min\{v_k + w_{kj} | k \neq j\}$ gilt, können wir ein $i$ wählen, für das dieses Minimum angenommen wird, für das also $v_j = v_i + w_{ij}$ ist. Von $i$ ausgehend kommen wir zu einem $h$ mit $v_i = v_h + w_{hi}$ usw., bis der Punkt $1$ erreicht wird. Die Ecke $1$ muss erreicht werden, da man andernfalls einen Kreis der Länge Null konstruieren würde. Auf diese Weise kommt man somit zu einem Weg von $1$ nach $j$, so dass der Teil von $1$ zu einem Punkt $g$ auf diesem Weg stets die Länge $v_g$ hat. Entsprechend wird nun für allen noch nicht vorkommenden Ecken fortgefahren, wobei ein Weg nur solange rückwärts konstruiert wird, bis man eine auf einem bereits konstruierten Weg liegende Ecke erreicht. Resultat ist dann ein gerichteter (spannender) Baum mit Wurzel $1$.
Ein weiteres Ergebnis dieser Konstruktion ist, dass für jede Ecke $jv_j \geq u_j$ gelten muss.
Wendet man die Konstruktion auf die Lösung $u_1, \ldots, u_n$ der Bellmanschen Gleichungen an, so nennt man das Ergebnis auch einen **Baum kürzester Wege** (shortest path tree). Sobald Satz 6.1-3 bewiesen ist, wissen wir natürlich, dass oben (ausgehend von $v_1, \ldots, v_n$) ein Baum kürzester Wege konstruiert wurde. **Baum kürzester Wege**

**Beweis:** (Satz 6.1-3): $v_1, \ldots, v_n$ seien eine Lösung, und $B$ sei ein - wie soeben in der Konstruktion beschrieben - konstruierter Baum kürzester Wege (gerichteter spannender Baum mit Wurzel 1). In diesem Baum gibt es von der Ecke 1 zu jeder anderen Ecke jeweils genau einen Weg. Wir zeigen nun, dass für jede Ecke $j$ gilt $v_j = u_j$, wobei Induktion über die Anzahlder Kanten in dem eindeutigen Weg in $B$ von 1 nach $j$ verwendet wird. Für $j = 1$ ist nichts zu zeigen. Nun sei $j \neq 1$ gegeben, wobei nach Induktionsvoraussetzung für jede Ecke $k$, die von 1 aus in $B$ mit weniger Kanten als $j$ erreichbar ist, $v_k = u_k$ gelte. Wir führen nun die Annahme $v_j > u_j$ zum Widerspruch. Es sei $ij$ die Kante mit Endpunkt $j$ in $B$. Wir wissen, dass $v_i = u_i$ gilt. Man hat nun $v_j > u_j = u_i + w_{ij} = v_i + w_{ij}$, und die erhaltene Beziehung

$$v_j > v_i + w_{ij}$$

steht zu der Gleichung

$$v_j = \ min\{v_k + w_{kj} | k \neq j\}$$

im Widerspruch.
Es muss also gelten $v_j = u_j$.

Das eigentliche Ziel ist es selbstverständlich, die kürzesten Abstände bzw. Wege in einem Graphen effektiv zu bestimmen. Dazu sind die Bellmanschen Gleichungen keine unmittelbare Hilfe bis auf den folgenden Fall: Es sei $(G, w)$ ein Netz mit azyklischem $G$. In Kapitel 4 wurde gezeigt, dass in $O(|K|)$ vielen Schritten eine topologische Anordnung hergestellt werden kann, also eine Numerierung der Ecken mit den Zahlen $1, \ldots, n$ derart, dass aus der Existenz einer Kante $ij$ stets $i < j$ folgt. Mit dieser Numerierung reduzieren sich die Bellmanschen Gleichungen zu

$$u_1 = 0\ , \qquad u_i = \ min\{u_k + w_{ki} | k = 1, \ldots, i-1\} \ \ (i = 2, \ldots, n)\ ,$$

und diese Gleichungen können in $O(|K|)$ Schritten rekursiv gelöst werden.

Als nächstes werden nun Netze $(G, w)$ betrachtet, in denen alle Kantenlängen nichtnegativ sind. ($G$ wird nicht als azyklisch vorausgesetzt.) Die Bellman-Gleichungen können hier mit dem Algorithmus von Dijkstra gelöst werden, der das bekannteste Verfahren zur Bestimmung kürzester Wege liefert:

**Algorithmus von Dijkstra**

**Algorithmus von Dijkstra** Gegeben sei ein Netz $(G, w)$, für das $w(k) \geq 0$ für jede (gerichtete) Kante $k$ gilt, ferner sei $s$ eine Ecke in $G$. Es werden die Abstände von $s$ zu allen anderen Ecken bestimmt. (Dabei muss nicht jede Ecke von $s$ aus erreichbar sein.)

---

**Algorithmus**: Algorithmus von Dijkstra

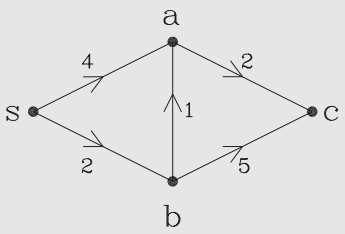
---

```
begin
   Setze d(s) := 0, T := E;
   for e ∈ E \ { s } do setze d(e) := ∞;
   while T ≠ ∅ do
   begin
      finde ein f ∈ T, für das d(f) minimal ist;
```

```
      setze T := T \ { f };
      for e ∈ T do
      setze d(e) :=    min { d(e), d(f) + w_fe };
   end
end
```

Die Vorgehensweise des Algorithmus von Dijkstra lässt sich in Worten so beschreiben: Zu Anfang besteht die Menge $T$ aus allen Ecken, wo bei der Ecke $s$ als vorläufiger Abstand $0$ und allen anderen $\infty$ zugeordnet wird. Im weiteren wird dann die Menge $T$ sukzessiv abgebaut: es wird jeweils die Ecke $e$ mit dem kürzesten vorläufigen Abstand aus $T$ entfernt (im ersten Schritt also $e = s$), und dieser Abstand ist auch ihr gesuchter kürzester Abstand; die vorläufigen Abstände der in $T$ verbliebenen Ecken werden aktualisiert, in dem geprüft wird, ob das Laufen über die soeben aus $T$ entfernte Ecke $e$ zu einem kürzeren Weg zu der betreffenden Ecke führt.

**Beispiel 6.1-3:**
Der Algorithmus von Dijkstra wird auf das folgende Netzwerk angewendet:




Zunächst ist $T = \{s, a, b, c\}$, $d(s) = 0$, $d(a) = d(b) = d(c) = \infty$.

1. Schritt: Für $s \in T$ ist $d$ minimal, also $T = \{a, b, c\}$,
   $d(s) = 0$; $d(a) = 4$, $d(b) = 2$, $d(c) = \infty$.
2. Schritt: Für $b \in T$ ist $d$ minimal, also $T = \{a, c\}$,
   $d(s) = 0$, $d(b) = 2$; $d(a) = 3$, $d(c) = 7$.
3. Schritt: Für $a \in T$ ist $d$ minimal, also $T = \{c\}$,
   $d(s) = 0$, $d(b) = 2$, $d(a) = 3$; $d(c) = 5$.
4. Schritt: Für $c$ wird $d(c) = 5$ endgültig festgestellt.

Es muss nun die Korrektheit des Algorithmus von Dijkstra gezeigt werden

**Satz 6.1-4:** *Der Algorithmus von Dijkstra berechnet die Abstände von $s$ zu allen anderen Ecken in $(G, w)$: am Ende des Algorithmus gilt $d(e) = |s, e|$ für jede Ecke $e \in E$. Der Algorithmus hat die Komplexität $O(|E|^2)$.*

**Beweis:** Zunächst überlegt man sich, dass nach dem Ablauf des Algorithmus nur dann $d(e) = \infty$ ist, wenn $e$ von $s$ aus nicht erreicht werden kann. Für diesen Fall ist $d(e) = |s, e|$ richtig.

Für $e$ mit $d(e) \neq \infty$ gilt $|s, e| \leq d(e)$, da es aufgrund der Definition von $d(e)$ im Algorithmus einen gerichteten Weg der Länge $d(e)$ von $s$ nach $e$ gibt. Die umgekehrte Ungleichung $d(e) \leq |s, e|$ wird nun bewiesen durch Induktion über die Anzahl der Elemente, die bereits aus der Menge $T$ entfernt wurden.

Der Induktionsanfang ist leicht zu machen, denn als erster Punkt wird $s$ aus $T$ entfernt, und es gilt $d(s) = 0 = |s, s|$.

Nun gelte $d(e) \leq |s, e|$ bereits für alle Ecken, die vor der Ecke $f$ aus $T$ entfernt wurden. Wir können $d(f) < \infty$ annehmen, denn sonst gilt für $f$ und alle noch in $T$ verbliebenen Ecken das obige Argument.

Es sei nun $s = e_1, e_2, \ldots, e_n = f$ ein kürzester Weg in $(G, w)$ von $s$ nach $f$. Es gilt also

$$|s, e| = w(e_1e_2) + \ldots + w(e_{n-1}e_n).$$

Über die $e_i (2 \leq i \leq n-1)$ wissen wir zunächst nicht, ob sie noch in $T$ sind oder nicht. Es sei $j$ der maximale Index der $e_i$, für den $e_i$ vor $f$ aus $T$ entfernt wurde. Da $s = e_1, e_2, \ldots, e_j$ ein kürzester Weg von $s$ nach $e_j$ ist, hat man aufgrund der Induktionsvoraussetzung

$$d(e_j) = |s, e_j| = w(e_1e_2) + \ldots + w(e_{j-1}e_j).$$

Es gilt auf jeden Fall

$$d(e_{j+1}) \leq d(e_j) + w(e_je_{j+1})$$

nach dem Durchlaufen der while-Schleife, wo die Ecke $e_j$ aus $T$ entfernt wurde. Da $d(e_{j+1})$ auch im weiteren höchstens verkleinert wird, gilt diese Ungleichung auch an dem Punkt noch, wo $f$ aus $T$ herausgenommen wird.

Man hat dann

$$d(e_{j+1}) \leq d(e_j) + w(e_je_{j+1}) = |s, e_j| + w(e_je_{j+1}) = |s, e_{j+1}| \leq |s, f|\,.$$

Es kann nun $e_{j+1} = f$ bzw. $j+1 = n$ sein oder $j+1 < n$. Im ersten Fall kann man $d(f) \leq |s, f|$ sofort ablesen. Im zweiten Fall führt die Annahme $|s, f| < d(f)$ zu der Beziehung $d(e_{j+1}) < d(f)$, und $e_{j+1}$ wäre im Algorithmus vor $f$ aus $T$ entfernt worden im Widerspruch zur Festsetzung des Index $j$. Also ist $d(f) \leq |s, f|$ gezeigt.

Zu der Aussage über die Komplexität beachte man, dass in der while-Schleife zur Bestimmung des minimalen $d(e)$ $|T| - 1$ viele Vergleiche nötig sind, was insgesamt zu $O(|E|^2)$ führt. Gleiches gilt für die Aktualisierung der $d$-Werte in derselben Schleife, weshalb es insgesamt zu der angegebenen Komplexität kommt.

**Beispiel 6.1-4:**
In einem Datennetz seien den verschiedenen Leitungen (Kanten des Graphen) Wahrscheinlichkeiten für ihr Versagen zugeordnet. Man möchte zwischen den

Punkten die Wege nutzen, die mit der größten Wahrscheinlichkeit nicht zusammenbrechen. Auf folgende Weise kann man dieses Problem auf die Bestimmung kürzester Wege zurückführen:

Die Ecken seien $1, \ldots, n$ durch numeriert, und für eine Kante $ij$ sei $p(ij)$ die Wahrscheinlichkeit dafür, dass sie nicht versagt. Unter der Annahme, dass Störungen von Leitungen unabhängig von einander auftreten (dies ist nicht ganz realistisch), ist $p(k_1) \cdot \ldots \cdot p(k_n)$ die Wahrscheinlichkeit dafür, dass der Kantenzug $k_1, \ldots, k_n$ ohne Störung genutzt werden kann. Gesucht ist zwischen den Punkten natürlich ein Kantenzug, für den diese Wahrscheinlichkeit maximal ist. Ersetzt man jeweils $p(k)$ durch $log\ p(k)$, so kann stattdessen

$$log\ (p(k_1) \cdot \ldots \cdot p(k_n)) = log\ p(k_1) + \ldots + log\ p(k_n)$$

maximiert werden. Geht man schießlich zu der Kantenbewertung $w(k) := -log\ p(k)$ über, so gilt wegen $0 \leq p(k) \leq 1$ stets $w(k) \geq 0$, und man hat als Ziel die Minimierung von $w(k_1) + \ldots + w(k_n)$. Mit anderen Worten ist das Problem der Bestimmung von Wegen größter Zuverlässigkeit auf das kürzester Wege zurückgeführt. Dies kann, wenn man alle Ecken als Anfangs- und Endpunkte betrachten will, z. B. durch mehrfache Anwendung des Algorithmus von Dijkstra angegangen werden. Im Folgenden werden für dieses Problem noch alternative Vorgehensweisen behandelt.

Wir wenden uns nun dem Algorithmus von Floyd-Warshall zu. Dabei sind auch negative Kantenlängen zugelassen, aber natürlich keine negativen Kreise. Der Algorithmus bestimmt die kürzesten Abstände zwischen je zwei Ecken eines gegebenen Netzwerkes. Im Falle nicht-negativer Kantenlängen könnte man diese auch durch die mehrfache Anwendung des Dijkstra-Algorithmus bestimmen (indem man jede Ecke einmal als $s$ wählt).

**Algorithmus von Floyd-Warshall**

**Algorithmus von Floyd-Warshall**

Gegeben sei ein Netz $(G, w)$ ohne negative Kreise, die Eckenmenge sei $\{1, \ldots, n\}$. Es werden die Abstände zwischen je zwei Ecken bestimmt.

**Algorithmus**: Algorithmus von Floyd-Warshall

```
begin
   for i = 1 to n do
      for j = 1 to n do
         if i ≠ j then setze d^0 (i, j) := w_ij
                  else setze d^0 (i, j) := 0;
   for k = 1 to n do
      for i = 1 to n do
         for j = 1 to n do
           setze
           d^k (i, j) := min { d^{k-1} (i, j), d^{k-1} (i, k) + d^{k-1} (k, j)};
end
```

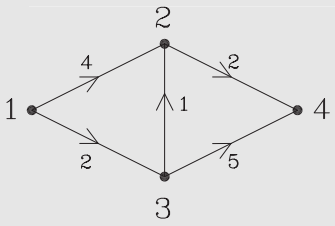
Man beachte, dass in dem Algorithmus zunächst vorläufige Abstände $d^0(i,j)$ mithilfe der Matrix $(w_{ij})$ (s. Definition 6.1-2) initialisiert werden, allerdings mit dem Unterschied, dass "auf der Diagonalen" $d^0(i,i) = 0$ gesetzt wird - dies geschieht, damit der folgende Satz so allgemein richtig ist.

**Satz 6.1-5:** *Nach dem Durchlaufen des Algorithmus von Floyd-Warshall ist $d^n(i,j) = |i,j|$ für alle Ecken $i,j$. Die Komplexität ist $O(|E|^3)$.*

**Beweis:** Die mit den ersten beiden for-Iterationen durchgeführte Initialisierung führt zu einer Matrix $D_0 = (d^0_{ij})$. In den darauf folgenden Iterationen wird für jeden Wert von $k$ eine Matrix $D_k = (d^k_{ij})$ erzeugt. Mit Induktion kann nun leicht gezeigt werden, dass $d^k_{ij}$ die kürzeste Länge eines (gerichteten) Weges von $i$ nach $j$ ist, der nur Ecken aus $\{1,\ldots,k\}$ verwendet (von $i$ und $j$ abgesehen). Mit $k = n$ ist dann die Behauptung gezeigt. Die Komplexitätsaussage ergibt sich sofort aus der Ineinanderschachtelung der drei for-Iterationen.

Die obere Indizierung $k$ bei $d^k(i,j)$ im Algorithmus von Floyd-Warshall hilft zum Verständnis des logischen Ablaufs. Eine kurze Überlegung zeigt jedoch, dass es auch ausreichen würde, im ganzen Algorithmus nur eine doppelt indizierte Variable $d(i,j)$ zuverwenden.

**Beispiel 6.1-5:**
Wir betrachten dasselbe Netz wie in Beispiel 6.1-3.




Mit dem Algorithmus von Floyd-Warshall ergeben sich die folgenden Matrizen:

$$D_0 = \begin{pmatrix} 0 & 4 & 2 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & 1 & 0 & 5 \\ \infty & \infty & \infty & 0 \end{pmatrix} \qquad D_1 = \begin{pmatrix} 0 & 4 & 2 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & 1 & 0 & 5 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

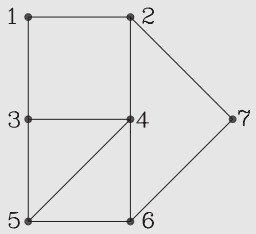
$$D_2 = \begin{pmatrix} 0 & 4 & 2 & 6 \\ \infty & 0 & \infty & 2 \\ \infty & 1 & 0 & 3 \\ \infty & \infty & \infty & 0 \end{pmatrix} \qquad D_3 = \begin{pmatrix} 0 & 3 & 2 & 5 \\ \infty & 0 & \infty & 2 \\ \infty & 1 & 0 & 3 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$D_4 = D_3$

Die nach einer Anwendung des Algorithmus von Floyd-Warshall zuletzt erhaltene Matrix $D_n$ kann auch dazu genutzt werden, eine Ecke des gegebenen Graphen zu bestimmen, die möglichst **zentral** in dem Netzwerk liegt: der weiteste Abstand zu irgendeiner der anderen Ecken soll möglichst klein sein. Dies bedeutet, dass zuerst in $D_n$ das Maximum in jeder Zeile zu bestimmen ist und der kleinste dieser Werte herausgesucht werden muss. Diese Anwendung ist natürlich am interessantesten für Netze, bei denen es zwischen je zwei Ecken einen Weg gibt.

**zentrale Ecke**

**Beispiel 6.1-6:**
Wir wenden den Algorithmus von Floyd-Warshall auf den folgenden Graphen an, der - wie schon oft an früherer Stelle - ein Datennetz darstellen möge. Jede Kante sei mit $1$ bewertet.




Als erste und letzte Matrix ergeben sich:

$$D_0 = \begin{pmatrix} 0 & 1 & 1 & \infty & \infty & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty & \infty & 1 \\ 1 & \infty & 0 & 1 & 1 & \infty & \infty \\ \infty & 1 & 1 & 0 & 1 & 1 & \infty \\ \infty & \infty & 1 & 1 & 0 & 1 & \infty \\ \infty & \infty & \infty & 1 & 1 & 0 & 1 \\ \infty & 1 & \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

$$D_7 = \begin{pmatrix} 0 & 1 & 1 & 2 & 2 & 3 & 2 \\ 1 & 0 & 2 & 1 & 2 & 2 & 1 \\ 1 & 2 & 0 & 1 & 1 & 2 & 3 \\ 2 & 1 & 1 & 0 & 1 & 1 & 2 \\ 2 & 2 & 1 & 1 & 0 & 1 & 2 \\ 3 & 2 & 2 & 1 & 1 & 0 & 1 \\ 2 & 1 & 3 & 2 & 2 & 1 & 0 \end{pmatrix} \begin{matrix} 3 \\ 2 \\ 3 \\ 2 \\ 2 \\ 3 \\ 3 \end{matrix}$$

Neben $D_7$ sind die Maxima der jeweiligen Zeilen notiert. Wie man sieht, sind die Ecken $2$, $4$ und $5$ zentral.

Man kann den Algorithmus von Floyd-Warshall natürlich auch ablaufen lassen für ein Netz, welches einen negativen Kreis enthält. Allerdings stimmt dann die Aussage des Satzes Satz 6.1-5 nicht mehr - es ist ja nicht einmal der Abstand $|i, j|$ für alle Eckenpaare definiert. Offenbar gibt es in $(G, w)$ genau dann einen negativen

Kreis durch die Ecke $i$, wenn im Algorithmus $d^k(i,i) < 0$ wird. Mit der folgenden Abwandlung des Algorithmus von Floyd-Warshall kann also für ein beliebiges Netz

- entweder $|i,j|$ für alle Ecken $i,j$ berechnet werden
- oder festgestellt werden, dass ein negativer (gerichteter) Kreis existiert. (In diesem Fall bricht der Algorithmus vorzeitig ab, oder es wird bei $k = n$ ein $d^n(i,i) < 0$.)

**Algorithmus**: Algorithmus von Floyd-Warshal

```
begin
   for i = 1 to n do
      for j = 1 to n do
         if i ≠ j then setze d^0 (i, j) := w_ij
           else setze d^0 (i, j) := 0;
   setze k := 0;
   while d^k (i, i) ≥ 0 für alle i und k < n do
     begin
        setze k := k + 1;
        for i = 1 to n do
         for j = 1 to n do
           setze
           d^k (i, j) := min { d^{k-1} (i, j), d^{k-1} (i, k) + d^{k-1} (k, j) };
     end
end
```

Bei den Algorithmen dieses Abschnitts haben wir uns bisher mit der Berechnung kürzester Abstände begnügt. Dazu gehörige kürzeste Wege können in verschiedener Weise bestimmt werden - z. B. steckt eine Idee dazu in der bei Satz 6.1-3 verwendeten Konstruktion . Es ist jedoch auch einfach, die kürzesten Wege in den Algorithmen gleich mitzunotieren. Exemplarisch zeigen wir dies anhand der soeben behandelten Variante des Algorithmus von Floyd-Warshall. Es wird entweder ein negativer Kreis angegeben, oder nach dem Ablauf des Algorithmus ist für alle Eckenpaare $i,jv_{ij}$ eine Vorgängerecke von $j$ in einem kürzesten Weg von $i$ nach $j$; die kürzesten Wege können daraus sofort rekonstruiert werden. (Es ist $v_{ij} = \infty$, falls kein Weg von $i$ nach $j$ existiert, und $v_{ij} = 0$ für $i = j$.)

**Algorithmus**: Algorithmus von Floyd-Warshal

```
begin
   for i = 1 to n do
      for j = 1 to n do
         if i ≠ j then setze d^0 (i, j) := w_ij
            else setze d^0 (i, j) := 0;
         if i ≠ j then
               if w_ij < ∞ then setze v_ij := i
                    else setze v_ij := ∞
              else setze v_ij := 0;
   setze k := 0;
   while d^k (i, i) ≥ 0 für alle i und k < n do
      begin
```

```
        setze k := k + 1;
        for i = 1 to n do
          for j = 1 to n do
           if d^{k-1}(i, k) + d^{k-1}(k, j) < d^{k-1}(i, j)
             then setze v_ij := v_kj
             und d^k(i, j) := d^{k-1}(i, k) + d^{k-1}(k, j)
             else setze d^k(i, j) := d^{k-1}(i, j);
    end
end
```

**Beispiel 6.1-7:**
Wir behandeln nochmals das Beispiel 6.1-5. Im Folgenden ist eine Übersicht gegeben, wie sich nach der Initialisierung $v_{ij} = i$ die $v_{ij}$-Werte mit der Iteration über $k$ ändern:

$$k = 1 : (v_{ij}) = \begin{pmatrix} 0 & 1 & 1 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 3 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$k = 2 : (v_{ij}) = \begin{pmatrix} 0 & 1 & 1 & 2 \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 2 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$k = 3 : (v_{ij}) = \begin{pmatrix} 0 & 3 & 1 & 2 \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 2 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$k = 4 : (v_{ij}) = \begin{pmatrix} 0 & 3 & 1 & 2 \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 2 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Aus der ersten Zeile der letzten Matrix lässt sich z. B. rekonstruieren, dass ein kürzester Weg von $1$ nach $4$ den Verlauf $1 \to 3 \to 2 \to 4$ hat. Der Eintrag $v_{14} = 2$ besagt zunächst, dass $2$ der Vorgänger von $4$ auf dem gesuchten kürzesten Weg ist; $v_{12} = 3$ und $v_{13} = 1$ führen dann zu dem Weg $1 \to 3 \to 2 \to 4$.

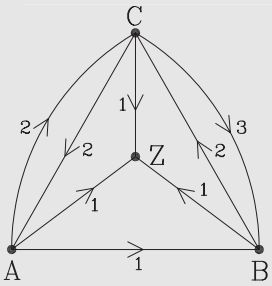
Es wurde oben schon einmal darauf hingewiesen, dass es auch bei der Existenz negativer Kreise stets kürzeste **Wege** gibt - wenn auch für einige Eckenpaare keine kürzesten Kantenfolgen. Einen guten Algorithmus zur Bestimmung kürzester Wege in diesen Fällen hat man allerdings bisher nicht gefunden.

Es gibt Anwendungen, bei denen man für ein Netzwerk nicht nur an einem kürzesten Weg zwischen zwei Ecken, sondern beispielsweise an den zwei kürzesten Wegen interessiert ist. Ein solches Problem stellt sich etwa, wenn in einem Kommunikationsnetz eine Station zum Verschicken von Nachrichten an eine andere Station - wegen möglicher Ausfälle von Leitungen oder Zwischenpunkten - zwei Wege zu diesem Ziel zur Verfügung haben soll.

**Definition 6.1-3:** *Mit Hilfe der bisher behandelten Algorithmen ist es leicht, für jedes Eckenpaar die beiden kürzesten Wege zu bestimmen. Jedoch kann es - je nach dem betrachteten Kommunikationsprotokoll - nun zu weiteren Komplikationen kommen, etwa dem Kreisen von Nachrichten im Netz. Mit solchen* ***Routing-Problemen*** *werden wir uns im zweiten Teil des Kurses beschäftigen.*

**Routing-Probleme**

**Beispiel 6.1-8:**
Für das in dem Diagramm dargestellte Kommunikationsnetz sei ein Kommunikationsprotokoll vereinbart, bei dem jede Ecke (in der Praxis ein Vermittlungsrechner) eine nicht für ihn bestimmte Nachricht entsprechend der von ihr getragenen Zieladresse zu einem seiner Nachbarn sendet. Diesen Nachbarn bestimmt er nach einer von ihm gespeicherten Routing-Tafel, die nach dem Prinzip der zwei kürzesten Wege erstellt wurde.




Für das Ziel $Z$ haben die Punkte $A$, $B$, $C$ die folgenden Tafeln:

$$A : Z, B \qquad B : Z, C \qquad C : Z, A \qquad \text{6.1-1}$$

(Dies bedeutet: $A$ schickt eine für $Z$ bestimmte Nachricht nach $B$, falls die direkte Leitung nach $Z$ ausgefallen ist, usw.)

Fallen nun alle an $Z$ angeschlossenen Leitungen aus, und initiiert $A$ eine Nachricht mit Ziel $Z$, so kreist diese Nachricht im Netz ($A \to B \to C \to A \to \ldots$), sofern das Protokoll keine Vorkehrungen enthält, die dies "bemerken"und die Nachricht eliminieren.

**längster Weg**

In einem Netz $(G, w)$ versteht man unter einem **längsten Weg** von Ecke $a$ nach Ecke $b$ einen Weg mit der Eigenschaft, dass kein Weg von $a$ nach $b$ eine größere Länge hat. Im Prinzip kann das Problem der Bestimmung längster Wege durch Übergang

zu der Bewertung $-w$ mit den im vorigen Abschnitt behandelten Methoden angegangen werden, denn ein längster Weg in $(G, w)$ entspricht einem kürzesten Weg in $(G, -w)$. Es ist allerdings zu beachten, dass beim Übergang von $w$ zu $-w$ negative Kreise entstehen können. Die von negativen Kreisen herrührenden Komplikationen bei kürzesten Wegen entsprechen natürlich den von positiven Kreisen kommenden bei längsten Wegen.

**Beispiel 6.1-9:**
Wir gehen noch einmal zu Beispiel 6.1-4, wo in einem Datennetz mit Wahrscheinlichkeiten $p(k)$ für das Nicht-Versagen der Kanten $k$ nach den verlässlichsten Wegen zwischen den Punkten gefragt wurde. Durch Übergang zu der Bewertung $log\ p(k)$ gelangte man zu der Frage nach längsten Wegen. (Die Bewertung $-log\ p(k)$ führte schließlich zu kürzesten Wegen.)

Die Bestimmung längster Wege in kreisfreien Digraphen ist auch in der Projektplanung (als Teilgebiet von Operations Research) von Interesse. Wir betrachten die folgende spezielle Problemstellung: Ein Projekt bestehe aus Teilaufgaben, deren Bearbeitung jeweils eine gewisse Zeit benötigt. Gewisse Teilaufgaben können nicht vor der Beendigung anderer Aufgaben begonnen werden. Gefragt wird nach der kürzest möglichen Gesamtdauer des Projekts. Man ordnet jeder der $n$ vielen Teilaufgaben eine Ecke $i (1 \leq i \leq n)$ eines Digraphen $G$ zu, wobei eine gerichtete Kante $ij$ existiert, wenn $i$ vor Beginn von $j$ beendet sein muss. Eine zusätzliche Ecke $s$ wird mit allen $i$ verbunden, die keinen Vorgänger haben, eine andere zusätzliche Ecke $z$ ist Endpunkt von Kanten $iz$ für alle $i$, die sonst keinen Nachfolger haben. Schließlich werden die Kanten $ij$ mit der Bearbeitungsdauer von $i$ bewertet (dabei ist $j = z$ eingeschlossen), die Kanten $si$ mit $0$.

**Definition 6.1-4:** *Es ist nun offensichtlich, dass die Längen aller Wege (damit auch die eines längsten Weges) von $s$ nach $z$ untere Schranken für die Gesamtdauer des Projekts sind. Deshalb nennt man die längsten Wege in diesem Zusammenhang auch* ***kritische Wege****.*

**kritischer Weg**

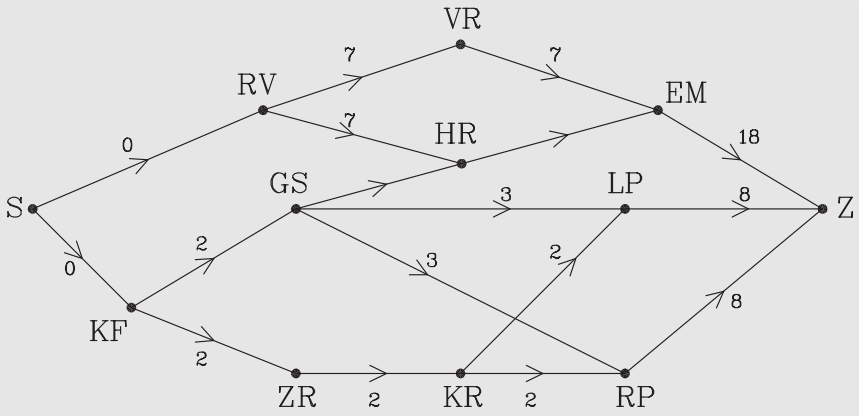
In Operations Research gibt es verschiedene Vorgehensweisen zur Bestimmung kritischer Wege bei solchen und ähnlichen Fragestellungen, wo die Probleme oft noch durch Fragen wie ”Wann kann Teilaufgabe $i$ frühestens begonnen werden?” angereichert sind. Man spricht hier allgemein von der Methode der kritischen Wege (”critical pathmethod”).

**Beispiel 6.1-10:**
In der Montageabteilung einer Fahrradfabrik wird der Zusammenbau eines Fahrrads in die folgenden Tätigkeiten aufgespalten:

KF - Kettenführung an Rahmen montieren (2 min.)
ZR - Zahnrad an Kurbel montieren (2 min.)
KR - Zahnrad/Kurbel am Rahmen montieren (2 min.)

LP - Linke Pedale anbringen (8 min.)
RP - Rechte Pedale anbringen (8 min.)
GS - Gangschaltung an Hinterrad montieren (3 min.)
HR - Hinterrad montieren (7 min.)
VR - Vorderrad montieren (7 min.)
RV - Rahmen vorbereiten (Gabel, Schutzblech etc.) (7 min.)
EM - Endmontage (Sattel, Bremsen etc.) (18 min.)

Es ist jeweils angegeben, welche Zeit ein Monteur für die Tätigkeit benötigt. Durch Berücksichtigung der Einschränkungen in der Reihenfolge kommt man zu folgendem Digraphen mit Kantenbewertung:




$s \to RV \to VR \to EM \to z$ ist ein längster (und damit kritischer) Weg von $s$ nach $z$, seine Länge ist 32. Dies bedeutet: die Montage eines Fahrrads dauert mindestens 32 Minuten, auch wenn einzelne Tätigkeiten - sofern dies möglich ist - von mehreren Monteuren parallel verrichtet werden.

**Selbsttestaufgabe 6.1-1:**

*Erläutern Sie die Problematik der Existenz kürzester Wege bzw. Kantenfolgen in einem Netz mit einem negativen Kreis.*

## 6.2 Minimale Gerüste

In Beispiel 5.1-2 haben wir ein Beispiel kennengelernt, wo man für einen bewerteten Graphen an einem Gerüst interessiert ist, für das die Summe der Bewertungen der zu dem Gerüst gehörenden Kanten möglichst klein ist. Ein solches Gerüst heißt **mimimales Gerüst**. Hauptthema dieses Abschnitts sind Algorithmen, minimale Gerüste zu bestimmen. Alle Graphen werden als ungerichtet vorausgesetzt.

**mimimales Gerüst**

Wir wollen uns auch hier klarmachen, dass es unbedingt lohnend ist, nach guten Algorithmen zur Bestimmung minimaler Gerüste zu suchen. Wir zeigen, dass die Anzahl der Gerüste der vollständigen Graphen $K_n$ eine exponentielle Funktion der Eckenzahl $n$ ist:

**Satz 6.2-1: Satz von Cayley**

*Für beliebiges $n$ hat der vollständige Graph $K_n$ genau $n^{n-2}$ viele Gerüste.*

**Beweis:** Im Beweisgang folgen wir Ideen von Prüfer, die in einer Arbeit von 1918 enthalten sind. (Der Originalbeweis von Cayley stammt aus dem Jahre 1889.) Es wird gezeigt, dass es auf der Eckenmenge $\{1, 2, \ldots, n\} n^{n-2}$ viele verschiedene Bäume gibt - dies ist offenbar äquivalent zu der Behauptung des Satzes. Es sei $\mathbb{B}_n$ die Menge dieser Bäume. Wie üblich, bezeichnet $\{1, \ldots, n\}^{n-2}$ die Menge aller $(n-2)$-tupel von Zahlen zwischen $1$ und $n$, wovon es natürlich $n^{n-2}$ viele gibt. Wir konstruieren nun eine bijektive Abbildung zwischen $\mathbb{B}_n$ und $\{1, \ldots, n\}^{n-2}$. Damit ist dann der Satz gezeigt.

Mithilfe der folgenden beiden Algorithmen konstruieren wir zu gegebenem Baum $B \in \mathbb{B}_n$ eine Folge $f(B) \in \{1, \ldots n\}^{n-2}$ und zu gegebener Folge $x \in \{1, \ldots n\}^{n-2}$ einen Baum $g(x) \in \mathbb{B}_n$. Es ist dann $g(f(B)) = B$ und $f(g(x)) = x$, womit $f$ und $g$ als bijektive Abbildungen nachgewiesen sind.
Definition von $f$:
Gegeben sei ein Baum $B \in \mathbb{B}_n$. Es wird eine Folge $x = f(B) \in \{1, \ldots, n\}^{n-2}$ konstruiert.

**Algorithmus**:

```
begin
   Setze i := 1;
   while i < n − 2 do
      begin
   sei j die Endecke des verbliebenen Baums
         mit kleinster Nummer, entferne j und
         die Kante jk aus dem Baum,
         setze x_i := k und i := i + 1;
      end
end
```

Definition von $g$:
Gegeben sei eine Folge $x = (x_1, \ldots, x_{n-2}) \in \{1, \ldots, n\}^{n-2}$. Es wird ein Baum $B = g(x) \in \mathbb{B}_n$ konstruiert.

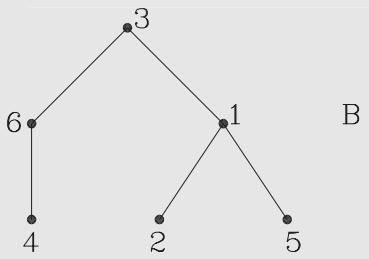
**Algorithmus**:

```
begin
   for t = 1 to n do
      setze d(t) := 1+ (Anzahl der r mit x_r = t);
   setze i := 1;
   while i < n − 2 do
      begin
      sei j die kleinste Ecke mit d(j) = 1, ziehe
      eine Kante von j nach x_i und setze
```

```
        d(j) := 0, d(x_i) := d(x_i) − 1 und i := i + 1;
        end
    verbinde die beiden Ecken mit d-Wert 1
    durch eine Kante;
end
```

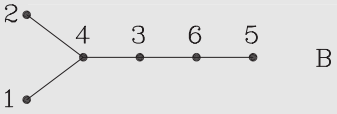
Den nun noch zu erbringenden Nachweis $g(f(B)) = B$ und $f(g(x)) = x$ wollen wir dem Leser als Übung überlassen.

**Beispiel 6.2-1:**
Wir gehen zunächst von folgendem Baum $B$ aus:




Die Anwendung des ersten Algorithmus ergibt die Folge $x = f(B) = (1, 6, 1, 3)$: Zunächst ist $2$ die kleinste Endecke, deren Nachbar ist $1$, folglich wird $x_1 = 1$, usw. Anwendung des zweiten Algorithmus auf die Folge $(1, 6, 1, 3)$ ergibt nun wieder den Baum $B$ als $g(x) = g(f(B))$:
Zuerst ist $2$ die kleinste Endecke mit $d$-Wert $1$, sie wird mit $x_1 = 1$ verbunden; danach wird $4$ mit $6$, dann $5$ mit $1$ und $1$ mit $3$ verbunden; im letzten Schritt wird die Kante von $3$ nach $6$ gezogen.

Umgekehrt kann von einem beliebigen $x \in \{1, \ldots, 6\}^4$ ausgegangen werden, z. B. $x = (4, 4, 3, 6)$. Mit dem zweiten Algorithmus ergibt sich folgender Baum $B$ als $g(x)$:




Anwendung des ersten Algorithmus ergibt wieder $x = f(B) = f(g(x))$.

Wir wollen die Frage nach Gerüsten hier auch zum Anlass nehmen, noch einmal auf die Suchprinzipien ”Breadth first search” und ”Depth first search” einzugehen. Wenn wir bei dem Algorithmus Breath first search des vorrigen Kapitels die Berechnung der $d$-Werte herausnehmen und stattdessen ein Gerüst konstruieren, so kommen wir zu dem folgenden Algorithmus. Er unterscheidet sich von dem oben

behandelten Algorithmus nur dadurch, dass die Reihenfolge des Hinzunehmens von Ecken genauer beschrieben wird.

### Algorithmus "Gerüst mit Breadth first search"

**Algorithmus "Gerüst mit Breadth first search"**

Gegeben sei ein zusammenhängender Graph $G = (E, K)$ auf der Eckenmenge $\{1, \ldots, n\}$, $N(e) \subseteq E$ sei zu $e \in E$ die Menge der Nachbarecken, $s \in E$ fest gewählt. Es wird ein Gerüst $(E, K')$ konstruiert.

Der Algorithmus verwendet als Datenstruktur eine Liste $L$.

**Algorithmus**:

```
begin
   Setze L := { s }, E':={ s }, K':=∅;
   while E' ≠ E do
      begin
      entferne die erste Ecke v von L aus L;
      for w ∈ N(v) do
         begin
           if w ∉E'
           then setze E':=E'∪ { w },
           K':=K'∪ { vw },
            füge w ans Ende von L an;
         end
      end
end
```

In Kontrast zu obigem Algorithmus, der den Graphen "in die Breite" untersucht und so ein Gerüst konstruiert, steht der folgende Algorithmus, der "in die Tiefe" geht. Der wesentliche Unterschied im Ablauf ist, dass die Liste $L$, an die Elemente angefügt werden, auch wieder von hinten abgebaut wird:

### Algorithmus "Gerüst mit Depth first search"

**Algorithmus "Gerüst mit Depth first search"**

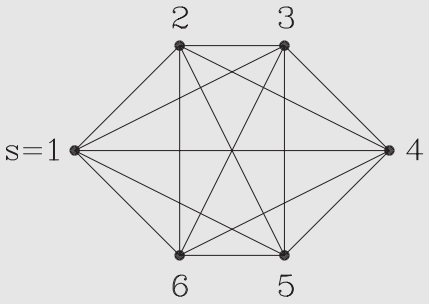
**Algorithmus**: Gerüst mit Depth first search

```
begin
   Setze L:= { s }, E' := { s }, K' := ∅;
   while E' ≠ E do
      begin
         repeat
           entferne den letzten Punkt v
           von L aus L
         until N(v) ⊈ E';
         sei w die kleinste Ecke
         mit w ∈ N(v), w ∉ E';
         setze E':= E' ∪ { w }, K':= K' ∪ { vw },
         füge w ans Ende von L an;
      end
end
```

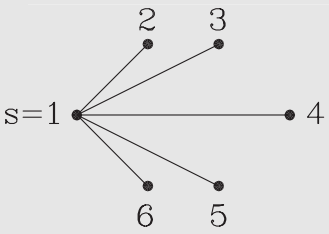
Man kann die Situation folgendermaßen beschreiben: funktioniert die Liste $L$ nach dem Prinzip ”first in - first out” (dem Prinzip einer Warteschlage), so macht man Breadth first search; arbeitet $L$ nach ”last in - first out” (Stapelspeicher), so ergibt sich Depth first search.

Die beiden unterschiedlichen Algorithmen werden an einem Beispiel illustriert.

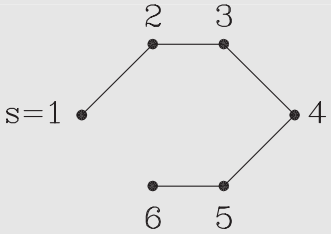
**Beispiel 6.2-2:**
Gegeben sei der vollständige Graph $K_6$ mit Eckenmenge $\{1, \ldots, 6\}$, die Ecke $1$ sei als $s$ gewählt.




Mit Breadth first search erhält man das Gerüst $G_1$, mit Depth first search $G_2$.




$G_1$




$G_2$

Wir wenden uns nun der Bestimmung minimaler Gerüste in beliebig bewerteten Graphen zu. Zuerst sollen jedoch Charakterisierungen minimaler Gerüste gegeben werden, mit deren Hilfe man sich dann später von der Korrektheit der angegebenen Algorithmen überzeugen kann.

Es sei $B$ ein Gerüst in dem Graphen $G$ und $s$ eine nicht zu $B$ gehörende Kante, also eine Sehne. Bereits in Definition 1.3-5 hatten wir festgestellt, dass durch das Hinzufügen von $s$ zu $B$ genau ein (fundamentaler) Kreis entsteht, den wir nun mit $C_B(s)$ bezeichnen wollen.

**Satz 6.2-2:** *Es sei $(G, w)$ ein Netz und $G$ zusammenhängend, $B$ sei ein Gerüst von $G$. $B$ ist genau dann minimal, wenn für jede Sehne $s$ gilt:*

$w(s) \geq w(t)$ *für alle Kanten* $t$ *in* $C_B(s)$.

**Beweis:** Zunächst sei $B$ als minimal vorausgesetzt. Angenommen, es gibt eine Sehne $s_0$ und eine Kante $t_0$ in $C_B(s_0)$ mit $w(s_0) < w(t_0)$. Wenn wirnun aus $B$ $t_0$ entfernen und statt dessen $s_0$ hinzufügen, so erhalten wir einen Baum $B_0$ von $G$ mit kleinerem Gewicht, und dies ist ein Widerspruch. Also muss für jede Sehne $s$ gelten $w(s) \geq w(t)$ für alle Kanten $t$ in $C_B(s)$.
Umgekehrt sei nun $w(s) \geq w(t)$ für jede Sehne $s$ und alle $t$ in $C_B(s)$ erfüllt. Es sei $B'$ ein minimales Gerüst. Wir zeigen, dass $\sum\limits_{k\in B} w(k) = \sum\limits_{k\in B'} w(k)$ gilt und somit $B$ ebenfalls minimal ist. Gehört jede Kante von $B'$ auch zu $B$, so muss $B = B'$ sein, und es ist nichts zu beweisen. Es sei nun $k'$ eine Kante in $B'$, die nicht zu $B$ gehört. Entfernt man $k'$ aus $B'$, so zerfällt $B'$ in zwei Teile $B'_1$ und $B'_2$; nimmt man nun die Kanten von $C_B(k')$ (mit Ausnahme von $k'$) wieder hinzu, so werden $B'_1$ und $B'_2$ verbunden, d. h. es kann eine Kante $k$ aus $C_B(k')\backslash\{k'\}$ gewählt werden, die eine Ecke von $B'_1$ mit einer von $B'_2$ verbindet. So sind wir also von $B'$ durch Herausnahme von $k'$ und Hinzufügen von $k$ zu einem Gerüst $B''$ gekommen, und wegen der Minimalität von $B'$ gilt $w(k) \geq w(k')$.
Andererseits muss nach der gemachten Voraussetzung aber auch $w(k') \geq w(k)$ gelten, d. h. es ist auch $B''$ als minimales Gerüst nachgewiesen, denn es gilt

$$\sum_{k\in B''} w(k) = \sum_{k\in B'} w(k).$$

$B''$ hat mit $B$ eine Kante mehr gemeinsam als $B'$. Mit Induktion folgt jetzt, dass auch $B$ ein minimales Gerüst ist.

Auch mit Hilfe fundamentaler Schnitte können die minimalen Gerüste charakterisiert werden. Dazu verwenden wir die folgende Bezeichnung:
Ist $B$ ein Gerüst in dem Graphen $G$ und $z$ ein Zweig von $B$, so ist $S_B(z)$ der zugehörige fundamentale Schnitt.

**Satz 6.2-3:** *Es sei* $(G, w)$ *ein Netz und* $G$ *zusammenhängend,* $B$ *sei ein Gerüst von* $G$. $B$ *ist genau dann minimal, wenn für jeden Zweig* $z$ *von* $B$ *gilt:*

$w(z) \leq w(t)$ *für alle Kanten* $t$ *in* $S_B(z)$.

**Beweis:** Zunächst sei $B$ als minimal vorausgesetzt. Angenommen, es gibt einen Zweig $z_0$ und eine Kante $t_0$ in $S_B(z_0)$ mit $w(z_0) > w(t_0)$. Man erhält durch Entfernen von $z_0$ aus $B$ und Hinzufügen von $t_0$ ein Gerüst mit geringer Kantenbewertungssumme, ein Widerspruch zur Minimalität von $B$.
Umgekehrt gelte nun für jeden Zweig $z$

$w(z) \leq w(t)$ für alle Kanten $t$ in $S_B(z)$.

Wir führen den Beweis auf Satz 6.2-2 zurück, indem wir zeigen, dass für jede Sehne $s$ die Ungleichung

$w(s) \geq w(t)$ für jede Kante t in $C_B(s)$ gilt.

Es sei also $s$ eine Sehne bezüglich $B$ und $t$ ein Kante in $C_B(s)$. Wir betrachten den von dem Zweig $t$ induzierten Schnitt $S_B(t)$. Da $s$ zu $S_B(t)$ gehört, gilt aufgrund der Voraussetzung $w(t) \leq w(s)$, was auch gezeigt werden sollte.

Die folgende Bezeichnung wird im weiteren verwendet: ist $E'$ eine Teilmenge der Eckenmenge $E$ eines Graphen, so ist $[E', E\backslash E']$ der zugehörige Schnitt, also die Menge aller Kanten, die eine Ecke von $E'$ mit einer aus $E\backslash E'$ verbinden.

**Algorithmus von Prim**

**Algorithmus von Prim** Gegeben sei ein Netz $(G, w)$ auf dem zusammenhängenden Graphen $G = (E, K)$, $s \in E$ sei eine Ecke. Es wird ein minmales Gerüst $(E, K')$ bestimmt.

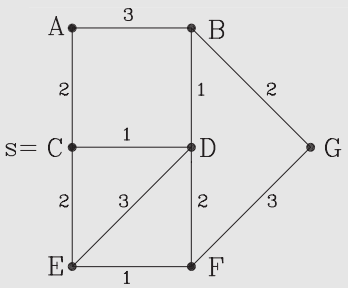
**Algorithmus**: Algorithmus von Prim

```
begin
   Setze E':={ s }, K':=∅;
   while E'≠ E do
      begin
         sei k eine Kante aus [E',E\E'] mit mini-
         maler
         Bewertung w(k), e ihre Endecke in E \E';
         setze E':=E'∪ { e }, K':= K'∪ { k };
      end
end
```

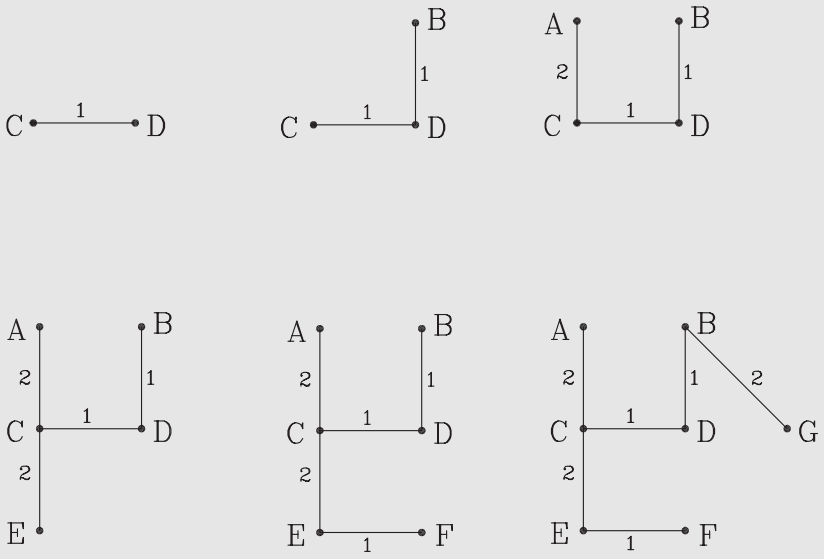
**Satz 6.2-4:** *Der Algorithmus von Prim bestimmt ein minimales Gerüst mit Komplexität $O(|E|^2)$.*

**Beweis:** Dass mit dem Algorithmus ein Gerüst aufgebaut wird, leuchtet sofort ein, denn es wird bei jedem Schritt von dem bereits konstruierten Baum eine Kante zu einer noch nicht erfassten Ecke gezogen. Mit Satz 6.2-3 folgt, dass es sich bei dem Ergebnis um ein minimales Gerüst handelt, denn die Auswahl der Kanten im Algorithmus erfolgt gerade so, dass die Bedingungin Satz 6.2-3 erfüllt ist. Die Komplexitätsaussage ist ebenfalls offensichtlich.

**Beispiel 6.2-3:**
Wir greifen das Netz aus Beispiel 5.1-2 noch einmal auf, die Ecke $C$ sei als Startecke $s$ gewählt.



Die folgenden Diagramme zeigen die Folge der Bäume, wie sie beim Durchlaufen des Algorithmus von Prim produziert werden (mit der Zusatzregel, dass bei mehreren Möglichkeiten die alphabetisch erste Ecke genommen wird). Endresultat ist dasselbe minimale Gerüst wie in Beispiel 5.1-2.




Die Grundidee des Algorithmus von Prim ist es, von einer Ecke ausgehend einen Baum schrittweise zu einem Gerüst wachsen zu lassen, indem man in jedem Schritt einen möglichst kurzen Zweig neu hinzufügt. Beim nun zum Abschluss behandelten Algorithmus von Kruskal können die Zwischenergebnisse Wälder sein, die sich erst zum Schluss zu einem Gerüst zusammenfügen; er hat oft eine geringere Laufzeit als der Algorithmus von Prim, geht aber von "der Größe nach" geordneten Kanten aus:

**Algorithmus von Kruskal**

**Algorithmus von Kruskal** Gegeben sei ein Netz $(G, w)$ auf dem zusammenhängenden Graphen $G = (E, K)$ mit $K = \{k_1, \ldots, k_m\}$, und es gelte $w(k_1) \leq w(k_2) \leq \ldots \leq w(k_m)$. Es wird ein minimales Gerüst $(E, K')$ bestimmt.

**Algorithmus**: Algorithmus von Kruskal

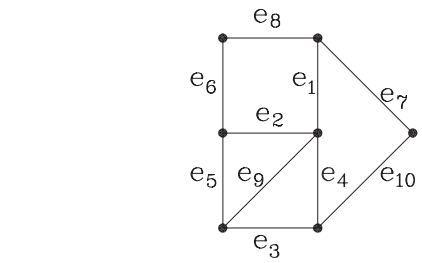
```
begin
   Setze K':=∅;
   for i = 1 to m do
      if k_i bildet mit den Kanten in K' keinen Kreis
         then setze K':=K'∪ { k_i };
end
```

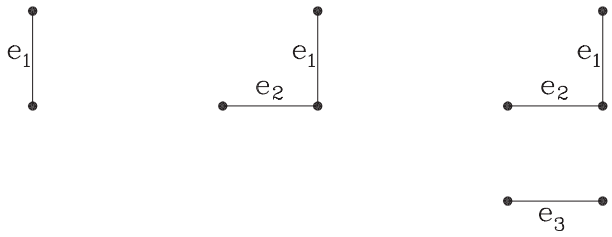
**Satz 6.2-5:** *Der Algorithmus von Kruskal bestimmt ein minimales Gerüst.*

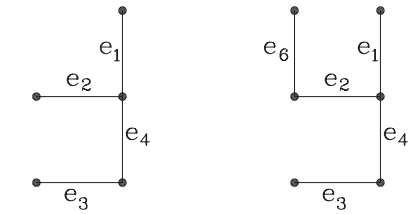
**Beweis:** Der Beweis dieses Satzes ergibt sich sofort aus Satz 6.2-2, denn durch die Auswahl der Kanten im Algorithmus ist sichergestellt, dass die Bedingung für die Sehnen in Satz 6.2-2 erfüllt ist.

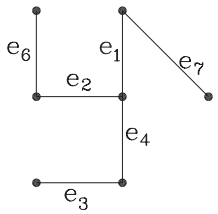
Mit gutem Grund haben wir in Satz 6.2-5 keine Komplexitätsaussagen gemacht. Um dies tun zu können, müsste nämlich der in dem Algorithmus verwendete Test, ob $k_i$ mit den bereits gewählten Kanten einen Kreis bildet, genauer beschrieben werden. Wir wollen darauf an dieser Stelle nicht näher eingehen und nur erwähnen, dass man bei geschicktem Vorgehen so zu einer Komplexität von $O(|K|log|K|)$ für den Algorithmus von Kruskal kommen kann. Dies ist für Graphen mit relativ wenigen Kanten (”dünne Graphen”) besser als $O(|E|^2)$ beim Algorithmus von Prim.

Es wird nochmals das Beispiel 6.2-3 aufgegriffen, die Kanten seien wie im folgenden Bild numeriert. Die weiteren Diagramme zeigen dann die erhaltenen Teilgraphen mit einem minimalen Gerüst als Endresultat.




($e_5$ wurde nicht gewählt, denn es führt zu einem Kreis)

**Selbsttestaufgabe 6.2-1:**

*Erläutern Sie den Unterschied zwischen den Algorithmen von Prim und Kruskal.*

# 7 Flüsse

## 7.1 Einführung

Bei der Untersuchung von Flüssen in Netzen geht es um die folgende Grundfrage: Wieviel kann von einer "Quelle" $q$ zu einer "Senke" $s$ transportiert werden, wenn obere Kapazitätsbeschränkungen für die einzelnen Verbindungen gegeben sind? Dabei werden wir im folgenden stets von einem Digraphen ausgehen, die Ergebnisse lassen sich in der Regel leicht auf den ungerichteten Fall übertragen. Ein Beispiel für einen Fluss in einem Netz haben wir bereits bei der Einführung von Bewertungen in Kapitel 4 kennengelernt. Es folgt nun die formale Definition.

**Definition 7.1-1:** *Es sei $G = (E, K)$ ein Digraph und $c : K \to \mathbb{R}_0^+$ eine Kantenbewertung (Kapazität) mit Werten aus den nicht-negativen reellen Zahlen; $q$ und $s$ seien Ecken in $G$ derart, dass $s$ von $q$ aus erreichbar ist (auf einem gerichteten Weg), jedoch kein Bogen von $s$ nach $q$ führt. Die Struktur $N = (G, c, q, s)$ wird ein* ***Fluss-Netz*** *genannt. Ein* ***Fluss*** *auf $N$ ist eine Kantenbewertung $f : K \to \mathbb{R}_0^+$ mit den folgenden Eigenschaften:*

**Fluss-Netz**
**Fluss**

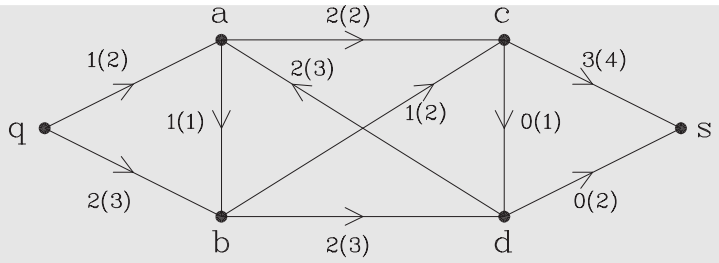
i. $0 \leq f(k) \leq c(k)$ für alle $k \in K$;

ii. für jede Ecke $e$ mit $e \neq q$, $e \neq s$ gilt

$$\sum_{k^+=e} f(k) = \sum_{k^-=e} f(k).$$

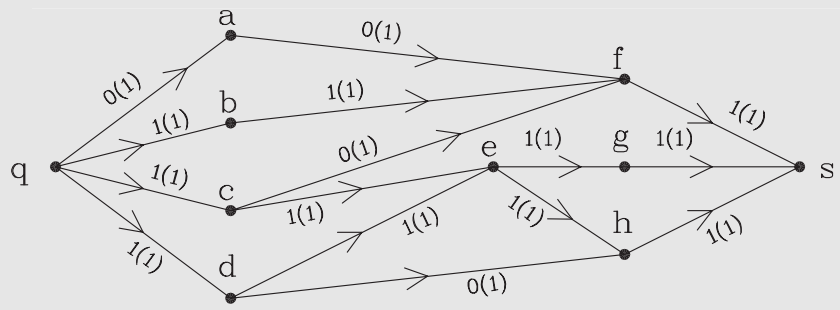
Die beiden Bedingungen besagen, dass der Fluss auf keiner Kante die Kapazität überschreitet, und dass (von Quelle $q$ und Senke $s$ abgesehen) in jede Ecke genauso viel hineinfließt wie herauskommt. Die Voraussetzung, dass es keine Kante $sq$ gibt, ist für die nächsten Überlegungen nicht nötig, jedoch ist sie in den Anwendungen meist gegeben und hilft außerdem, eine schärfere Abgrenzung zu den später behandelten Zirkulationen vorzunehmen.

**Beispiel 7.1-1:**
In dem folgenden Diagramm gibt bei jeder Kante die in Klammern gesetzte Zahl die Kapazität, die davor stehende Zahl den Fluss auf der betreffenden Kante an:




Fluss-Netz $N_1$ mit Fluss $f_1$.

Beim zweiten Beispiel haben alle Kanten dieselbe Kapazität 1:




Fluss-Netz $N_2$ mit Fluss $f_2$.

Bei beiden obigen Beispielen fließt aus $q$ ebensoviel heraus wie in $s$ hinein, nämlich hier jeweils ein Gesamtfluss von 3. Dies ist kein Zufall, wie die folgende Überlegung zeigt:

**Hilfssatz 7.1-1:** *Für einen Fluss $f$ auf einem Fluss-Netz $N = (G, c, q, s)$ gilt stets*

$$\sum_{k^-=q} f(k) - \sum_{k^+=q} f(k) = \sum_{k^+=s} f(k) - \sum_{k^-=s} f(k).$$

**Beweis:** Da der Fluss auf jeder Kante aus einer Ecke heraus und in eine zweite hineinfließt, gilt

$$\sum_{e\in E} \sum_{k^+=e} f(k) = \sum_{e\in E} \sum_{k^-=e} f(k)$$

oder

$$\sum_{e\in E\setminus\{q,s\}} \sum_{k^+=e} f(k) + \sum_{k^+=q} f(k) + \sum_{k^+=s} f(k) = \sum_{e\in E\setminus\{q,s\}} \sum_{k^-=e} f(k) + \sum_{k^-=q} f(k) + \sum_{k^-=s} f(k).$$

Aufgrund von Bedingung (ii) in der Definition eines Flusses sind die Doppelsummen auf beiden Seiten der Gleichung identisch, und die Behauptung folgt.

**Wert eines Flusses**
**maximaler Fluss**

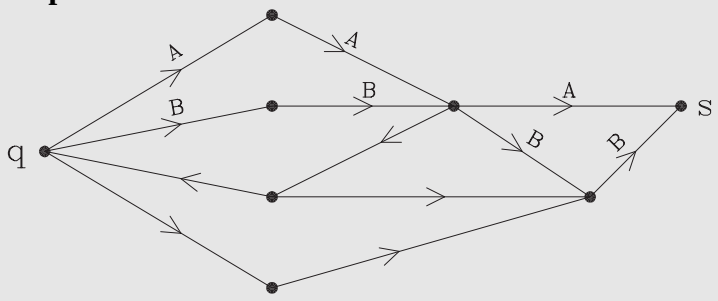
**Definition 7.1-2:** *Der in obigem Hilfssatz betrachtete Wert heißt der* ***Wert des Flusses*** $f$ *und wird mit* $w(f)$ *bezeichnet. Ein Fluss* $f$ *wird* ***maximal*** *genannt, wenn für jeden anderen Fluss* $f'$ *auf demselben Netz gilt* $w(f') \leq w(f)$.

Das Hauptproblem, mit dem wir uns als nächstes beschäftigen, ist die Konstruktion eines maximalen Flusses für ein gegebenes Fluss-Netz. Es sei jedoch darauf hingewiesen, dass - abgesehen von dem Problem der effektiven Konstruktion - aus unserer bisherigen Darstellung selbst die Existenz eines maximalen Flusses nicht klar ist; zwar wird das gegebene Netz mit den Kapazitäten als endlich vorausgesetzt, aber es wäre ja auch denkbar, dass eine Folge $f_1, f_2, \ldots f_k, \ldots$ von Flüssen existiert mit $w(f_i) < w(f_{i+1})$ für alle $i$, ohne dass eine Art "Grenzwertfluss" existiert. Es wird jedoch in Kürze gezeigt werden, dass es stets einen maximalen Fluss gibt.

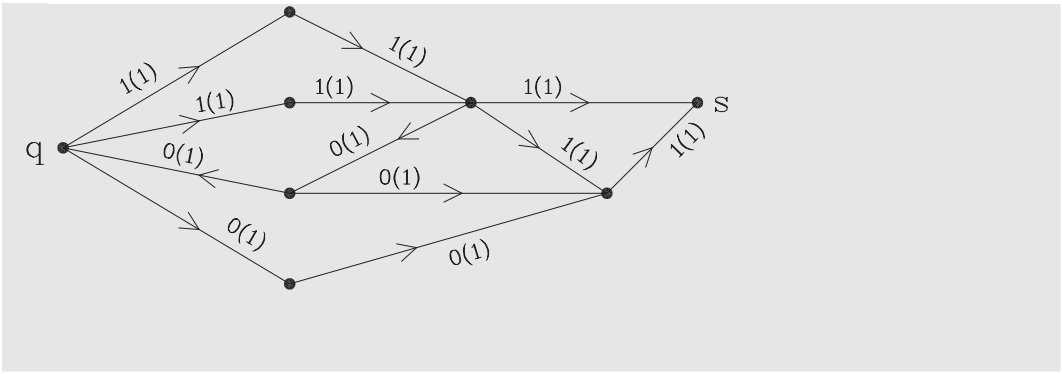
Im zweiten Beispiel von Beispiel 7.1-1 hat der Fluss auf den Kanten immer den Wert $0$ oder $1$, und es gilt $w(f_2) = 3$. Man kann daraus drei kantendisjunkte Wege (d. h. Wege ohne gemeinsame Kanten) von $q$ nach $s$ konstruieren: von $q$ ausgehend wählt man einen Weg nach $s$, der aus lauter Kanten $k$ mit $f_2(k) = 1$ besteht; dann ändert man auf diesen Kanten den Flusswert zu $0$ - so entsteht ein Fluss $f_2'$ mit $w(f_2') = 2$ - und verfährt genauso usw. In unserem Beispiel können sich so etwa die Wege $q \to b \to f \to s$, $q \to c \to e \to h \to s$ und $q \to d \to e \to g \to s$ ergeben.

Hat man umgekehrt $m$ viele kantendisjunkte Wege von einer Ecke $q$ zu einer Ecke $s$ in einem beliebigen Digraphen, so induzieren diese Wege einen Fluss $f$ mit $w(f) = m$ in dem Netz, welches entsteht, wenn allen Kanten des Digraphen die Kapazität $1$ zugeordnet wird.

**Beispiel 7.1-2:**




Im obigen Diagramm sind mit $A$ und $B$ zwei kantendisjunkte Wege von $q$ nach $s$ in dem Digraphen angegeben. Sie führen zu dem im folgenden Bild dargestellten Fluss-Netz bzw. Fluss:




Einen Fluss, der die Kanten des gegebenen Fluss-Netzes nur mit den Werten $0$ oder $1$ bewertet, nennt man **0-1-Fluss**. Der soeben skizzierte Zusammenhang zwischen $0$-$1$-Flüssen und kantendisjunkten Wegen ist bei der Frage der Zuverlässigkeit von Kommunikationsnetzen von Interesse. Gibt es nämlich in einem Digraphen $m$ kantendisjunkte Wege von einer Ecke $q$ zu einer Ecke $s$, so bedeutet dies, dass beliebige $m-1$ Verbindungen (Kanten) ausfallen können und dann immer noch ein Weg von $q$ nach $s$ (auf dem z. B. Nachrichten gesendet werden können) intakt ist. Ein Maß für die Zuverlässigkeit des Netzes (bezüglich der Wege von $q$ nach $s$) ist also die maximale Anzahl kantendisjunkter Wege, und diese entspricht - wie oben gezeigt - dem maximalen Wert eines $0$-$1$-Flusses. Wie wir bald sehen werden, kann man für ein Fluss-Netz mit identischer Kapazitätsbewertung $1$ für alle Kanten einen maximalen Fluss konstruieren, der $0$-$1$-Fluss (und damit auch maximaler $0$-$1$-Fluss) ist. Damit ist dann das geschilderte Problem der Zuverlässigkeitsanalyse gelöst.

**0-1-Fluss**

Wir werden auf diesen Zusammenhang an späterer Stelle noch einmal zurückkommen.

In Kapitel 4 wurden Zirkulationen als solche Kantenbewertungen $f$ definiert, bei denen jede Ecke $e$ sozusagen eine ausgeglichene Bilanz hat, d. h. es gilt

$$\sum_{k^+=e} f(k) = \sum_{k^-=e} f(k).$$

Bei einem Fluss auf einem Fluss-Netz fordern wir dies für die Ecken außer $q$ und $s$, und wir haben Hilfssatz 7.1-1 gezeigt, dass dann genauso viel aus $q$ heraus wie in $s$ hineinfließt. Dies bedeutet: fügen wir dem Digraphen $G = (E, K)$ eine Kante $k_{sq}$ von $s$ nach $q$ hinzu, und erweitern wir einen gegebenen Fluss $f$ auf $N = (G, c, q, s)$ zu einer Abbildung $f' : K' = K \cup \{k_{sq}\} \to \mathbb{R}_0^+$, indem wir

$$f'(k_{sq}) = w(f)$$

setzen, so ist $f'$ eine Zirkulation auf $G' = (E, K')$.

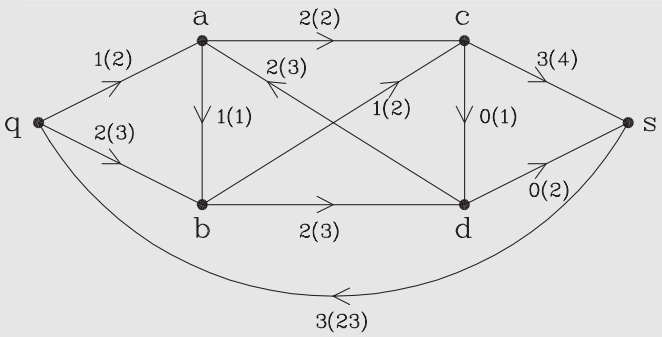
Durch den Übergang von den Flüssen zu den Zirkulationen fällt die Sonderrolle der Ecken $q$ und $s$ weg, wodurch die mathematische Behandlung oft etwas einfacher und durchsichtiger wird. Auch darauf wird an späterer Stelle noch einmal eingegangen.

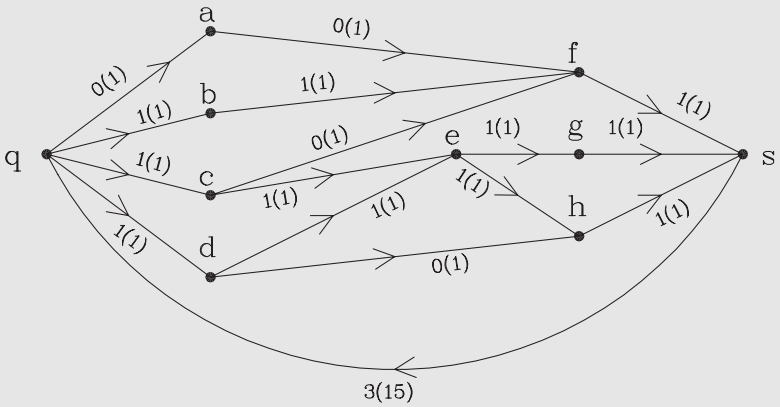
Eine Zirkulation, die Kapazitätsbeschränkungen der Kanten respektiert, wird **zulässige Zirkulation** genannt. Beim soeben beschriebenen Übergang von einem Fluss

**zulässige Zirkulation**

zu einer Zirkulation setzt man $c'(k_{sq})$ groß genug an, um eine zulässige Zirkulation zu erhalten (z. B. $c'(k_{sq}) := \sum_{k \in K} |c(k)|$).

**Beispiel 7.1-3:**
Die folgenden beiden Diagramme zeigen zu den in Beispiel 7.1-1 dargestellten Flüssen die erweiterten Digraphen mit den zulässigen Zirkulationen:




**Selbsttestaufgabe 7.1-1:**

*Erläutern Sie den Zusammenhang zwischen Flüssen und Zirkulationen.*

## 7.2 Die Sätze von Ford und Fulkerson

**Definition 7.2-1:** *Es sei $N = (G, c, q, s)$ ein Fluss-Netz. Der Wert $w(f)$ eines Flusses $f$ kann offenbar nicht größer sein als die Summe der Kapazitäten der Kanten, die aus $q$ hinausführen. Dies ist nur ein Spezialfall der folgenden Situation. $[Q, S]$ sei ein Schnitt von $G = (E, K)$ mit $q \in Q$ und $s \in S$. Einen solchen Schnitt nennt man einen **Schnitt des Fluss-Netzes** $N$. Die **Kapazität** des Schnittes $[Q, S]$ ist die Zahl*

**Schnitt eines Fluss-Netzes**
**Kapazität eines Schnittes**

$$c(Q, S) = \sum_{\substack{k^- \in Q \\ k^+ \in S}} c(k).$$

*Der Schnitt $[Q, S]$ heißt **minimal**, wenn er die kleinstmögliche Kapazität hat, d. h. wenn gilt $c(Q, S) \leq c(Q', S')$ für jeden Schnitt $[Q', S']$ von $N$.*

**minimaler Schnitt**

Der folgende Satz ist von grundsätzlicher Wichtigkeit. Er besagt insbesondere, dass die Kapazität eines minimalen Schnitts eine obere Schranke für den Wert eines maximalen Flusses ist.

**Hilfssatz 7.2-1:** *$N = (G, c, q, s)$ sei ein Fluss-Netz, $[Q, S]$ ein Schnitt und $f$ ein Fluss. Dann gilt*

$$w(f) \leq c(Q, S).$$

**Beweis:** Nach Definition ist

$$w(f) = \sum_{k^-=q} f(k) - \sum_{k^+=q} f(k).$$

Da für alle Ecken $e \in Q$ mit $e \neq q$ gilt

$$\sum_{k^-=e} f(k) - \sum_{k^+=e} f(k) = 0,$$

können wir auch schreiben

$$w(f) = \sum_{e \in Q} \left( \sum_{k^-=e} f(k) - \sum_{k^+=e} f(k) \right).$$

Da die Kanten, die innerhalb von $Q$ verlaufen, bei dieser Summation gerade herausfallen, denn ihr Flusswert wird einmal positiv und einmal negativ mitgezählt, ergibt sich

$$w(f) = \sum_{\substack{k^- \in Q \\ k^+ \in S}} f(k) - \sum_{\substack{k^+ \in Q \\ k^- \in S}} f(k).$$

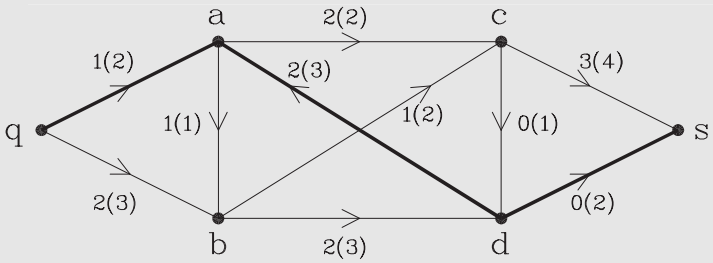
Wegen der Ungleichungen $f(k) \leq c(k)$ für die Kanten $k$ mit $k^- \in Q$ und $k^+ \in S$ und $f(k) \geq 0$ für die Kanten $k$ mit $k^+ \in Q$ und $k^- \in S$ folgt schließlich $w(f) \leq c(Q, S)$.

Es wird sich nun herausstellen, dass der Wert eines maximalen Flusses sogar gleich der Kapazität eines minimalen Schnittes ist. Um dies zeigen zu können, wird der Begriff des zunehmenden Weges benötigt.

zunehmender Weg

**Definition 7.2-2:** *$N = (G, c, q, s)$ sei ein Fluss-Netz, $f$ ein Fluss. Ein ungerichteter Weg $W$ von $q$ nach $s$ heißt **zunehmender Weg** (bezüglich $f$), wenn für jede Vorwärtskante $k$ auf diesem Weg $f(k) < c(k)$ und für jede Rückwärtskante $k'$ auf diesem Weg $f(k') > 0$ gilt.*

**Beispiel 7.2-1:**
Der ungerichtete Weg $q - a - d - s$ im ersten Beispiel aus Beispiel 7.1-1 ist ein zunehmender Weg:




Man kann den Wert des Flusses hier offenbar erhöhen, indem man die $f$-Werte auf $qa$ und $ds$ um 1 erhöht und auf $da$ um 1 erniedrigt.

Allgemein gilt der

**Satz 7.2-1:** *Ein Fluss $f$ auf einem Fluss-Netz $N$ ist genau dann maximal, wenn es keinen zunehmenden Weg (bezüglich $f$) gibt.*

**Beweis:** $f$ sei ein maximaler Fluss. Wir nehmen an, es gebe einen zunehmenden Weg $W$. Wir betrachten nun auf diesem Weg die Zahlen $f(k) - c(k)$ für die Vorwärtskanten und $f(k')$ für die Rückwärtskanten, $w_{min}$ sei die kleinste aller dieser Zahlen. Nach Definition eines zunehmenden Weges folgt $w_{min} > 0$. Wir erklären nun einen neuen Fluss $f' : K \to \mathbb{R}_0^+$, indem wir auf dem Weg $W$ die $f$-Werte verändern:

$$f'(k) := \begin{cases} f(k) + w_{min}, & \text{falls } k \text{ Vorwärtskante von } W \text{ ist} \\ f(k) - w_{min}, & \text{falls } k \text{ Rückwärtskante von } W \text{ ist} \\ f(k), & \text{sonst.} \end{cases}$$

Es ist leicht zu sehen, dass $f'$ ein Fluss ist mit Wert $w(f') = w(f) + w_{min} > w(f)$, und dies widerspricht der Maximalität von $f$. Folglich war die Annahme der Existenz eines zunehmenden Weges falsch.

Nun sei umgekehrt vorausgesetzt, dass es keinen zunehmenden Weg von $q$ nach $s$ gibt. $Q$ sei die Menge aller Punkte $e$, für die es einen zunehmenden Weg von $q$ nach $e$ gibt (dabei ist $q$ eingeschlossen), und $S := E \backslash Q$. $[Q, S]$ ist wegen $s \notin Q$ bzw. $s \in S$ ein Schnitt von $N$. Es folgt, dass für jede Kante $k$ mit $k^- \in Q$ und $k^+ \in S$ gelten muss $f(k) = c(k)$, denn im Falle $f(k) < c(k)$ würde es einen zunehmenden Weg von $q$ nach $k^+$ geben im Widerspruch zur

Definition von $Q$ bzw. $S$. Genauso folgt, dass für jede Kante $k$ mit $k^- \in S$ und $k^+ \in Q$ gelten muss $f(k) = 0$. Wegen

$$w(f) = \sum_{\substack{k^- \in Q \\ k^+ \in S}} f(k) - \sum_{\substack{k^+ \in Q \\ k^- \in S}} f(k)$$

(vgl. Beweis zu Hilfssatz 7.2-1 ) hat man $w(f) = c(Q, S)$. Daraus folgt, dass $f$ maximal ist, denn für einen Fluss $f'$ mit $w(f) < w(f')$ würde mit dem hier betrachteten Schnitt $[Q, S]$ gelten $c(Q, S) < w(f')$ im Widerspruch zur Aussage von Hilfssatz 7.2-1.

Dem letzten Schluss im obigem Beweis liegt das folgende allgemeinere Prinzip zugrunde: Es seien $A$ und $B$ Mengen reeller Zahlen, wobei in $A$ ein größtes Element $maxA$ und in $B$ ein kleinstes Element $minB$ existieren. Ferner gelte $a \leq b$ für alle $a \in A$ und $b \in B$. Gibt es nun ein $A$ und $B$ gemeinsames Element $x$ (also $x \in A \cap B$), so folgt $A \cap B = \{x\}$ und $x = maxA = minB$. In Worten heißt dies: liegt $A$ vollständig "links" von $B$, und überschneiden sich $A$ und $B$, so haben sie genau einen Punkt gemeinsam, der das Maximum von $A$ und das Minimum von $B$ ist. Auf diesem Prinzip und Verallgemeinerungen davon beruhen eine Reihe von Sätzen in der Mathematik, man spricht hier auch von **min-max-Sätzen**.

**min-max-Sätze**

Aus dem Beweis von Satz 7.2-1 lässt sich unmittelbar eine Folgerung ableiten. Dabei wird für einen Fluss $f$ auf einem Fluss-Netz $N = (G, c, q, s)$ die folgende Bezeichnung verwendet: $Q_f$ sei die Menge aller von $q$ aus auf einem zunehmenden Weg erreichbaren Ecken (einschließlich $q$) und $S_f = E \backslash Q_f$.

**Folgerung 7.2-1:** *Für einen Fluss $f$ auf einem Fluss-Netz $N$ sind die folgenden Bedingungen äquivalent:*

i. $f$ ist maximal.

ii. Es gilt $s \in S_f$.

iii. $[Q_f, S_f]$ ist ein Schnitt von $N$ mit $w(f) = c(Q_f, S_f)$.

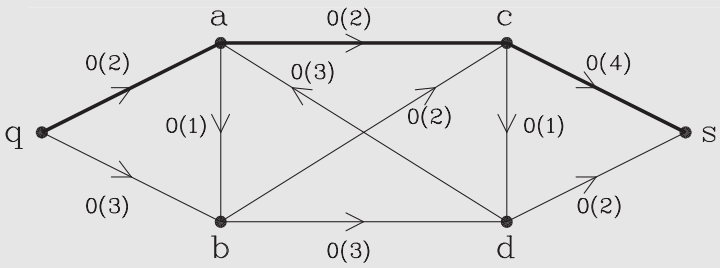
Bereits im vorigen Abschnitt wurde das Problem der Existenz eines maximalen Flusses angesprochen, welches nun angegangen werden soll. Die bisherigen Ergebnisse liefern nur Aussagen, unter welchen Bedingungen ein gegebener Fluss maximal ist. Satz 7.2-1 legt es nahe, einen gegebenen Fluss mit Hilfe zunehmender Wege so lange zu vergrößern, bis kein zunehmender Weg mehr existiert - der erhaltene Fluss muss dann maximal sein. Die entscheidende Frage ist aber, wieso man bei dieser Vorgehensweise nach endlich vielen Schritten in die Situation kommt, dass kein zunehmender Weg mehr existiert. Ein erstes wichtiges Ergebnis in der gewünschten Richtung liefert der folgende Satz:

**Satz 7.2-2:** *$N = (G, c, q, s)$ sei ein Netz, bei dem alle Kapazitäten $c(k)$ ganzzahlig sind. Dann gibt es einen maximalen Fluss $f$ auf $N$ derart, dass alle Werte $f(k)$ ganzzahlig sind.*

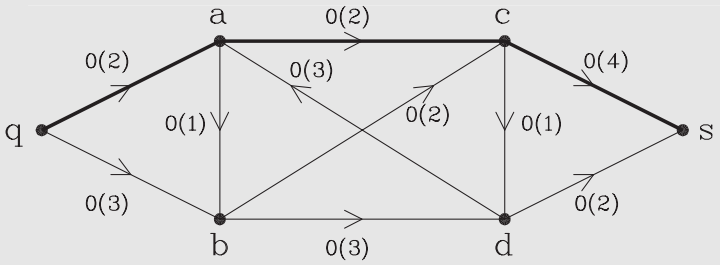
**Beweis:** Wir zeigen, wie ein solcher maximaler Fluss in endlich vielen Schritten mit Hilfe zunehmender Wege konstruiert werden kann.

Zunächst wird durch $f_0(k) := 0$ für alle $k$ ein Fluss mit Wert $0$ definiert. Ist $f_0$ nicht maximal, so muss es nach Satz 7.2-1 einen zunehmenden Weg $W_0$ (bezüglich $f_0$) geben. Die im Beweis von Satz 7.2-1 vorkommende Zahl $w_{min}$ ist jetzt eine positive ganze Zahl, die - wie dort dargestellt - Anlass gibt zur Vergrößerung des Flusses zu einem Fluss $f_1$ mit $w(f_1) = w_{min}$, wobei alle Werte $f_1(k)(k \in K)$ ganzzahlig sind. Ist $f_1$ nicht maximal, so muss wieder ein zunehmender Weg $W_1$ existieren usw. Da bei diesem Verfahren in jedem Schritt der Wert des Flusses um eine positive ganze Zahl erhöht wird, muss nach endlich vielen Schritten ein Fluss $f$ erhalten werden, für den es keinen zunehmenden Weg mehr gibt. $f$ ist dann ein maximaler Fluss der gewünschten Art.

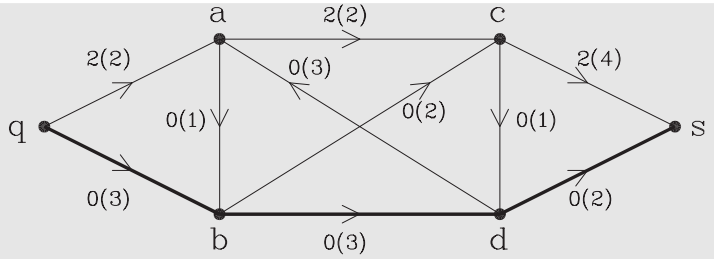
**Beispiel 7.2-2:**



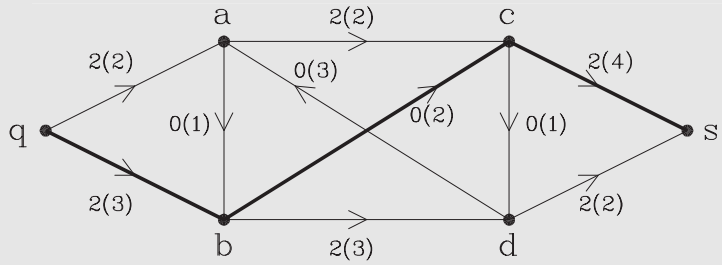

$f_0$ mit zunehmendem Weg $W_0, w_{min} = 2$,




$f_1$ mit zunehmendem Weg $W_1, w_{min} = 2$,




$f_2$ mit zunehmendem Weg $W_2, w_{min} = 1$,




$f_3$ ist maximal, es gibt keinen zunehmenden Weg (von $q$ nach $s$) bezüglich $f_3$. Es gilt sogar $Q_f = \{q\}$ (s. Folgerung 7.2-1).

Wir können mit dem soeben bewiesenen Satz an die obigen Bemerkungen zu $0$-1-Flüssen anknüpfen und erhalten:

**Folgerung 7.2-2:** *$N = (G, c, q, s)$ sei ein Fluss-Netz derart, dass $c(k)$ für jede Kante $k$ den Wert $0$ oder $1$ hat. Dann gibt es einen maximalen Fluss, der $0$-1-Fluss ist.*

Der obige Satz ist auch die Basis für das im folgenden formulierte zentrale Ergebnis zu Flüssen in Netzen. Locker gesprochen, besagt das Ergebnis: ”maximaler Fluss $=$ minimaler Schnitt”. Im Englischen spricht man vom ”max-flow min-cut theorem”.

**Satz 7.2-3:** *Ist $N$ ein Fluss-Netz, so ist der maximale Wert eines Flusses auf $N$ gleich der minimalen Kapazität eines Schnittes in $N$.*

**Beweis:** Sind alle Kapazitäten ganzzahlig, so folgt die Aussage des Satzes aus Satz 7.2-2 und Folgerung 7.2-1. Sind alle Kapazitäten rationale Zahlen (also Brüche), so kann das Problem durch Multiplikation aller vorkommenden Zahlen mit ihrem Hauptnenner in ein äquivalentes Problem umgeformt werden, bei dem alle Kapazitäten ganzzahlig sind, d. h. auch für diesen Fall ist somit der Satz bewiesen. Es bleibt der Beweis zu führen für den Fall, dass beliebige reelle (also auch irrationale) Kapazitäten auftreten. Ein mathematischer Beweis könnte hier z. B. Stetigkeitsargumente verwenden und wäre sehr anspruchsvoll. Wir wollen jedoch den im nächsten Abschnitt behandelten Satz von Edmonds und Karp

abwarten, mit dem ebenfalls die hier noch bestehende Lücke geschlossen werden kann.

Obiger Satz wurde im Jahre 1956 von L.R. Ford und D.R. Fulkerson bewiesen (s. auch deren im Literaturverzeichnis aufgeführtes Buch). Die noch bestehende Beweislücke spielt in der Praxis keine Rolle, da in einem Rechner ohnehin nur rationale Zahlen dargestellt werden können. Der im nächsten Abschnitt behandelte Satz von Edmonds und Karp hat außer dem Schließen der Beweislücke auch die Funktion, den Algorithmus zur Bestimmung eines maximalen Flusses zu verbessern, dem wir uns als nächstes zuwenden.

Die Beweise von Satz 7.2-2 und Satz 7.2-3 stellen sicher, dass man bei vorgegebenem Fluss-Netz $N$ durch wiederholte Flussvergrößerung mit Hilfe zunehmender Wege nach endlich vielen Schritten zu einem maximalen Fluss gelangt. Will man dies als Algorithmus formulieren, so muss man sich allerdings darüber Gedanken machen, wie man am geschicktesten jeweils einen zunehmenden Weg findet. Die Grundidee des Algorithmus von Ford und Fulkerson ist es, von $q$ ausgehend schrittweise diejenigen Ecken zu markieren, zu denen es (von $q$ aus) einen zunehmenden Weg gibt. Wird $s$ markiert, so kann der vorliegende Fluss vergrößert werden. Die Markierung der Ecken wird so vorgenommen, dass (falls $s$ markiert wurde) ein zunehmender Weg von $s$ aus zurückverfolgt und die zugehörige Zahl $w_{min}$, die zur Abänderung der Werte $f(k)$ benötigt wird, direkt an der Markierung von $s$ abgelesen werden kann.

**Markierungsalgorithmus von Ford und Fulkerson**

### Markierungsalgorithmus von Ford und Fulkerson

Gegeben sei ein Fluss-Netz $N = (G, c, q, s)$ mit ganzzahliger (oder rationaler) Kapazitätsfunktion $c$. Es wird ein maximaler Fluss $f$ und der zugehörige minimale Schnitt $[Q_f, S_f]$ mit $w(f) = c(Q_f, S_f)$ bestimmt.

Die Funktion $u$ mit den Werten $0$ oder $1$ dient dazu festzuhalten, welche markierten Ecken bereits abgesucht wurden; das Absuchen einer Ecke besteht darin, ihre noch nicht markierten Nachbarn zu markieren.

**Algorithmus**: Markierungsalgorithmus von Ford und Fulkerson

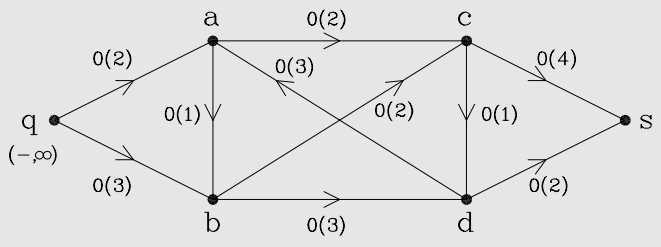
```
begin
   for k ∈ K do setze f(k) := 0;
   markiere q mit (− , ∞ );
   for e ∈ E do setze u(e) := 0, d(e) := ∞;
   repeat
     wähle eine markierte Ecke e mit u(e) = 0; *
     for k ∈ { k ∈ K | k⁻ = e } do if g = k⁺ ist nicht markiert
                   und f(k) < c(k)
      then setze d(g) := min { c(k) − f(k), d(e) }, und
        markiere g mit (e, +, d(g));
     for k ∈ { k ∈ K | k⁺ = e } do
      if g = k⁻ ist nicht markiert und f(k) > 0
      then setze d(g) := min { f(k), d(e) }, und
        markiere g mit (e, −, d(g));
     setze u(e) := 1;
     if s ist markiert
```

```
      then
       d sei die letzte Komponente der Markierung von
       s;
       setze e := s;
       while e ≠ q do
         begin
         finde die erste Komponente g der Markierung
          von e;
         if die zweite Komponente der Markierung
         von e ist +
         then setze f(k) := f(k) + d für die Kante k = ge
         else setze f(k) := f(k) − d für die Kante
         k = eg;
         setze e := g;
         end
       lösche alle Markierungen, außer der von q;
       for e ∈ E do setze d(e) := ∞ und u(e) := 0
   until u(e) = 1 für alle markierten Ecken e;
   Q_f sei die Menge der markierten Ecken und
   S_f := E \ Q_f;
end
```

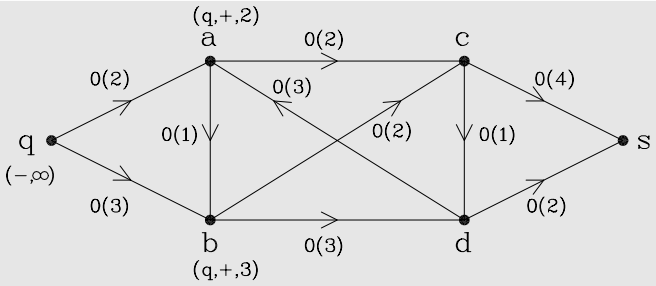
Die Korrektheit dieses Algorithmus ergibt sich, wie bereits gesagt, aus den vorangegangenen Sätzen und ihren Beweisen. Diese Korrektheitsaussage, dass also der Markierungsalgorithmus einen maximalen Fluss und einen minimalen Schnitt bestimmt, wird oft auch als eigenständiger Satz formuliert, und zwar ebenfalls unter dem Namen **Satz von Ford und Fulkerson**.

**Satz von Ford und Fulkerson**

**Beispiel 7.2-3:**
Wir wollen den Algorithmus an dem bereits mehrfach betrachteten Beispiel illustrieren. Nach der Initialisierung hat man Flusswert $0$ auf allen Kanten, und $q$ ist mit $(-, \infty)$ markiert. Neben den Diagrammen sind die $u$- und $d$-Werte dargestellt.

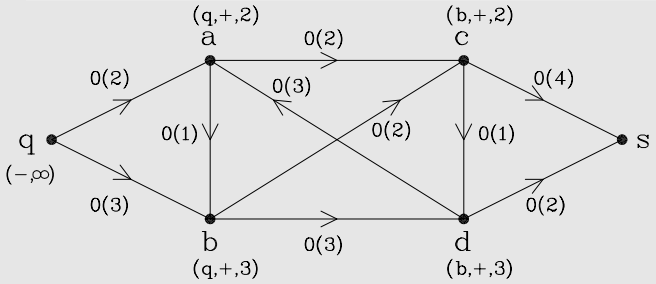



| | $u$ | $d$ |
|---|---|---|
| $q$ | $0$ | $\infty$ |
| $a$ | $0$ | $\infty$ |
| $b$ | $0$ | $\infty$ |
| $c$ | $0$ | $\infty$ |
| $d$ | $0$ | $\infty$ |
| $s$ | $0$ | $\infty$ |




| | $u$ | $d$ |
|---|---|---|
| $q$ | 1 | $\infty$ |
| $a$ | 0 | 2 |
| $b$ | 0 | 3 |
| $c$ | 0 | $\infty$ |
| $d$ | 0 | $\infty$ |
| $s$ | 0 | $\infty$ |

$s$ ist noch nicht markiert. Da nicht alle markierten Ecken abgesucht wurden (bzw. $u$-Wert 1 haben), kann in Zeile ***** nun z. B. die Ecke $b$ gewählt werden. Dies führt zur Markierung von $c$ und $d$:

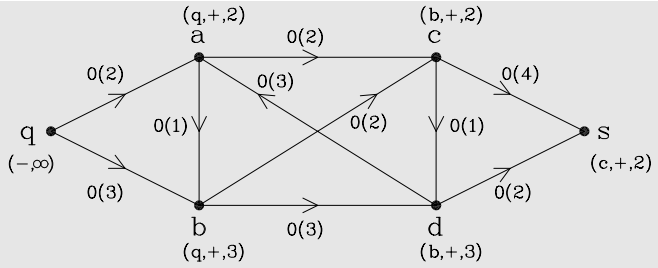



| | $u$ | $d$ |
|---|---|---|
| $q$ | 1 | $\infty$ |
| $a$ | 0 | 2 |
| $b$ | 1 | 3 |
| $c$ | 0 | 2 |
| $d$ | 0 | 3 |
| $s$ | 0 | $\infty$ |

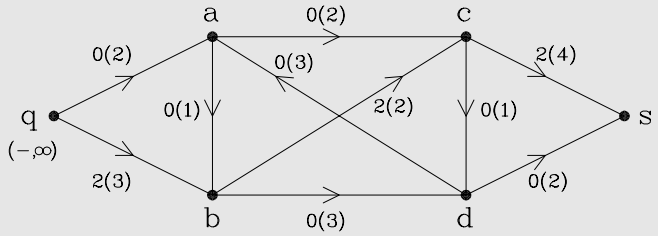
$s$ ist noch nicht markiert. Markierte, noch nicht abgesuchte Ecken sind $a, c$ und $d$. Wird nun in Zeile ***** die Ecke $a$ gewählt, so führt dies zu keinen neuen Markierungen; der einzige Effekt ist die änderung von $u(a)$:

| | $u$ | $d$ |
|---|---|---|
| $q$ | 1 | $\infty$ |
| $a$ | 1 | 2 |
| $b$ | 1 | 3 |
| $c$ | 0 | 2 |
| $d$ | 0 | 3 |
| $s$ | 0 | $\infty$ |

Beim nächsten Schritt wählen wir $c$ als markierte Ecke mit $u(c) = 0$; dies führt zu folgendem Diagramm:

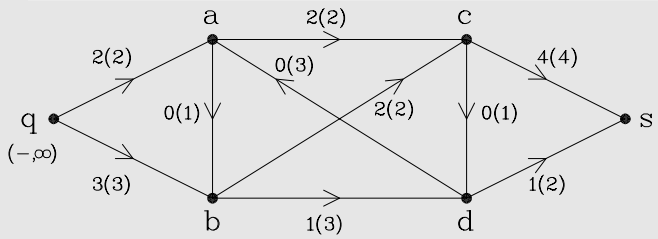



Nun wurde $s$ markiert. Dies führt zur Flussvergrößerung entlang des Weges $q \to b \to c \to s$, der von $s$ aus rückwärts ermittelt wird. Mit dem neuen Fluss wird wieder der Beginnzustand hergestellt (Löschen der Markierungen usw.), man hat die inm folgenden Diagramm dargestellte Situation:




| | $u$ | $d$ |
|---|---|---|
| $q$ | $0$ | $\infty$ |
| $a$ | $0$ | $\infty$ |
| $b$ | $0$ | $\infty$ |
| $c$ | $0$ | $\infty$ |
| $d$ | $0$ | $\infty$ |

Nun gibt es wieder eine markierte Ecke mit $u$-Wert $0$, nämlich $q$, und der Markierungs- und Absuchprozess startet von neuem. Wir wollen die Beschreibung des Beispiels hier abkürzen und stellen im nächsten Diagramm eine Situation dar, wie sie sich nach zwei Flussvergrößerungen (bei entsprechender Wahl in Zeile ***** des Algorithmus) darstellen kann:




| | $u$ | $d$ |
|---|---|---|
| $q$ | $0$ | $\infty$ |
| $a$ | $0$ | $\infty$ |
| $b$ | $0$ | $\infty$ |
| $c$ | $0$ | $\infty$ |
| $d$ | $0$ | $\infty$ |

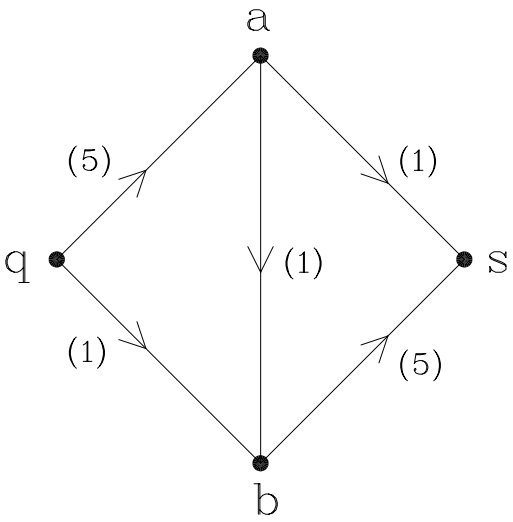
Nun kann in Zeile ***** wieder nur $q$ gewählt werden, was jedoch nicht zu weiteren Markierungen führt. Da $u(q) = 1$ gesetzt wird, ist die until-Bedingung erfüllt, die zum Abbruch der repeat-Schleife führt, und es wird $Q_f = \{q\}$ und $S_f = \{a, b, c, d, s\}$ gesetzt. Der vorliegende Fluss mit Wert $5$ ist maximal, $[Q_f, S_f]$ ist ein minimaler Schnitt mit $c(Q_f, S_f) = 5$.

In dem soeben behandelten Beispiel ist deutlich geworden, dass die getroffene Auswahl einer markierten, noch nicht abgesuchten Ecke (Zeile ***** im Algorithmus) den weiteren Ablauf des Algorithmus - etwa die Anzahl der noch auszuführenden Iterationen - beeinflusst. In der Tat kann man an Beispielen aufzeigen, dass der Algorithmus in der bisher beschriebenen Form, also mit dem Freiheitsgrad in Zeile *****, nicht polynomial ist. Bei einer Kapazitätsfunktion, die auch irrationale Werte annimmt, muss das Verfahren weder abbrechen noch gegen einen maximalen Fluss konvergieren - ein Beispiel hierfür ist in dem Buch von Ford und Fulkerson zu finden. All diese Probleme werden wir im nächsten Abschnitt lösen, wo die Zeile ***** des Algorithmus durch eine präzisere Anweisung ersetzt wird.

Zum Markierungsalgorithmus wollen wir schließlich noch anmerken, dass selbstverständlich nicht unbedingt - wie in unserer Formulierung - mit dem $0$-Fluss (d. h. $f(k) = 0$ für jede Kante $k$) begonnen werden muss. Stattdessen kann mit einem beliebigen Fluss $f$ initialisiert und dann der identische Algorithmus angewendet werden.

**Selbsttestaufgabe 7.2-1:**

*Gegeben sei das folgende Fluss-Netz $N$:*




*Bestimmen Sie den maximalen Fluss-Wert auf $N$ durch Betrachtung aller Schnitte und Anwendung des "max-flow min-cut"-Theorems. Warum führt diese Vorgehensweise nicht zu einem guten Algorithmus?*

## 7.3 Der Satz von Edmonds und Karp

Wir betrachten nun die folgende Modifikation des Markierungsalgorithmus von Ford und Fulkerson: die Zeile ***** des Algorithmus wird durch den Schritt

$e$ sei unter allen Ecken mit $u(e) = 0$ die am frühesten markierte Ecke; ******

ersetzt. Diese Modifikation wurde von J. Edmonds und R.M. Karp 1972 in einer Arbeit vorgeschlagen, der so modifizierte Algorithmus wird als **Algorithmus von Edmonds und Karp** bezeichnet.

**Algorithmus von Edmonds und Karp**

Es liegt auf der Hand, dass bei dieser Variante des Algorithmus immer ein zunehmender Weg mit möglichst wenig Kanten bestimmt und zur Flussvergrößerung verwendet wird. Der Grund ist, dass wie beim "Breadth first search"-Verfahren stets zunächst dort weitergesucht wird, wo man zuerst markiert hat.

**Satz von Edmonds und Karp** Der Algorithmus von Edmonds und Karp bestimmt einen maximalen Fluss $f$ und einen minimalen Schnitt $[Q_f, S_f]$ für ein beliebiges Fluss-Netz $N$ mit reeller Kapazitätsfunktion $c$. Der Algorithmus hat Komplexität $0(|E| \cdot |K|^2)$.

**Beweis:** Der Algorithmus werde auf ein beliebiges Fluss-Netz $N = (G, c, q, s)$ mit reeller Kapazitätsfunktion $c$ angewandt. Der Algorithmus geht aus von dem Null-Fluss $f_0$, die weiteren dann erhaltenen Flüsse bezeichnen wir mit $f_1, f_2, \ldots, f_i, \ldots$. $x_e(i)$ sei die kürzeste Länge (hinsichtlich Kantenanzahl) eines zunehmenden Weges von $q$ nach $e$ bezüglich $f_i$. Entscheidend für den Beweis ist nun die folgende Behauptung, die wir zuerst zeigen wollen: es gilt

$$x_e(i+1) \geq x_e(i) \text{ für alle } i \text{ und } e.$$

Angenommen, es gilt $x_e(i+1) < x_e(i)$ für eine Ecke $e$ und eine Zahl $i$. Wir können o.B.d.A. auch voraussetzen, dass $x_e(i+1)$ minimal ist unter allen Zahlen der Form $x_g(j+1)$, die obige Ungleichung verletzen (d. h. für die $x_g(j+1) < x_g(j)$ gilt). $k$ sei die letzte Kante in einem kürzesten zunehmenden Weg von $q$ nach $e$ bezüglich $f_{i+1}$. $k$ sei eine Vorwärtskante, also $k = he$ für eine Ecke $h$; es muss nun $f_{i+1}(k) < c(k)$ sein. Es gilt $x_e(i+1) = x_h(i+1) + 1$, und nach den Voraussetzungen über $e$ ist $x_h(i+1) \geq x_h(i)$, womit sich insgesamt die Beziehung $x_e(i+1) \geq x_h(i)+1$ ergibt. Es ist nun aber $f_i(k) = c(k)$, denn sonst müsste $x_e(i) \leq x_h(i) + 1$ und somit doch $x_e(i+1) \geq x_e(i)$ sein. Wegen $f_{i+1}(k) < c(k)$ bedeutet dies, dass die Kante $k$ beim Schritt von $f_i$ nach $f_{i+1}$ als Rückwärtskante verwendet worden sein muss. Da der gewählte zunehmende Weg bezüglich $f_i$ kürzeste Länge hatte, ergibt sich daraus $x_h(i) = x_e(i) + 1$ und $x_e(i+1) \geq x_e(i) + 2$ im Widerspruch zu unseren Voraussetzungen.

In ähnlicher Weise verläuft der Beweis, wenn die Kante $k$ eine Rückwärtskante ist. Damit ist die Allgemeingültigkeit der Ungleichung $x_e(i+1) \geq x_e(i)$ gezeigt.

Analog wird die Ungleichung

$$y_e(i+1) \geq y_e(i)$$

bewiesen, wobei $y_e(i)$ die Länge eines kürzesten zunehmenden Weges von $e$ nach $s$ bezüglich $f_i$ ist.

Aus all diesen abgeleiteten Ungleichungen lässt sich nun eine obere Schranke für die Anzahl der durch den Algorithmus vorgenommenen Flussvergrößerungen herleiten. Dazu beobachten wir zunächst, dass bei jeder änderung des Flusses mindestens eine Kante des zunehmenden Weges "kritisch" ist in dem Sinne, dass der Fluss durch diese Kante entweder bis zur Kapazität erhöht oder auf $0$ gesenkt wird.

Es sei $k = he$ eine kritische Kante im zunehmenden Weg bezüglich $f_i$; dieser Weg besteht aus $x_e(i) + y_e(i) = x_h(i) + y_h(i)$ vielen Kanten. Wenn $k$ beim nächsten Mal (etwa bezüglich $f_j$) in einem zunehmenden Weg verwendet wird, muss $k$ in umgekehrter Richtung verwendet werden: war es für $f_i$ eine Vorwärtskante, so ist es für $f_j$ eine Rückwärtskante (und umgekehrt). Angenommen, $k$ war für $f_i$ eine Vorwärtskante. Wir schließen $x_e(i) = x_h(i) + 1$ und $x_h(j) = x_e(j)+1$. Aus den gezeigten Ungleichungen folgt $x_e(j) \geq x_e(i)$ und $y_h(j) \geq y_h(i)$ und somit

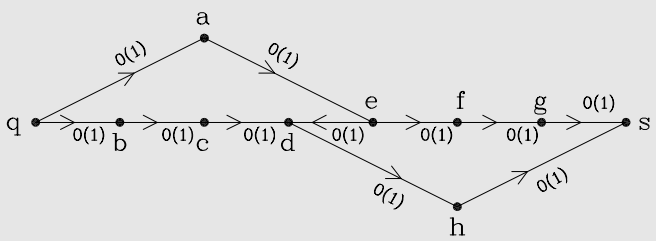
$$x_h(j)+y_h(j) = x_e(j)+1+y_h(j) \geq x_e(i)+1+y_h(i) = x_h(i)+y_h(i)+2.$$

Der zunehmende Weg bezüglich $f_j$ ist also um mindestens zwei Kanten länger als der zunehmende Weg bezüglich $f_j$. Ein analoger Schluss lässt sich natürlich ziehen, wenn $k$ bezüglich $f_i$ eine Rückwärtskante war.

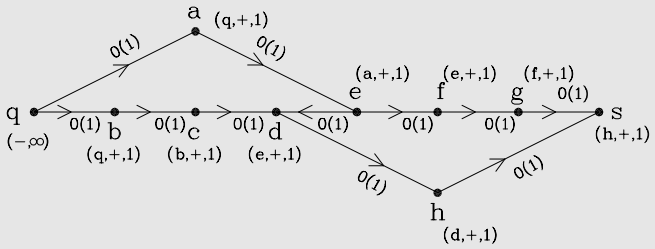
Da ein zunehmender Weg nicht mehr als $|E| - 1$ viele Kanten enthalten kann, kann folglich jede Kante in höchstens $\frac{|E|-1}{2}$ vielen Flussvergrößerungen kritisch sein, was insgesamt bedeutet: der Fluss kann in dem Algorithmus höchstens $0(|E| \cdot |K|)$-mal vergrößert werden. Der Algorithmus muss also abbrechen, auch bei irrationalen Kapazitätswerten.

Da das Auffinden eines zunehmenden Weges und die entsprechende Flussvergrößerung jeweils $0(|K|)$ viele Schritte erfordern, ergibt sich insgesamt eine Komplexität von $0(|E| \cdot |K|^2)$.

**Beispiel 7.3-1:**
Wir wollen den Algorithmus von Edmonds und Karp an folgendem Beispiel illustrieren, wo der $0$-Fluss und die Kapazitäten der Kanten im Diagramm eingetragen sind:

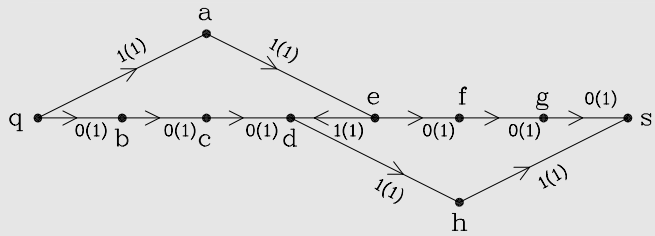


Damit bei der Bestimmung eines zunehmenden Weges die neue Zeile **** des Algorithmus ausgeführt werden kann, muss die Reihenfolge der Markierungen der Ecken festgehalten werden. Am einfachsten realisiert man das (auch bei einer Implementierung) mit Hilfe einer Warteschlange. Bei der Behandlung des Beispiels werden wir (neben dem Diagramm) die markierten Ecken untereinanderschreiben und aus dieser Spalte von oben jeweils die nächste abzusuchende Ecke bestimmen - nach dem Absuchen (d. h. der Markierung der Nachbarn) wird die Ecke ”abgehakt”. So entsteht beim ersten Schritt folgendes Bild:

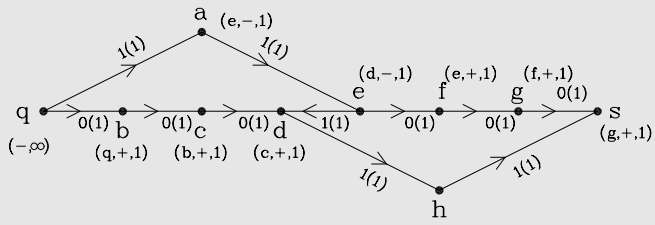



| | |
|---|---|
| $q$ | $\surd$ |
| $a$ | $\surd$ |
| $b$ | $\surd$ |
| $e$ | $\surd$ |
| $c$ | $\surd$ |
| $d$ | $\surd$ |
| $f$ | $\surd$ |
| $h$ | $\surd$ |
| $g$ | |
| $s$ | |

Der zunehmende Weg ist $q \to a \to e \to d \to h \to s$, man kommt zum hier dargestellten Fluss:

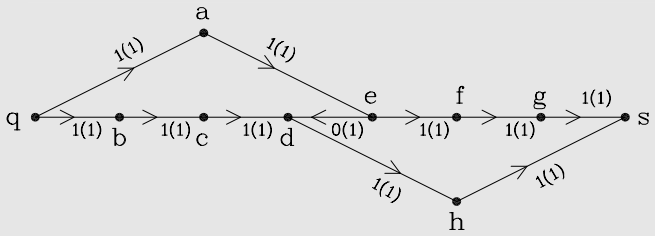



Beim zweiten Schritt bekommt man folgendes Bild:




| | |
|---|---|
| $q$ | $\surd$ |
| $b$ | $\surd$ |
| $c$ | $\surd$ |
| $d$ | $\surd$ |
| $e$ | $\surd$ |
| $f$ | $\surd$ |
| $a$ | $\surd$ |
| $g$ | $\surd$ |
| $s$ | |

Der ermittelte zunehmende Weg ist $q \to b \to c \to d \leftarrow e \to f \to g \to s$, man kommt zu folgendem Fluss: $f_2 = f$:




Der Fluss ist maximal: der Algorithmus stoppt beim nächsten Schritt, und man hat $Q_f = \{q\}$, $S_f = \{a, b, \ldots, h, s\}$ mit $c(Q_f, S_f) = 2$ und $w(f) = 2$.

Nach dem Satz und dem Algorithmus von Edmonds und Karp wollen wir das Problem der Bestimmung maximaler Flüsse vorerst nicht weiter behandeln. Auf Varianten bzw. andere Aspekte dieses Problems wird in den nächsten Abschnitten eingegangen.

Zum Abschluss sollte jedoch noch erwähnt werden, dass neben den Algorithmen von Ford und Fulkerson bzw. Edmonds und Karp weitere Verfahren untersucht wurden, die von der Komplexität her zwar schlechter dastehen, in konkreten Fällen (mit nicht zu großen Netzen) jedoch mitunter schneller zum Ziel kommen; z. B. besteht ein solches Verfahren darin, stets einen zunehmenden Weg zu suchen, der zu einer größtmöglichen Erhöhung des Flusses führt. Näher Interessierte müssen wir hier auf die umfangreiche Literatur zu diesem Thema verweisen.

**Selbsttestaufgabe 7.3-1:**

*Erläutern Sie die Vorteile des Algorithmus von Edmonds und Karp gegenüber dem Algorithmus von Ford und Fulkerson.*

## 7.4 Eine kombinatorische Anwendung: Der Satz von Menger

Bereits im ersten Abschnitt wurden kantendisjunkte Wege zwischen zwei Ecken in einem Digraphen betrachtet, und es wurde auf den Zusammenhang mit 0-1-Flüssen hingewiesen. Wir wollen nun einen $min$-$max$-Satz über die Anzahl kantendisjunkter Wege beweisen, der sich auf das "max-flow min-cut theorem" Satz 7.2-3 stützt.

**trennende Kantenmenge**

Dabei heißt eine Menge $A \subseteq K$ von Kanten in einem Digraphen $(E, K)$ eine $q$ und $s$ **trennende Kantenmenge** ($q$, $s$ sind beliebige Ecken), wenn jeder gerichtete Weg von $q$ nach $s$ eine Kante aus $A$ enthält.

**Satz 7.4-1:** *$G = (E, K)$ sei ein Digraph, $q$ und $s$ seien Ecken in $G$. Dann ist die Maximalzahl gerichteter kantendisjunkter Wege von $q$ nach $s$ gleich der minimalen Elementeanzahl in einer $q$ und $s$ trennenden Kantenmenge.*

**Beweis:** Wir betrachten das Fluss-Netz $N$ auf $G$, bei dem jede Kante Kapazität $1$ hat. Dabei ist vorausgesetzt, dass $s$ von $q$ aus erreichbar ist, sonst ist die Behauptung ohnehin erfüllt. Wie wir bereits bemerkt haben, liefern $i$ kantendisjunkte Wege von $q$ nach $s$ in $G$ einen $0$-$1$-Fluss $f$ auf $N$ vom Wert $i$, und umgekehrt können jedem $0$-$1$-Fluss $f$ auf $N$ mit Wert $i$ ebensoviele kantendisjunkte Wege von $q$ nach $s$ in $G$ zugeordnet werden (allerdings nicht in eindeutiger Weise). Die Maximalzahl kantendisjunkter Wege von $q$ nach $s$ ist also gleich dem maximalen Wert eines $0$-$1$-Flusses auf $N$, welcher nach Folgerung 7.2-2 mit dem Wert eines maximalen Flusses übereinstimmt. Nach Satz 7.2-3 ist dies die minimale Kapazität eines Schnitts in $N$. Wir zeigen nun, dass die minimale Elementeanzahl einer $q$ und $s$ trennenden Kantenmenge in $G$ gleich der Kapazität eines minimalen Schnitts in $N$ ist, womit dann die Behauptung folgt.

Jeder Schnitt $[Q, S]$ von $N$ liefert offenbar die trennende Kantenmenge $A = \{k \in K | k^- \in Q, k^+ \in S\}$ mit $|A| = c(Q, S)$. Umgekehrt sei nun $A$ eine minimale $q$ und $s$ trennende Kantenmenge. $Q_A$ sei die Menge derjenigen Ecken $e$, die von $q$ aus auf einem gerichteten Weg ohne Kanten aus $A$ erreichbar sind, $S_A := E \backslash Q_A$. $[Q_A, S_A]$ ist ein Schnitt in $N$. Es muss nun jede Kante $k$ mit $k^- \in Q_A$ und $k^+ \in S_A$ in $A$ enthalten sein - dabei ist o.B.d.A. vorausgesetzt worden, dass $s$ von jeder Ecke aus, insbesondere von $k^-$ aus, erreichbar ist. Wegen der Minimalität von $A$ folgt $A = \{k \in K | k^- \in Q_A, k^+ \in S_A\}$, und $|A|$ ist die Kapazität des (minimalen) Schnitts $[Q_A, S_A]$. Damit ist der Beweis erbracht.

Wir behandeln nun den Satz von Menger, bei dem es um eckendisjunkte Wege geht. Entsprechend werden trennende Eckenmengen betrachtet: bei gegebenem $G = (E, K)$ und $q, s \in E$ heißt $B \subseteq E$ eine $q$ und $s$**trennende Eckenmenge**, wenn jeder gerichtete Weg von $q$ nach $s$ eine Ecke aus $B$ enthält. Der folgende berühmte Satz wurde von K. Menger im Jahre 1927 bewiesen. Wir bedienen uns hier im Beweis allerdings des Satzes Satz 7.4-1, der erst 1956 von L.R. Ford und D.R. Fulkerson veröffentlicht wurde.

**trennende Eckenmenge**

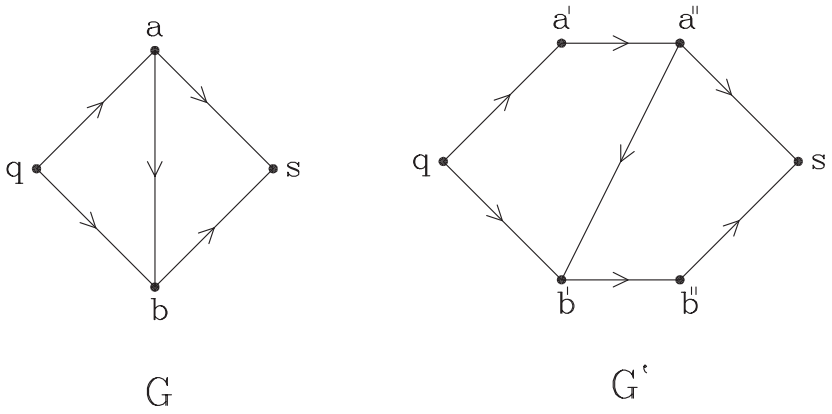
**Satz 7.4-2: Satz von Menger**

***Satz von Menger**:Es sei $G = (E, K)$ ein gerichteter Graph, $q, s \in E$ seien nicht-benachbarte Ecken. Dann ist die Maximalzahl eckendisjunkter Wege von $q$ nach $s$ gleich der minimalen Elementeanzahl einer $q$ und $s$ trennenden Eckenmenge.*

**Satz von Menger**

**Beweis:** Wir konstruieren von $G$ ausgehend einen Digraphen $G'$, indem wir jede Ecke $e$ von $G$ (außer $q$ und $s$) durch zwei neue Ecken $e'$ und $e''$ ersetzen. Die Kanten $k$ in $G$ mit $k^+ = e$ kommen in $G'$ bei $e'$ an (also $k^+ = e'$), im Falle $k^- = e$ in $G$ wird $k^- = e''$ in $G'$. Ferner führt für jedes $e$ in $G$ eine Kante von

$e'$ nach $e''$ in $G'$. Ein Beispiel des Übergangs von $G$ nach $G'$ ist im folgenden Diagramm dargestellt:




Wir wenden nun denSatz 7.4-1 über trennende Kantenmengen auf $G'$ an. Zunächst kann festgestellt werden, dass eckendisjunkte Wege in $G$ kantendisjunkten Wegen in $G'$ entsprechen. Damit folgt, dass die Maximalzahl eckendisjunkter Wege in $G$ gleich der minimalen Elementeanzahl einer $q$ und $s$ trennenden Kantenmenge in $G'$ ist. Da man sich nun darauf beschränken kann, solche trennenden Kantenmengen von $G'$ zu betrachten, die nur Kanten der Form $e'e''$ enthalten (eine Kante der Form $a''b'$ kann man immer durch $a'a''$ ersetzen), und da solche trennenden Kantenmengen wiederum trennenden Eckenmengen in $G$ entsprechen, ist damit der Beweis abgeschlossen.

Wir wollen das Problem der Zuverlässigkeit von Wegen in Kommunikationsnetzen noch einmal im Lichte der hier in Satz 7.4-1 und Satz 7.4-2 behandelten Sätze betrachten. Beide Sätze sind übrigens - bei entsprechender Übertragung der Begriffe - in identischer Formulierung auch für ungerichtete Graphen richtig. Ob man nun das Augenmerk auf Kanten- oder auf Eckendisjunktheit der Wege legt - Ergebnis der Sätze ist, dass sich die Zuverlässigkeit (oder maximale Anzahl disjunkter Wege) in jedem Falle nach der minimalen Anzahl von Leitungen bzw. Zwischenstationen richtet, die so ausgewählt sind, dass jeder Weg (von der Anfangs- zur Zielstation) eine von ihnen benutzen muss.

**Selbsttestaufgabe 7.4-1:**

*Ein Graph $G$ stelle ein Datennetz dar, $A$ und $B$ seien Stationen (Ecken). Beschreiben Sie, wie man feststellen kann, wieviele Leitungen (Kanten) maximal ausfallen dürfen, so dass auf jeden Fall immer noch ein Weg von $A$ nach $B$ genutzt werden kann.*

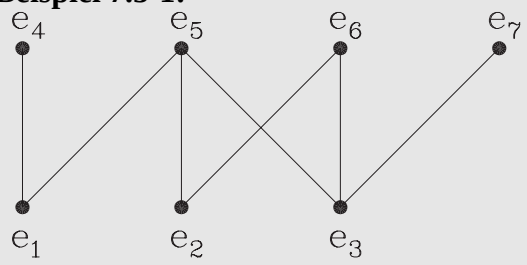
## 7.5 Weitere kombinatorische Anwendungen

Im Abschnitt 7.4 wurde mit dem Satz von Menger ein kombinatorischer $min$-$max$-Satz über Wege in Graphen behandelt, wobei der Beweis sich auf die zuvor erhaltenen Ergebnisse über Flüsse - insbesondere den Satz von Ford und Fulkerson - stützte. In diesem Abschnitt werden wir den Satz von Menger dazu nutzen, zwei weitere "klassische" Sätze der Graphentheorie abzuleiten, nämlich den Satz von König und Egerváry und den Satz von Hall. Bei diesen beiden Sätzen geht es um Korrespondenzen in bipartiten Graphen.

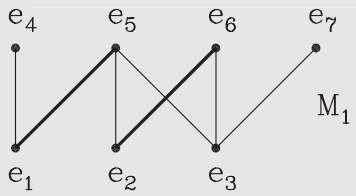
Es sei $G = (E_1 \cup E_2, K)$ ein (ungerichteter) bipartiter Graph. Nach Definition (s. Definition 1.2-8) hat jede Kante einen Endpunkt in $E_1$ und den anderen in $E_2$, wobei die Eckenmengen $E_1$ und $E_2$ disjunkt sind. Eine Menge $M \subseteq K$ von Kanten heißt eine **Korrespondenz** (oder: **Paarung**), wenn keine Kanten aus $M$ benachbart sind. Die Bezeichnung "Korrespondenz" erklärt sich so, dass mittels der Kanten aus $M$ einige Ecken aus $E_1$ zu gewissen Ecken aus $E_2$ korrespondieren bzw. diesen eindeutig zugeordnet sind.

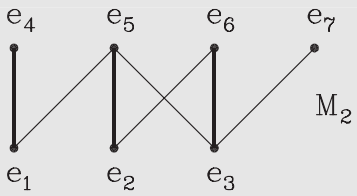
**Korrespondenz**
**Paarung**

**Beispiel 7.5-1:**




In den folgenden beiden Diagrammen sind Korrespondenzen $M_1$ bzw. $M_2$ durch verstärkt gezeichnete Kanten herausgehoben:




Bei $M_1$ korrespondiert $e_1$ zu $e_5$ und $e_2$ zu $e_6$; bei $M_2$ lauten die Paarungen $e_1e_4$, $e_2e_5$ und $e_3e_6$.

In obigem Beispiel besteht $M_1$ aus zwei, $M_2$ aus drei Kanten. Da $G$ nur sieben Ecken hat, kann es offenbar keine Korrespondenz mit mehr als drei Kanten geben. Allgemein wird eine Korrespondenz $M$ in $G = (E_1 \cup E_2, K)$ **maximal** genannt, werden für jede Korrespondenz $M'$ gilt $|M'| \leq |M|$. Wir wollen uns nun mit

**maximale Korrespondenz**

dem Problem beschäftigen, für einen beliebig vorgegebenen bipartiten Graphen eine maximale Korrespondenz zu finden.

Die Frage nach maximalen Korrespondenzen spielt in vielen Anwendungen eine Rolle. Zum Beispiel können die Ecken des bipartiten Graphen Personen (etwa in einer Abteilung einer Firma) und Arbeitsplätze repräsentieren, wobei eine Kante existiert, wenn eine Person für einen gewissen Arbeitsplatz qualifiziert ist. Die Frage nach einer maximalen Korrespondenz bedeutet dann, dass man möglichst vielen Personen einen der für sie in Frage kommenden Arbeitsplätze zuordnen möchte. Eine andere denkbare Situation ist folgende: In einem aus DV-Stationen und Peripherie-Einheiten (wie z. B. Drucker, Plotter) bestehenden Netz liegen für jede der DV-Stationen Anforderungen zur Nutzung gewisser Peripherie-Einheiten vor, wobei keine Peripherie-Einheit gleichzeitig von mehreren Stationen genutzt werden kann. Sollen die Ressourcen möglichst gut genutzt werden, so muss stets einer möglichst großen Anzahl von DV-Stationen jeweils eine der gewünschten Peripherie-Einheiten zugeordnet werden.

Der Satz von König und Egerváry liefert eine $min$-$max$-Aussage über die Größe einer maximalen Korrespondenz. Wir erinnern an den Begriff einer Knotenüberdeckung (siehe Abschnitt 3.3 in 3.3.3): eine Menge $U \subseteq E$ von Ecken eines Graphen $(E, K)$ ist eine Knotenüberdeckung, wenn für jede Kante $\{e, f\} \in K$ gilt $e \in U$ oder $f \in U$.

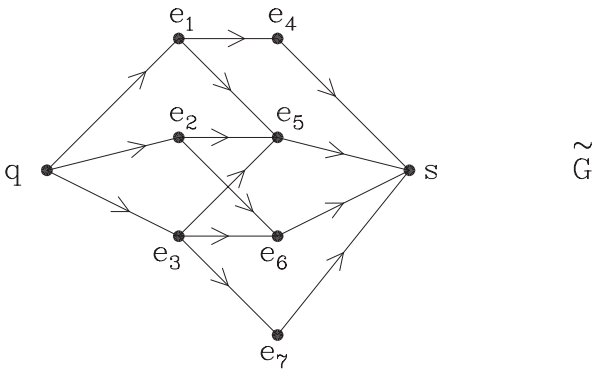
**Satz 7.5-1: König und Egerváry**

**Satz von König und Egerváry**

*$G = (E_1 \cup E_2, K)$ sei ein bipartiter Graph. Dann ist die maximale Mächtigkeit einer Korrespondenz in $G$ gleich der minimalen Mächtigkeit einer Knotenüberdeckung von $G$.*

**Beweis:** Der Satz wurde im Jahre 1931 unabhängig voneinander von D. König und E. Egerváry bewiesen. Wir leiten den Satz direkt aus dem Satz von Menger ab.

Dazu erweitern wir den Graphen $G$ zu einem Digraphen $\tilde{G}$, indem wir zwei neue Ecken $q$ und $s$ sowie neue Bögen $qe$ für $e \in E_1$ und $fs$ für $f \in E_2$ hinzunehmen. Die Kanten zwischen $E_1$ und $E_2$ werden von $E_1$ nach $E_2$ hin gerichtet. Auf diese Weise entsteht z. B. aus dem in Beispiel 7.5-1 verwendeten Graphen $G$ der hier dargestellte Digraph $\tilde{G}$:



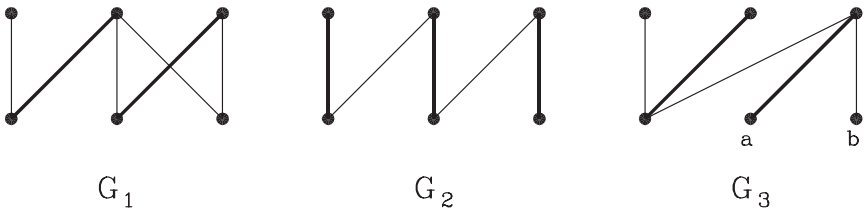

Es liegt nun auf der Hand, dass eine aus $p$ Kanten bestehende Korrespondenz in $G$ $p$ vielen eckendisjunkten Wegen von $q$ nach $s$ in $\tilde{G}$ entspricht. Andererseits entspricht eine Knotenüberdeckung von $G$ in eindeutiger Weise einer $q$ und $s$ trennenden Eckenmenge in $\tilde{G}$. Damit folgt die Behauptung unmittelbar aus dem auf $\tilde{G}$ angewendeten Satz von Menger.

Die Beweise der Sätze von Menger bzw. König und Egerváry zeigen auch auf, wie - letztlich basierend auf dem Algorithmus von Ford und Fulkerson - eine maximale Korrespondenz auf einem bipartiten Graphen effektiv bestimmt werden kann: Es wird wie oben der Digraph $\tilde{G}$ betrachtet, dort jede Kante $k$ mit $c(k) = 1$ bewertet, sodann wird für das Fluss-Netz $(\tilde{G}, c, q, s)$ ein maximaler 0-1-Fluss bestimmt; die zum ursprünglichen $G$ gehörenden Kanten, auf denen der Fluss den Wert 1 hat, bilden nun eine maximale Korrespondenz in $G$.

Es ist natürlich auch möglich, das soeben beschriebene Verfahren direkt als Algorithmus für $G$ zu formulieren, ohne den Umweg über die Konstruktion von $\tilde{G}$ zu gehen. Es ist dann allerdings nicht mehr ersichtlich, dass es sich im Grunde um eine spezielle Anwendung des Algorithmus von Ford und Fulkerson handelt. Die genaue Formulierung eines Algorithmus soll an dieser Stelle dem Leser überlassen werden.

Bisher wurde nicht vorausgesetzt, dass in dem betrachteten bipartiten Graphen $G = (E_1 \cup E_2, K)$ die Mengen $E_1$ und $E_2$ die gleiche Anzahl von Elementen haben. Ist dies der Fall, so ist es möglich, dass eine maximale Korrespondenz $M$ alle Ecken erfasst, d. h. dass jede Ecke zu einer Kante von $M$ gehört. In diesem Falle spricht man von einer **vollständigen Korrespondenz**

**vollständige Korrespondenz**




Die in $G_2$ hervorgehobene Korrespondenz ist vollständig, die in $G_1$ nicht - jedoch gibt es auch in $G_1$ eine vollständige Korrespondenz. Auch die für $G_3$ eingezeichnete Korrespondenz ist nicht vollständig; $G_3$ besitzt keine vollständige Korrespondenz.

Die Aussage, dass $G_3$ in obigem Beispiel keine vollständige Korrespondenz besitzt, lässt sich leicht begründen: die Ecken $a$ und $b$ haben zusammen nur eine Nachbarecke und können daher niemals beide von den Kanten einer Korrespondenz erfasst werden.

Die an diesem Beispiel gewonnene Einsicht lässt sich zu einer notwendigen Bedingung für die Existenz einer vollständigen Korrespondenz verallgemeinern. Dazu führen wir die folgende Schreibweise ein: ist $G = (E_1 \cup E_2, K)$ ein bipartiter Graph und $A \subseteq E_1$, so ist

$$\Phi(A) := \{b \in E_2 | \text{ es gibt ein } a \in A \text{ mit } \{a, b\} \in K\},$$

d. h. $\Phi(A)$ besteht aus den Ecken von $E_2$, die mit irgendeiner Ecke aus $A$ durch eine Kante verbunden sind. Die notwendige Bedingung für die Existenz einer vollständigen Korrespondenz, auf die wir hinaus wollen, lautet:

es gilt $|\Phi(A)| \geq |A|$ für jede Teilmenge $A \subseteq E_1$.

Existiert nämlich eine vollständige Korrespondenz $M$, und ist $A \subseteq E_1$ vorgegeben, so bilden die in $\Phi(A)$ enthaltenen Endpunkte der Kanten von $M$, deren Endpunkte in $E_1$ sogar in $A$ liegen, eine zu $A$ gleichmächtige Teilmenge von $\Phi(A)$, mithin folgt $|\Phi(A)| \geq |A|$.

Überraschenderweise ist diese notwendige Bedingung auch hinreichend. Es gilt nämlich der folgende Satz:

**Satz 7.5-2: Satz von Hall**

**Satz von Hall**

*$G = (E_1 \cup E_2, K)$ sei ein bipartiter Graph mit $|E_1| = |E_2|$. Dann gibt es in $G$ genau dann eine vollständige Korrespondenz, wenn für jede Teilmenge $A \subseteq E_1$ gilt $|\Phi(A)| \geq |A|$.*

**Beweis:** Wir zeigen zunächst, dass jede Knotenüberdeckung mindestens so viele Ecken enthält wie $E_1$, und wenden dann den Satz von König und Egerváry an.

Es sei $U$ eine Knotenüberdeckung, $U_1 := U \cap E_1$ und $U_2 := U \cap E_2$; es soll $|U| \geq |E_1|$ gezeigt werden.

$E' := E_1 \backslash U_1$ seien die nicht zu $U$ gehörenden Ecken von $E_1$. Man kann von $E' \neq \emptyset$ ausgehen, denn sonst wäre $E_1 \subseteq U_1 \subseteq U$, und $|E_1| \leq |U|$ würde bereits folgen.

Wichtig ist nun die Beobachtung, dass $\Phi(E') \subseteq U_2$ gelten muss. Gäbe es nämlich eine Ecke $u \in \Phi(E') \backslash U_2$, so hätte man ein $e \in E'$ mit $k = \{e, u\} \in K$, $e \in E_1 \backslash U_1$ und $u \in E_2 \backslash U_2$, und $U$ könnte keine Knotenüberdeckung sein.

Mit $|U_2| \geq |\Phi(E')|$ können wir jetzt schließen

$$U| = |U_1| + |U_2| \geq |U_1| + |\Phi(E')| \geq |U_1| + |E'| = |E_1|,$$

d. h. $|U| \geq |E_1|$ ist gezeigt.

Da jede Knotenüberdeckung also mindestens $|E_1|$ viele Elemente hat, gilt dies auch für eine minimale Knotenüberdeckung. Es folgt, dass $|E_1|$ die minimale Mächtigkeit einer Knotenüberdeckung ist. Nach dem Satz von König und Egerváry ist dies auch die maximale Mächtigkeit einer Korrespondenz, d. h. in $G$ muss es eine vollständige Korrespondenz geben.

Der obige Satz wurde im Jahre 1935 von P. Hall bewiesen. Wir wollen nur darauf hinweisen, dass der Satz von König und Egerváry auch umgekehrt aus dem Satz von Hall hergeleitet werden kann. In der Literatur findet man auch direkte Beweise des Satzes von Hall, d. h. ohne Herstellung eines Zusammenhangs mit Flüssen und $min$-$max$-Sätzen.

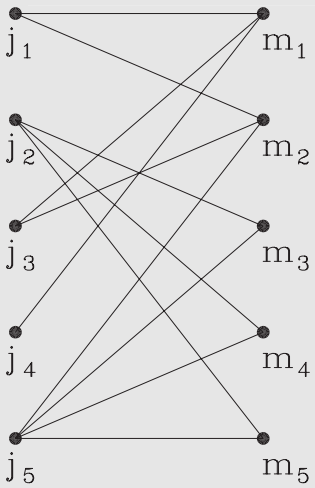
**Heiratssatz**

Oft findet man den Satz von Hall unter dem Namen **Heiratssatz**. Dieser Name leitet sich aus dem folgenden "Heiratsproblem" ab:

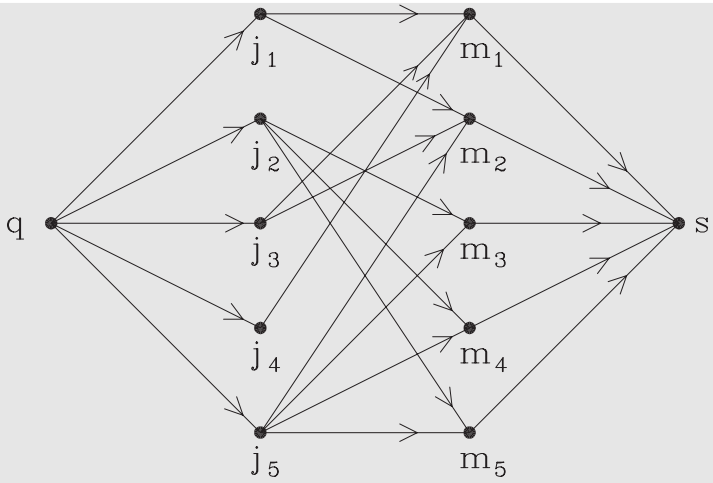
Angenommen, man hat $n$ viele Jungen und gleichviele Mädchen, und jeder der Jungen kennt einige der Mädchen. Es sollen nun die Jungen mit den Mädchen so verheiratet werden, dass jeder Junge eines der Mädchen, die er kennt, heiratet. Der Heiratssatz sagt nun, dass dies genau dann möglich ist, wenn für beliebiges $i$ mit $1 \leq i \leq n$ gilt: je $i$ viele Jungen kennen insgesamt mindestens $i$ viele Mädchen.

Es gibt ein Teilgebiet der Kombinatorik, die "Transversalentheorie", welches sich mit Problemen dieser Art beschäftigt. Wir gehen darauf hier nicht näher ein.

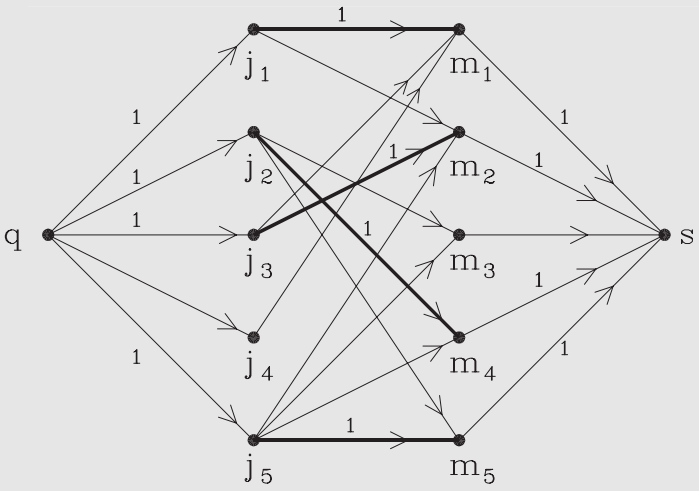
**Beispiel 7.5-2:**
Im folgenden bipartiten Graphen seien die Bekanntschaften zwischen fünf Jungen und fünf Mädchen dargestellt:



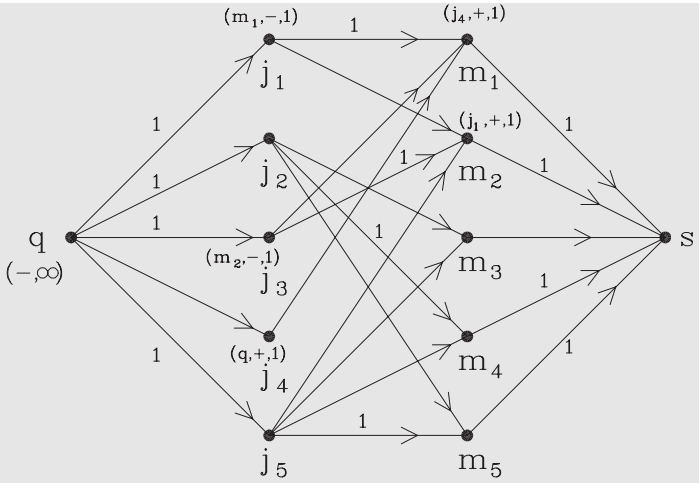

Um festzustellen, ob es eine vollständige Korrespondenz gibt, bilden wir zunächst folgendes Fluss-Netz, wo jede Kante $k$ die Kapazität $c(k) = 1$ hat (dies nehmen wir nicht ins Diagramm auf):




Mit dem Algorithmus von Ford und Fulkerson kann man z. B. den im folgenden Bild dargestellten Fluss erhalten. Aus Gründen der Übersichtlichkeit sind die Fluss-Werte nur bei den mit $1$ bewerteten Kanten (nicht bei denen mit Fluss-Wert $0$) dazugeschrieben, die mit $1$ bewerteten Originalkanten sind dabei hervorgehoben.




Der vorliegende Fluss ist maximal. Ein weiterer Algorithmusschritt führt zu den im nächsten Diagramm gezeigten Markierungen, dann stoppt der Algorithmus.




Es existiert also keine vollständige Korrespondenz - ”nicht alle können heiraten”. Die zuletzt erhaltenen Markierungen liefern auch eine Menge $A \subseteq \{j_1, \ldots j_5\}$ mit $|\Phi(A)| < |A|$: für $A = \{j_1, j_3, j_4\}$ ist $\Phi(A) = \{m_1, m_2\}$.

**Selbsttestaufgabe 7.5-1:**

*Erläutern Sie, wie man mit dem Algorithmus von Ford und Fulkerson eine maximale Korrespondenz in einem bipartiten Graphen finden kann.*

## 7.6 Zulässige Flüsse und Zirkulationen

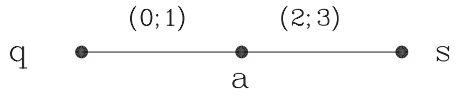
Bei der bisherigen Behandlung von Flüssen wurde immer davon ausgegangen, dass die Fluss-Werte $f(k)$ nach unten durch den Wert $0$ und nach oben durch die Kapazität $c(k)$ beschränkt sind.

**Definition 7.6-1:** *Wir betrachten nun die Situation, dass für ein Fluss-Netz $N = (G, c, q, s)$ noch eine weitere Kantenbewertung $b : K \to \mathbb{R}_0^+$ gegeben ist und $b(k) \leq c(k)$ für alle $k \in K$ gilt. $b(k)$ wird als untere Kapazitätsschranke aufgefasst, d. h. für einen Fluss $f$ soll gelten*

$$b(k) \leq f(k) \leq c(k) \quad \textit{für alle } k \in K.$$

*In diesem Fall wird $f$ ein **zulässiger Fluss** auf dem Fluss-Netz $N$ mit unterer Kapazitätsgrenze $b$ genannt.* **zulässiger Fluss**

Das Hauptinteresse gilt auch hier der Bestimmung eines maximalen (zulässigen) Flusses. Neu hinzugekommen ist jedoch nun das Problem, dass es überhaupt keinen zulässigen Fluss (und damit auch keinen maximalen) geben muss. Ein simples Beispiel dafür ist in dem folgenden Bild dargestellt. (Eine Kante $k$ trägt die Beschriftung $(b(k); c(k))$.)




In Abschnitt 7.1 wurde bereits der Zusammenhang zwischen Flüssen und Zirkulationen beleuchtet. Entsprechend kommen wir nun zu der folgenden Begriffsbildung: Gegeben sei ein Digraph $G = (E, K)$ mit zwei Kantenbewertungen $b, c : K \to \mathbb{R}_0^+$ und $b(k) \leq c(k)$ für alle $k \in K$. Eine Zirkulation $f$ auf $G$ heißt **zulässige Zirkulation**, wenn für alle $k \in K$

**zulässige Zirkulation**

$$b(k) \leq f(k) \leq c(k)$$

erfüllt ist.

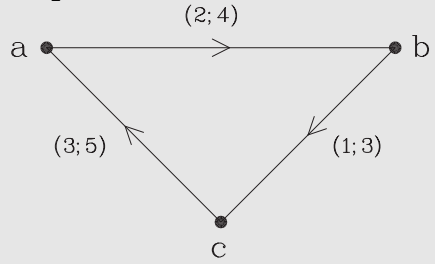
Wie bei den zulässigen Flüssen gilt natürlich auch hier, dass es keine zulässige Zirkulation geben muss. Wir werden uns nun zunächst dem Problem der Existenz zulässiger Zirkulationen zuwenden und danach zu den Flüssen zurückkommen.

Ein Digraph $G = (E, K)$ mit $b$ und $c$ wie oben sei gegeben. $G$ wird zu einem Digraphen $\tilde{G}$ erweitert, indem zusätzliche Ecken $q$ und $s$ hinzugefügt werden sowie gerichtete Kanten $qe$ und $es$ für alle $e \in E$. Auf den Kanten von $\tilde{G}$ wird eine Kapazitätsfunktion $\tilde{c}$ erklärt:

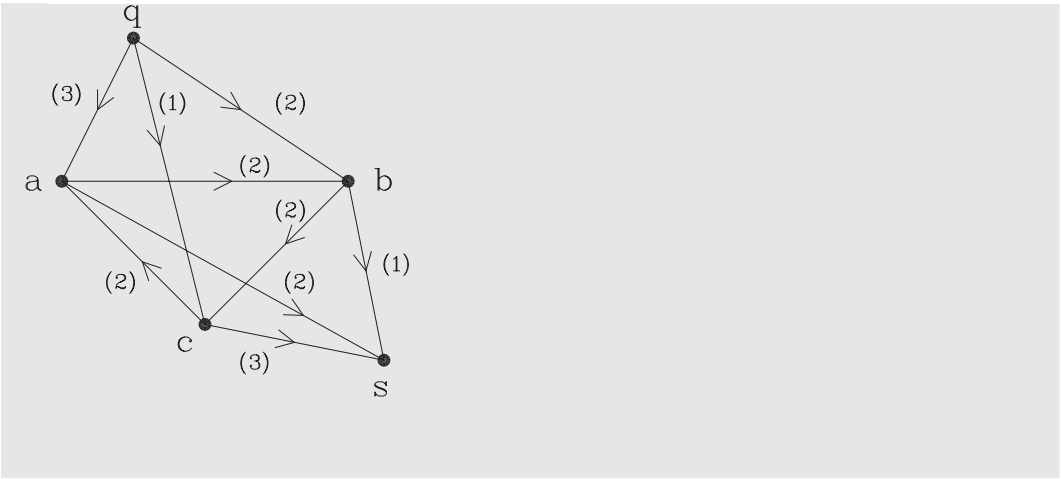
$$\begin{aligned}
\tilde{c}(k) &:= c(k) - b(k) \quad \text{für jede Kante } k \text{ von } G;\\
\tilde{c}(qe) &:= \sum_{k^+=e} b(k) \quad \text{für jede Ecke } e \text{ von } G;\\
\tilde{c}(es) &:= \sum_{k^-=e} b(k) \quad \text{für jede Ecke } e \text{ von } G.
\end{aligned}$$

Man hat nun ein Fluss-Netz $N = (\tilde{G}, \tilde{c}, q, s)$.

**Beispiel 7.6-1:**




Das Diagramm zeigt einen Digraphen $G$ mit Kapazitäten $b$ und $c$. Das folgende Bild stellt das Fluss-Netz $N = (\tilde{G}, \tilde{c}, q, s)$ dar.




Für das Fluss-Netz $N = (\tilde{G}, \tilde{c}, q, s)$ kann ein maximaler Fluss $\tilde{f}$ mit Wert $w(\tilde{f})$ bestimmt werden. Es muss natürlich

$$w(\tilde{f}) \leq \sum_{k \in K} b(k)$$

sein, wobei nach Definition gilt

$$\sum_{k \in K} b(k) = \sum_{e \in E} \tilde{c}(qe) = \sum_{e \in E} \tilde{c}(es).$$

$w(\tilde{f})$ nimmt offenbar genau dann diesen Maximalwert an, wenn jede Kante der Form $qe$ (und jede der Form $es$) durch $\tilde{f}$ "gesättigt" ist, d. h. $\tilde{f}(qe) = \tilde{c}(qe)$ und $\tilde{f}(es) = \tilde{c}(es)$ ist. Der folgende Satz sagt aus, dass diese Bedingungen an $\tilde{f}$ zur Existenz einer zulässigen Zirkulation auf $G$ äquivalent sind.

**Satz 7.6-1:** *Es sei $G = (E, K)$ ein Digraph mit Kapazitäten $b$ und $c$. $N = (\tilde{G}, \tilde{c}, q, s)$ sei das dazu wie oben konstruierte Fluss-Netz. Es gibt genau dann eine zulässige Zirkulation auf $G$, wenn der maximale Wert eines Flusses auf $N$ durch $\sum_{k \in K} b(k)$ gegeben ist.*

**Beweis:** Es sei $\tilde{f}$ ein Fluss auf $N$ mit $w(\tilde{f}) = \sum_{k \in K} b(k)$. Wir behaupten, dass durch

$$f(k) := \tilde{f}(k) + b(k) \text{ für } k \in K$$

eine zulässige Zirkulation auf $G$ definiert wird. Da für $k \in K$ gilt $0 \leq \tilde{f}(k) \leq \tilde{c}(k) = c(k) - b(k)$, folgt $b(k) \leq f(k) \leq c(k)$. Es muss noch gezeigt werden, dass für jede Ecke $e \in E$

$$\sum_{k^+ = e} f(k) = \sum_{k^- = e} f(k)$$

richtig ist, dann ist $f$ als Zirkulation nachgewiesen. $e$ sei also vorgegeben. Da $\tilde{f}$ ein Fluss ist, hat man

$$\tilde{f}(qe) + \sum_{k^+ = e} \tilde{f}(k) = \tilde{f}(es) + \sum_{k^- = e} \tilde{f}(k).$$

Da $qe$ und $es$ durch $\tilde{f}$ gesättigt sind, erhält man

$$\sum_{k^+=e} b(k) + \sum_{k^+=e} \tilde{f}(k) = \sum_{k^-=e} b(k) + \sum_{k^-=e} \tilde{f}(k)$$

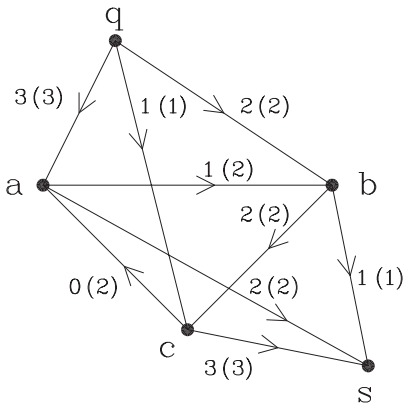
und daraus das gewünschte Ergebnis.

Nun sei umgekehrt $f$ eine zulässige Zirkulation auf $G$. Wir definieren auf den Kanten von $\tilde{G}$ die Abbildung $\tilde{f}$ durch

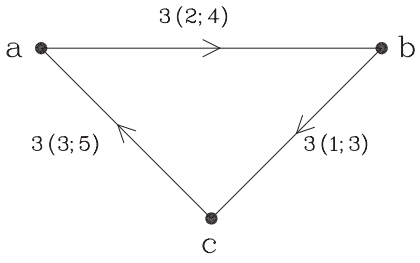
$$\tilde{f}(k) := f(k) - b(k) \text{ für jede Kante } k \text{ von } G;$$
$$\tilde{f}(qe) := \sum_{k^+=e} b(k) \text{ für jede Ecke } e \text{ von } G;$$
$$\tilde{f}(es) := \sum_{k^-=e} b(k) \text{ für jede Ecke } e \text{ von } G.$$

Es ist jetzt leicht zu sehen, dass $\tilde{f}$ ein Fluss auf $N$ ist mit $w(\tilde{f}) = \sum_{k \in K} b(k)$.

Der Satz soll an dem Beispiel Beispiel 7.6-1 illustriert werden. Mit dem Algorithmus von Ford und Fulkerson gelangt man z. B. zu dem im folgenden Bild dargestellten maximalen Fluss $\tilde{f}$ mit $w(\tilde{f}) = 6$.




Da $w(\tilde{f}) = \sum_{k \in K} b(k)$ gilt, lässt sich - wie im Beweis des Satzes beschrieben - daraus eine zulässige Zirkulation für den ursprünglich vorgegebenen Digraphen konstruieren. Man erhält mit $f(k) := \tilde{f}(k) + b(k)(k \in K)$ das folgende Ergebnis:



Es sei nun wieder ein Fluss-Netz $N = (G, c, q, s)$ mit zusätzlicher unterer Kapazität $b$ gegeben. Die Bestimmung eines maximalen zulässigen Flusses kann in folgenden Schritten geschehen:

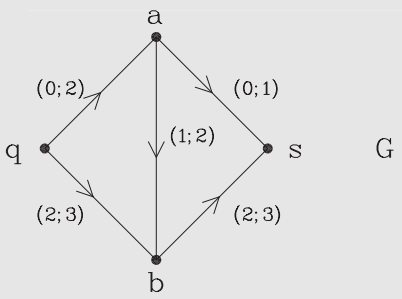
1. Bestimmung irgendeines zulässigen Flusses $f_0$, sofern ein solcher existiert.

2. Bestimmung eines maximalen zulässigen Flusses $f_{max}$, ausgehend von $f_0$. Ergibt sich bei Schritt 1, dass kein zulässiger Fluss existiert, so ist Schritt 2 natürlich gegenstandslos.

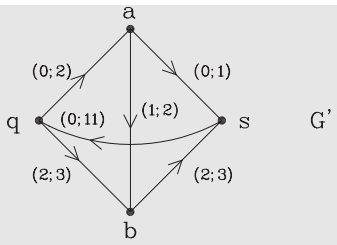
Für Schritt 1 kann folgendermaßen vorgegangen werden: Zunächst erweitert man (wie in Abschnitt 7.1) den Digraphen $G = (E, K)$ durch Hinzunahme einer Kante $sq$ von $s$ nach $q$, der untere Kapazität $b'(sq) = 0$ und obere Kapazität $c'(sq) = \sum_{k \in K} c(k)$ zugeordnet werden, der entstehende Digraph $G'$ ist mit den Bewertungen $b'$ und $c'$ versehen. Von $G'$ ausgehend, wird nun (wie oben beschrieben) das Fluss-Netz auf $\tilde{G}'$ konstruiert und durch Bestimmung eines maximalen Flusses auf diesem Fluss-Netz eine zulässige Zirkulation auf $G'$ ermittelt oder aber festgestellt, dass $G'$ keine zulässige Zirkulation besitzt. Im positiven Falle hat man mit der so gefundenen zulässigen Zirkulation auf $G'$ (durch "Vergessen" der Zusatzkante mit ihren Bewertungen) einen zulässigen Fluss $f_0$ auf $N$.

Schritt 2 kann mit dem Algorithmus von Ford und Fulkerson oder dem von Edmonds und Karp ausgeführt werden. Es sind allerdings kleine Änderungen an den Stellen nötig, wo die untere Kapazitätsgrenze eine Rolle spielt: eine nichtmarkierte Vorgängerecke $g = k^-$ einer bereits markierten Ecke $e = k^+$ kann dann markiert werden, wenn $f(k) > b(k)$ ist, und zwar mit $(e, -, d(g))$, wobei hier gilt $d(g) = min\{f(k) - b(k), d(e)\}$.

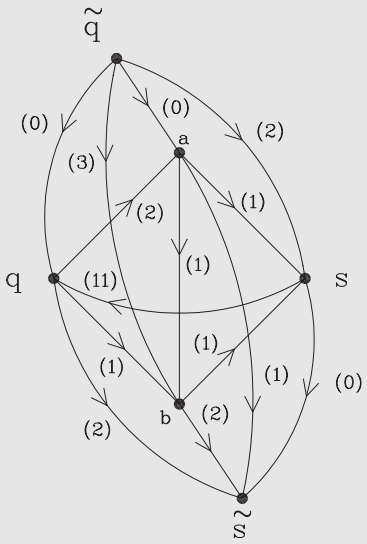
**Beispiel 7.6-2:**
Für den folgenden Digraphen $G$ mit Kapazitäten $b$ und $c$ soll ein maximaler (zulässiger) Fluss von $q$ nach $s$ bestimmt werden:



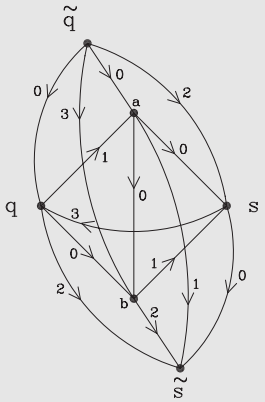

Die Erweiterung $G'$ sieht so aus:



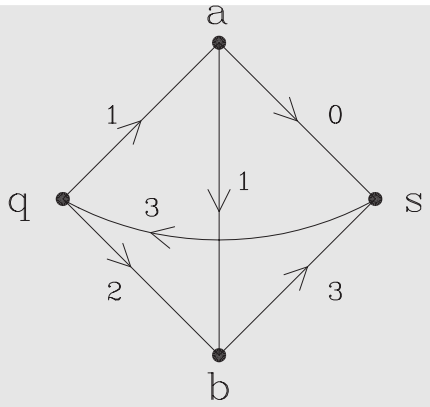

Nun wird $\tilde{G}'$ gebildet:



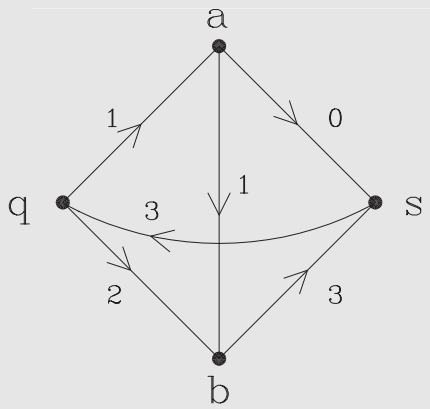

Mit dem Algorithmus von Ford und Fulkerson ergibt sich der Fluss $\tilde{f}$ mit $w(\tilde{f}) = 5$:



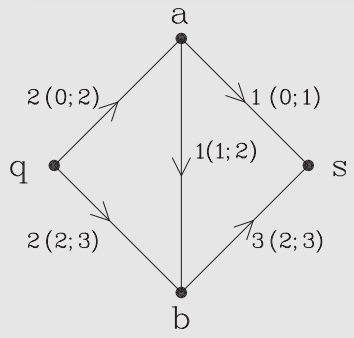

Daraus leitet sich die folgende zulässige Zirkulation auf $G'$ ab:




Damit hat man als zulässigen Fluss $f_0$ von $q$ nach $s$ in $G$ erhalten:




Nun wird (als Schritt $2$) dieser Fluss mit dem Algorithmus von Ford und Fulkerson verbessert. Man sieht sofort, dass sich auf dem zunehmenden Weg $q - a - s$ der Fluss um $1$ vergrößern lässt, der damit erhaltene Fluss ist bereits maximal:




Die Ausführungen über zulässige Flüsse und Zirkulationen sollen hiermit beendet werden. Ähnlich wie bei der schon behandelten Bestimmung maximaler Flüsse gibt es für das Problem der Bestimmung maximaler zulässiger Flüsse (also mit unterer Kapazität $b$) zahlreiche weitere Untersuchungen und Ergebnisse, für die auf die

Literatur verwiesen werden muss. Interessant ist auch, dass hier ebenfalls eine Verbindung von dem Wert eines maximalen zulässigen Flusses zur (geeignet definierten) Kapazität eines Schnitts gezogen werden kann; man kann so eine Verallgemeinerung des "max-flow min-cut" Theorems formulieren.

**Kostenfunktion**

Eine weitere Verkomplizierung ergibt sich, wenn man für ein Fluss-Netz $N = (G, c, q, s)$ mit unterer Kapazität $b$ noch eine **Kostenfunktion**

$$\gamma : K \to \mathbb{R}$$

**Kosten**

gegeben hat. Die **Kosten** eines Flusses $f$ sind definiert als

$$\gamma(f) := \sum_{k \in K} \gamma(k) f(k).$$

Man möchte nun einen maximalen zulässigen Fluss mit möglichst geringen Kosten finden. Auch hierfür wird auf weitere Literatur verwiesen.

**Selbsttestaufgabe 7.6-1:**

*Beschreiben Sie, welchen Einfluss auf die Theorie der Flüsse es hat, wenn für Fluss-Netze auch untere Kapazitätsfunktionen gegeben sind.*

## 7.7 Synthese minimaler Netze

Bisher wurden Flüsse und Zirkulationen auf gegebenen Netzen untersucht. Man kann sich dem Thema jedoch auch aus einer anderen Richtung nähern: Zwischen gegebenen Punkten (Ecken) sollen gewisse Flussbedingungen gelten, und es ist auf dieser Eckenmenge ein Netz gesucht, welches mit möglichst geringen Kosten konstruiert werden kann und die Anforderungen an die Flüsse erfüllt.

Es gibt eine Vielzahl von Möglichkeiten, diese Problemstellung zu präzisieren. Wir wollen hier einen kurzen Einblick geben und nur den Fall betrachten, in dem die Kosten für die Errichtung einer Kante einfach durch deren Kapazität gegeben sind und in dem man ein ungerichtetes Netz mit gegebenen Mindestwerten für die maximalen Flusswerte zwischen den Eckenpaaren sucht.

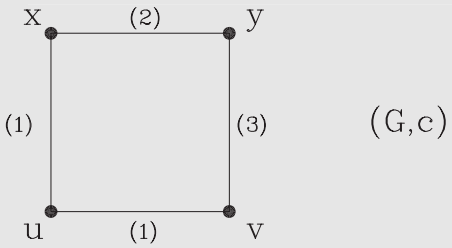
**Definition 7.7-1: Fluss-Funktion**
*Es sei ein Netz $(G, c)$ gegeben, wo $G = (E, K)$ ein (ungerichteter) Graph sei und $c : K \to \mathbb{R}_0^+$ eine nicht-negative Kapazitätsfunktion. (In der Literatur findet man hierfür auch die Bezeichnung "symmetrisches Netzwerk".)*

*Wir betrachten nun für je zwei Ecken $e, f \in E$ das Fluss-Netz $N_{ef} = (G, c, e, f)$ mit $e$ als Quelle und $f$ als Senke, den maximalen Fluss-Wert bezeichnen wir mit $w(e, f)$. (Da die Begriffe und Verfahren bisher nur für Digraphen formuliert wurden, muss man hier zu dem Digraphen übergehen, der entsteht, wenn jede Kante*

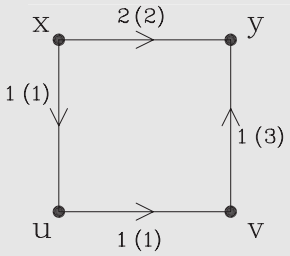
*von $G$ durch zwei entgegengesetzte Bögen ersetzt wird. Man kann aber die bisherigen Ergebnisse auch leicht direkt übertragen; eine ungerichtete Kante kann dann in beiden Richtungen "genutzt" werden.)*

*Das so definierte $w$ ist offenbar eine symmetrische Funktion, d. h. es gilt $w(e,f) = w(f,e)$ für alle Paare $e, f \in E$. $w$ wird die **Fluss-Funktion** des Netzes $(G,c)$ genannt.* **Fluss-Funktion**

**Beispiel 7.7-1:**




Es ist z. B. $w(x,y) = 3$, denn der im folgenden Diagramm dargestellte Fluss von $x$ nach $y$ mit Wert 3 ist maximal:




Insgesamt ergibt sich:

$$
\begin{aligned}
w(x,y) &= w(y,x) = 3\\
w(x,u) &= w(u,x) = 2\\
w(x,v) &= w(v,x) = 3\\
w(y,u) &= w(u,y) = 2\\
w(y,v) &= w(v,y) = 4\\
w(u,v) &= w(v,u) = 2
\end{aligned}
$$

Neben der Symmetrie hat eine Fluss-Funktion die folgende wichtige Eigenschaft: für beliebige $x, y, z \in E$ gilt stets

$$w(x,y) \geq \ min\{w(x,z), w(z,y)\}.$$

Dies kann man etwa so einsehen: Nach dem max-flow min-cut Theorem gibt es einen Schnitt $[X,Y]$ mit $x \in X$ und $y \in Y$, für den $w(x,y) = c(X,Y)$ gilt. Ist jetzt $z \in X$, so folgt $w(z,y) \leq c(X,Y)$ und damit $w(z,y) \leq w(x,y)$, im anderen Falle $z \in Y$ ergibt sich entsprechend $w(x,z) \leq w(x,y)$.

Die gezeigte Ungleichung kann mit Induktion zu folgender Aussage erweitert werden: für beliebige $x_1, \ldots, x_k \in E (k \geq 3)$ gilt

$$w(x_1, x_k) \geq \ min\{w(x_1, x_2), \ldots, w(x_{k-1}, x_k)\}.$$

Wie angekündigt, wenden wir uns nun der Konstruktion eines minimalen Netzes zu. Dazu sei eine Eckenmenge $E$ vorgegeben, ferner eine symmetrische Funktion $r : E \times E \to \mathbb{R}_0^+$. $r$ soll die **Flussanforderungen** für das zu errichtende Netz darstellen. Konsequenterweise nennen wir ein Netz $(G,c)$ auf einem zusammenhängenden Graphen $G = (E,K)$ **zulässig** für $r$, wenn für alle $e, f \in E$ die Bedingung

**Flussanforderungen**

**zulässiges Netz**

$$w(e,f) \geq r(e,f)$$

erfüllt ist.

Ein **minimales Netz** für $r$ ist ein zulässiges Netz, für welches die Summe der Kapazitäten

**minimales Netz**

$$c(K) := \sum_{k \in K} c(k)$$

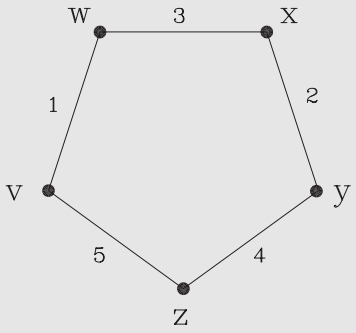
unter allen zulässigen Netzen minimal ist.

Bei der Konstruktion eines minimalen Netzes geht es also darum, die Ecken auf eine solche Weise durch bewertete Kanten zu verbinden, dass deren Kapazitäten insgesamt möglichst gering und gleichzeitig vorgegebene Mindestflüsse zwischen den Ecken möglich sind.

Im folgenden wird das Konstruktionsverfahren für ein minimales Netz beschrieben, wie es von R.E. Gomory und T.C. Hu im Jahre 1961 in einer Untersuchung angegeben wurde. Um den formalen Aufwand zu begrenzen, wird das Verfahren in Worten beschrieben und parallel dazu anhand eines Beispiels illustriert.

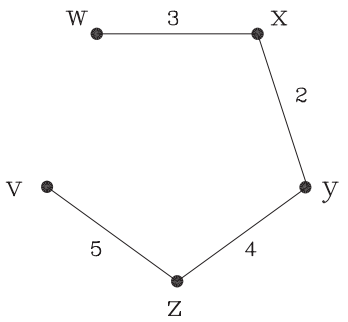
Von $E$ und $r$ wie oben ausgehend, wird zuerst das dadurch induzierte Netz $(G_r, c_r)$ mit $G_r = (E, K_r)$ gebildet: $e$ und $f$ werden im Falle $r(e,f) > 0$ durch eine Kante $ef$ verbunden, und es wird $c_r(ef) := r(e,f)$ gesetzt.

**Beispiel 7.7-2:**
Wir betrachten als Beispiel $E = \{v, w, x, y, z\}$ mit Flussanforderungen $r(v, w) = 1$, $r(w, x) = 3$, $r(x, y) = 2$, $r(y, z) = 4$ und $r(z, v) = 5$, für alle anderen $r$-Werte gelte $r(e, f) = 0$. Es ergibt sich hier folgendes Netz $(G_r, c_r)$:



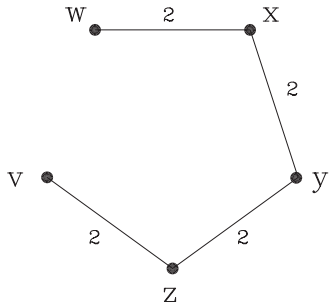

Für $(G_r, c_r)$ wird nun ein maximales Gerüst $B$ bestimmt. Dazu können die Algorithmen von Kruskal oder Prim (siehe Kapitel 5) verwendet werden. Dann wird $B$ in "uniforme Bäume" zerlegt - also Bäume, deren Kanten die gleiche Bewertung tragen. Dies ist folgendermaßen zu verstehen: Ist $p$ die kleinste in $B$ vorkommende Bewertung, so ist der zuerst gewählte uniforme Baum identisch mit $B$, wo jedoch alle Kanten mit $p$ bewertet sind. Dann wird dieser uniforme Baum vom ursprünglichen $B$ "abgezogen", wobei nun mit $0$ bewertete Kanten wegfallen, und mit dem übriggebliebenen Wald wird genauso verfahren usw., bis nur noch uniforme Bäume übrig sind.

In unserem Beispiel $(G_r, c_r)$ hat man als maximales Gerüst:

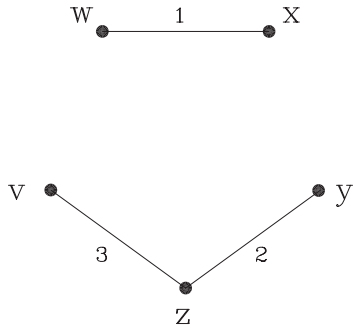



B

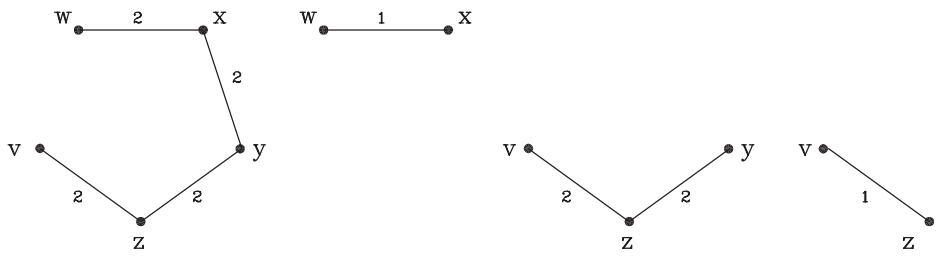
Als uniformen Baum hat man im ersten Schritt:




Als ”Rest” ergibt sich der folgende Wald:

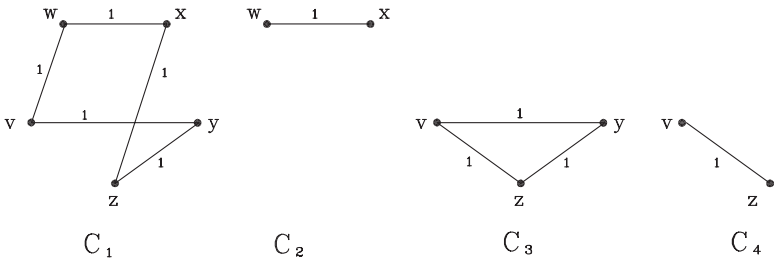



Nach einem weiteren Schritt ist man schließlich insgesamt bei folgender Zerlegung von $B$ in uniforme Bäume angelangt:

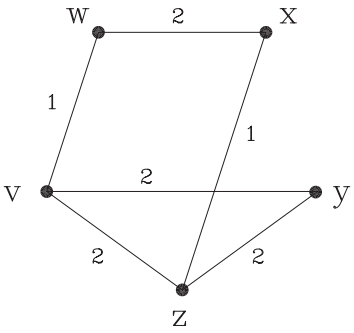



$B$ sei nun in die uniformen Bäume $B_1, \ldots, B_m$ zerlegt worden. Für jeden Baum $B_i$, der mindestens drei Ecken enthält, wird auf den Ecken von $B_i$ in beliebiger Reihenfolge ein einfacher Kreis $C_i$ gebildet. Die Kanten dieses Kreises werden sämtlich mit der Hälfte der Bewertung von $B_i$ bewertet. Die Bäume $B_j$ mit nur zwei Ecken werden unverändert (auch mit ihrer Bewertung) als $C_j$ genommen. Schließlich wird der Graph $G = (E, K)$ gebildet, dessen Kantenmenge $K$ die Vereinigung der Kantenmengen von $C_1, \ldots, C_m$ ist. Die Kantenbewertungen addieren sich dabei zu $c$ auf.

Im Beispiel ergeben sich z. B. als $C_1, \ldots, C_4$:




Als $(G, c)$ erhält man schließlich:




Eine andere Wahl des Kreises $C_1$ hätte zu einem anderen Netz $(G, c)$ geführt, jedoch mit derselben Summe von Kantenbewertungen.

Wir zeigen nun, dass das so konstruierte Netz $(G, c)$ minimal ist.

**Satz 7.7-1:** *Ist $E$ eine Eckenmenge und $r$ eine symmetrische Funktion von Flussanforderungen, so ist das wie oben konstruierte Netz $(G, c)$ ein minimales (zulässiges) Netz für $r$. Zur Konstruktion dieses Netzes ist ein Aufwand von $0(|E|^2)$ nötig.*

**Beweis:** Zunächst wird nachgewiesen, dass $(G, c)$ zulässig ist. Dazu bezeichne $b_i$ die (für alle Kanten gleiche) Bewertung in $B_i$, die Bewertung in $C_i$ ist dann $b_i/2$.

Ist $ef$ eine Kante in $B$, so ist $w(e, f) \geq r(e, f)$ folgendermaßen nachzuweisen: In jedem $C_i$, welches $e$ und $f$ enthält, kann ein Fluss von Wert $b_i$ zwischen $e$ und $f$ fließen. Überlagerung ergibt dann nach Konstruktion einen Fluss von Wert

$$\sum_{\substack{i \\ e,f \in B_i}} b_i = r(e, f).$$

Jetzt seien $e$ und $f$ beliebige Ecken und $W$ der eindeutige Weg von $e$ nach $f$ in $B$,

$$W = e, w_1, \ldots, w_n, f.$$

Nach den obigen Ungleichungen für $w(e,f)$ ist

$$w(e,f) \geq \ min\{w(x,y)|xy \text{ Kante im Weg } W\},$$

und nach dem bereits Gezeigten ist dieses Minimum größer oder gleich

$$min\{r(x,y)|xy \text{ Kante im Weg } W\}.$$

Wäre dieses Minimum der $r(x,y)$ kleiner als $r(e,f)$, so könnte $B$ kein maximales Gerüst von $(G_r, c_r)$ sein. Damit folgt

$$min\{r(x,y)|xy \text{ Kante im Weg } W \geq r(e,f)$$

und insgesamt

$$w(e,f) \geq r(e,f),$$

was gezeigt werden sollte. $(G,c)$ ist also zulässig. Es bleibt die Minimalität nachzuweisen.

Für jede Ecke $e \in E$ sei

$$u(e) := \ max\{r(e,f)|f \neq e\},$$

$u(e)$ ist also der maximale aus $e$ heraus geforderte Flusswert; ferner wird gesetzt $u(E) := \sum\limits_{e\in E} u(e)$. Es wird nun gezeigt, dass für jedes $r$-zulässige Netz $(G', c')$ gilt

$$\sum_{k\in K'} c'(k) \geq \frac{u(E)}{2}$$

und andererseits für $(G,c)$

$$\sum_{k\in K} c(k) = \frac{u(E)}{2}$$

ist. Daraus folgt dann die Minimalität von $(G,c)$.

Da für jedes $e \in E$ $[\{e\}, E\backslash\{e\}]$ ein Schnitt ist (auch im Sinne der Definition in Definition 7.2-1 für Fluss-Netze), folgt für jedes $r$-zulässige Netz $(G', c')$

$$c'(\{e\}, E\backslash\{e\}) \geq u(e)$$

und durch Summation über alle $e$

$$\sum_{e,f\in E} c'(ef) \geq \sum_{e\in E} u(e) = u(E).$$

(Dabei muss $c'(ef) = 0$ eingesetzt werden, wenn in $G'$ keine Kante $ef$ existiert.) Es muss also

$$\sum_{k\in K'} c'(k) \geq \frac{u(E)}{2}$$

sein, denn in $\sum\limits_{e,f\in E} c'(ef)$ ist jede Kante $k$ zweimal betrachtet worden.

Um den letzten Schritt zu machen, setzen wir für $e \in E$

$$u'(e) := max\{r(e,f) | ef \text{ Kante in } B\}.$$

Selbstverständlich gilt $u'(e) \leq u(e)$. Aus der Konstruktion von $(G, c)$ ergibt sich $c(\{e\}, E\backslash\{e\}) = u'(e)$ für jede Ecke $e \in E$. Damit hat man schließlich

$$\sum_{e,f \in E} c(ef) = \sum_{e \in E} u'(e) \leq u(E),$$

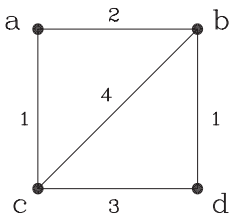
woraus folgt

$$\sum_{k \in K} c(k) \leq \frac{u(E)}{2}.$$

Die Komplexitätsaussage des Satzes ist leicht zu begründen: Ein maximaler Baum $B$ kann mit dem Algorithmus von Prim in $0(|E|^2)$ vielen Schritten bestimmt werden. Danach lässt sich $B$ auch mit Aufwand $0(|E|^2)$ in uniforme Bäume zerlegen. Die Herstellung von $(G, c)$ benötigt dann ebenfalls $0(|E|^2)$ viele Schritte.

**Selbsttestaufgabe 7.7-1:**

*Die Eckenmenge $E = \{a, b, c, d\}$ mit Flussanforderungen $r$ sei gegeben, es ergebe sich für $(G_r, c_r)$ das im folgenden Bild dargestellte Netz:*




$(G_r, c_r)$

*Konstruieren Sie dazu ein minimales (zulässiges) Netz $(G, c)$.*

# 8 Wegauswahl in Netzen

## 8.1 Das Problem der Wegauswahl im Kommunikationsnetzen

Es geht in diesem Kapitel um das folgende Problem. Gegeben sei ein Kommunikationsnetz, in dem einzelne Systeme (oder "Stationen") unter Nutzung von Verbindungsleitungen miteinander kommunizieren. Es muss geregelt werden, welche der Verbindungsleitungen genutzt werden sollen, wenn irgendwelche zwei Systeme miteinander in Kommunikation treten wollen.

**Wegeauswahl**

Es ist üblich, bei diesem Problem der **Wegeauswahl** auch vom "Routing-Problem" zu sprechen. Wir werden diese Sprechweise im folgenden ebenfalls verwenden. Es ist allerdings zu beachten, dass in der Literatur auch im Zusammenhang des Entwurfs hochintegrierter Schaltungen vom **"Routing-Problem"** die Rede ist. Dort geht es um die Plazierung von Bahnen zwischen den Schaltelementen auf einem Chip. Auf diesen Problemkreis wird in diesem Kurs nicht eingegangen.

**Routing-Problem**

**OSI-Schichtenmodell**

Entsprechend dem **OSI-Schichtenmodell** (OSI steht für "Open Systems Interconnection") werden die technischen Abläufe in offenen digitalen Kommunikationssystemen in sieben sogenannte **Schichten** funktional gegliedert. Grundidee dabei ist, dass eine Schicht (in Englisch: Layer) für die nächsthöhere Schicht Dienste erbringt und dazu Dienste der unter ihr liegenden Schicht in Anspruch nehmen kann. Die logischen Abläufe zwischen gleichen Schichten in verschiedenen Systemen werden durch **Protokolle** beschrieben, die Schichten eines Systems verständigen sich untereinander mit Hilfe von **Primärmeldungen**.

**Schicht**

**Protokoll**

**Primärmeldung**

Physikalisch findet die Kommunikation nur auf der untersten Schicht, der Bitübertragungsschicht statt. Je höher eine Schicht angesiedelt ist, desto näher ist sie der eigentlichen Anwendung. Die Schichten sind im einzelnen:

| | |
|---|---|
| Schicht 7 | Anwendungsschicht |
| Schicht 6 | Darstellungsschicht |
| Schicht 5 | Kommunikationssteuerungsschicht |
| Schicht 4 | Transportschicht |
| Schicht 3 | Vermittlungsschicht |
| Schicht 2 | Sicherungsschicht |
| Schicht 1 | Bitübertragungsschicht |

Die Routing-Problematik ist den Aufgaben der Vermittlungsschicht (in Englisch: Network Layer) zuzuordnen. Insgesamt sind deren Aufgaben die Erstellung und Unterhaltung von Netzverbindungen (für verbindungsorientierte Datenübertragung) und von Netzrouten (für verbindungslose Datenübertragung) zwischen Endsystemen im Kommunikationsnetz unter Verwendung von gesicherten Teilstrecken (d. h. unter Verwendung der Schicht 2-Dienste).

In gewisser Hinsicht hat Schicht 3 die komplexesten Aufgaben, denn hier müssen (netzübergreifend) außer der Wegesuche u. a. Flussregelung und Optimierung im Kommunikationsnetz organisiert werden (also z. B. Kostenminimierung, Verzögerungsminimierung, Überlastbehandlung, Blockierungsauflösung).

Das Routing-Problem beinhaltet somit nur einen von mehreren Aufgabenbereichen der Schicht 3. Wir werden zunächst vorwiegend die rein graphentheoretischen Aspekte dieses Problems betrachten. An späterer Stelle kommen wahrscheinlichkeitstheoretische Methoden hinzu, wenn die (vom gewählten Routing-Verfahren abhängige) Zuverlässigkeit eines Netzes untersucht wird.

Im folgenden wird ein Kommunikationsnetz stets durch einen Graphen modelliert, bei dem die Ecken den einzelnen Systemen (Stationen) und die Kanten den Verbindungsleitungen zwischen ihnen entsprechen. Die Kanten werden meist als gerichtet, mitunter auch als ungerichtet angenommen. Oft sind die Kanten mit einer Bewertung versehen, wobei dies in der Anwendung verschiedene Bedeutung haben kann: es kann eine geografische Entfernung gemeint sein, ebenso kann die zum betrachteten Zeitpunkt von der Kante getragene Verkehrsbelastung (oder die Länge der Warteschlange von Nachrichten am Anfang dieser Kante) dahinter stehen.

**leitungsvermittelte, paketvermittelte Kommunikation**

In der Kommunikationstechnik unterscheidet man grundsätzlich zwischen **leitungsvermittelter** und **paketvermittelter** Kommunikation. Bei der Leitungsvermittlung wird den Kommunikationspartnern A und B (in der Regel für Kommunikation in beide Richtungen) für die Dauer des Kommunikationsprozesses ein Weg durch das Netz exklusiv zur Verfügung gestellt, d. h. die beteiligten Leitungen (Kanten des Graphen) sind für jede andere Kommunikation gesperrt. Hingegen werden bei der Paketvermittlung die von A nach B gesendeten Nachrichten einzeln (mit Zieladresse versehen) durch das Netz geroutet, so dass verschiedene Nachrichten von A nach B durchaus auch verschiedene Wege durch das Netz nehmen können. Auch kann eine Leitung gleichzeitig Nachrichten für verschiedene Partner transportieren, es finden keine "Reservierungen" der Kanten statt. Sind zusätzlich die Nachrichten in "Pakete" genormter gleicher Größe aufgeteilt (die u. U. im Ziel wieder zusammengesetzt werden müssen), so spricht man vom **Datagramm-Modus**.

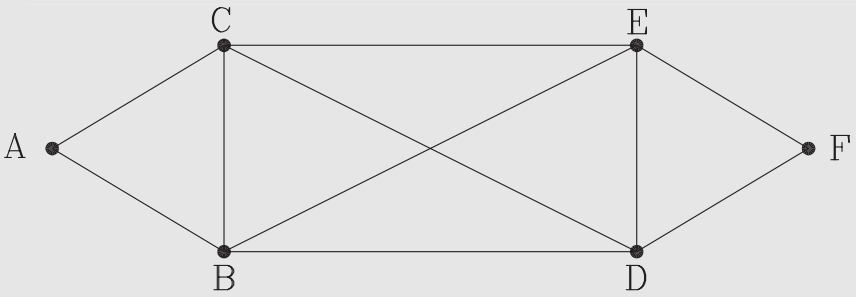
**Datagramm-Modus**

**verbindungslose, verbindungsorientierte Datenübermittlung**

**virtuelle Verbindung**

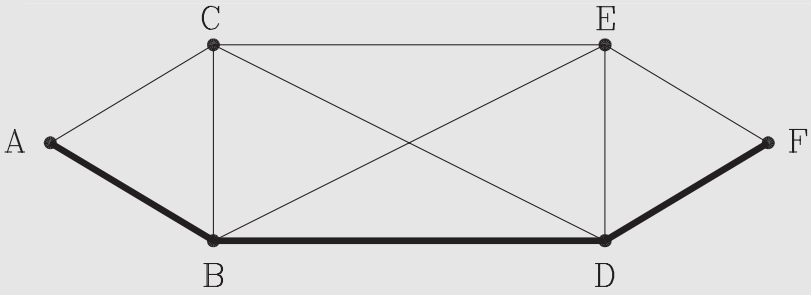
Bei der paketvermittelten Kommunikation, wie sie soeben beschrieben wurde, spricht man auch von der **verbindungslosen Datenübermittlung**. Es gibt allerdings auch die Variante der **verbindungsorientierten Datenübermittlung**: hier wird vor Beginn der Übertragung von Daten von A nach B ein Weg durch das Netz festgelegt, den dann alle Nachrichten nehmen - man sagt, man habe eine **virtuelle Verbindung** aufgebaut. Dadurch stellt man sicher, dass die Nachrichten auf jeden Fall in ihrer ursprünglichen Reihenfolge das Ziel erreichen, weshalb sie keine zusätzliche Sequenzierungsinformation tragen müssen. Die beteiligten Leitungen sind jedoch nicht für zwei Partner reserviert.

**Beispiel 8.1-1:**
Den jetzt folgenden Beispielen liegt immer die hier gezeigte Netzstruktur zugrunde:

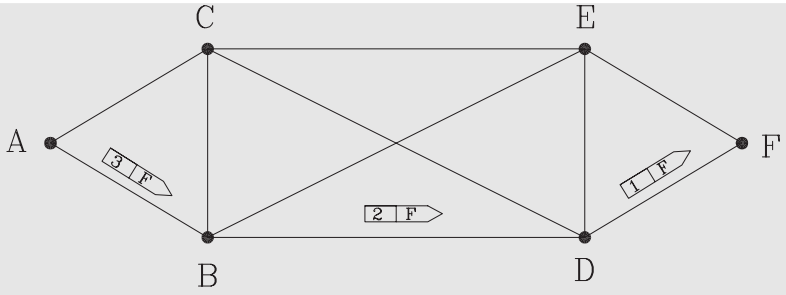



System A möchte Nachrichten an F schicken.

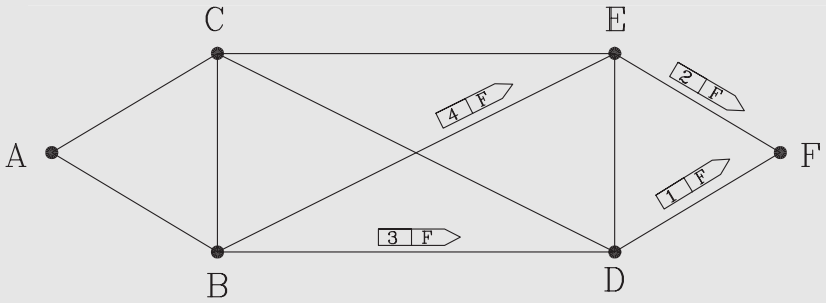
Liegt ein leitungsvermitteltes Netz vor, so muss ein Weg von A nach F durchgeschaltet werden, z. B. der im folgenden Diagramm herausgehobene. Dieser Weg steht dann A und F exklusiv zur Verfügung, bis einer der Partner die Verbindung abbricht.



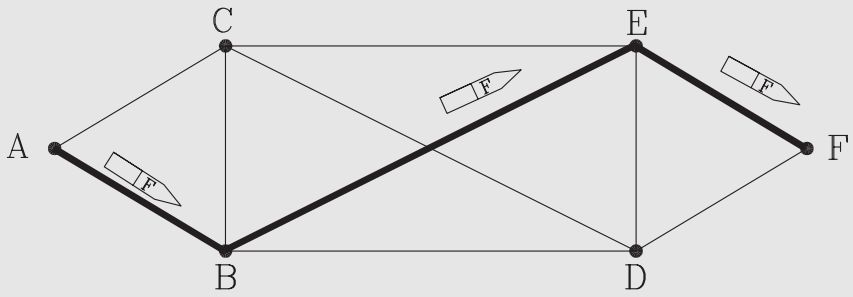

Hat man ein paketvermitteltes Netz, so müssen bei der verbindungslosen Datenübermittlung die Nachrichten durchnumeriert sein, da sie (z. B. durch zeitweilige Störungen im Netz oder durch Lastteilung) bei F in veränderter Reihenfolge ankommen können.
Im folgenden Diagramm nehmen alle Nachrichten von A nach F den "Normalweg" $\{A \rightarrow B \rightarrow D \rightarrow F\}$, sofern keine Störungen vorliegen.




Fällt z. B. BD zeitweise aus, so mag dann $\{A \to B \to E \to F\}$ der Ersatzweg sein. Im nächsten Bild gibt es bei B eine **Lastteilung**: die für F bestimmten Nachrichten schickt B abwechselnd nach D und E.




Handelt es sich schließlich um verbindungsorientierte Datenübermittlung, so wird zunächst ein Weg festgelegt (hier: $\{A \to B \to E \to F\}$), den dann alle Nachrichten nehmen; fällt hier eine der Kanten aus, so muss eine neue Verbindung aufgebaut werden.



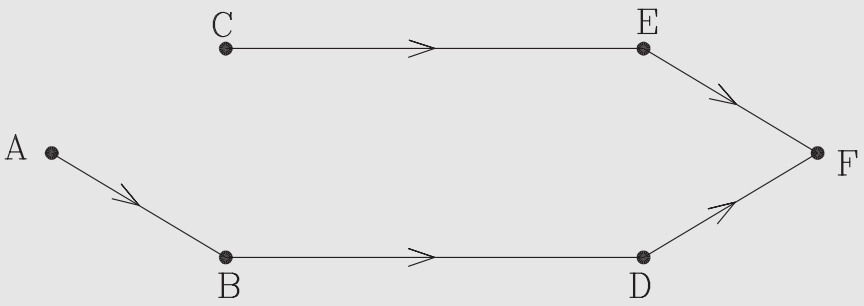

Die obigen Beispiele zeigen, dass sich das reine Problem der Wegeauswahl immer in ähnlicher Form stellt - ob es sich nun um ein leitungs- oder ein paketvermitteltes Kommunikationsnetz handelt. Wir werden im folgenden in der Regel von verbindungsloser Paketvermittlung ausgehen, und zwar soll dabei keine Lastteilung stattfinden - m. a. W. schickt ein Knoten A alle Nachrichten mit Zieladresse Z immer zu demselben Nachbarn B, bis wegen einer Netzänderung (z. B. Ausfall einer Kante) ein anderer Nachbar an die Stelle von B tritt.

Viele der erzielten Ergebnisse lassen sich leicht auf andere Netztypen übertragen.

**Baum kürzester Wege**

Gegeben sei ein Netz $(G, w)$. Eine naheliegende erste Idee ist es, für jede Ecke $e$ als Zielecke einen **Baum kürzester Wege** $B_e$ (vgl. Kapitel 6) zu erstellen und jede Nachricht entsprechend ihrer Zieladresse $e$ so zu routen, wie der Baum $B_e$ dies vorschreibt.

**Beispiel 8.1-2:**
Für die in Beispiel 8.1-1 verwendete Netzstruktur haben alle Kanten die Bewertung 1. Für das Ziel F zeigt das folgende Bild einen möglichen Baum kürzester Wege $B_F$:




Danach würde also B jede Nachricht für F an D schicken usw..

Dieses Verfahren versagt natürlich, wenn der Anspruch besteht, auch im Falle von Störungen das Netz leistungsfähig zu erhalten: fällt im obigen Beispiel etwa die Kante BD aus, so kann B keine Nachrichten mehr an D schicken. Daher muss entweder die Möglichkeit vorgesehen werden, bei einer Netzänderung kürzeste Wege neu zu berechnen und dann diese zu nutzen, oder aber es müssen Zweitwege (bzw. mehrere Ersatzwege) von vornherein vorgesehen und gespeichert werden. Wie sich zeigen wird, bergen beide Varianten eine Vielzahl theoretischer und praktischer Probleme. Man kann sagen, dass hier der eigentliche Kern des Routing-Problems liegt.

**zentrales, verteiltes Routing**

In der Literatur findet man eine Reihe von Möglichkeiten, Routing-Verfahren zu klassifizieren. So kann man etwa von **zentralem** oder **verteiltem** Routing sprechen je nachdem, ob die Auswahlen von einem zentralen Knoten getroffen werden oder aber die einzelnen Knoten rechnen, entscheiden und unter Umständen Informationen über ihre Entscheidung an ihre Umgebung weitergeben. Eine andere Einteilung

**statisches, adaptives Routing**

ist die in **statische** und **adaptive** Routing-Algorithmen: im statischen Fall bleibt ein für ein Ursprung-Ziel-Paar gewählter Weg in Gebrauch, solange er nur intakt ist, während adaptives Routing auch bei übermäßiger Belastung (die u. U. zu großen Verzögerungen führt) auf andere Wege umschaltet.

Man muss sagen, dass diese Begriffe (und weitere, die in der Literatur zu finden sind) nicht ganz scharf definiert und voneinander zu trennen sind. Auch können konkrete Implementierungen eines Routing-Algorithmus sehr verschieden aussehen und durchaus gegensätzliche Eigenschaften (im Sinne der obigen Begriffe) aufwei-

sen. Die real existierenden Netze, die wir an späterer Stelle betrachten werden, sind bezüglich ihrer Routing-Verfahren größtenteils ebenfalls Mischformen.

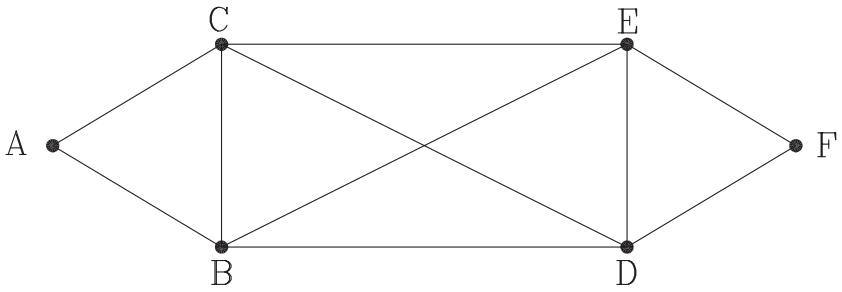
Im nächsten Abschnitt werden wir uns einigen grundlegenden Problemen des Routing mit kürzesten Wegen widmen und insbesondere einen häufig verwendeten Algorithmus für verteiltes adaptives Routing untersuchen.

Wir werden diesen Abschnitt abschließen, indem wir zwei ganz simple Routing-Verfahren vorstellen, das **random routing** und das **flooding**. Varianten des "flooding" werden uns an anderer Stelle noch begegnen; meist wird es angewendet, um in einem Netz Managment-Nachrichten zu übertragen, also z. B. die Information über den Ausfall einer Kante zu streuen.

**random routing**
**flooding**

Random Routing:
Jeder Knoten schickt eine nicht für ihn selbst bestimmte Nachricht zufällig an einen seiner Nachbarn.Ausgeschlossen ist dabei der Nachbar, von dem er die Nachricht erhalten hat.

Als Beispiel sei wieder folgendes Netz betrachtet:




Eine Nachricht von A nach F könnte z. B. den Weg $\{A \to B \to C \to E \to B \to C \to D \to E \to F\}$ nehmen oder auch (mit verschwindend geringer Wahrscheinlichkeit) niemals ankommen.

Flooding: Ein Knoten X schickt eine nicht für ihn bestimmte Nachricht, die er von Y bekam, an alle seine Nachbarn außer Y. Damit die so ausgelöste "Flut" irgendwann stoppt, tragen die Nachrichten einen Zähler; Knoten X erzeugt nur dann neue Nachrichten (mit um 1 erhöhtem Zählerwert), wenn der Zähler der bei ihm eingegangenen Nachricht unter einem Schwellwert lag. Der die Nachrichtenflut initiierende Knoten setzt den Zähler auf 1.

Will (wieder im schon oben als Beispiel gewählten Netz) A eine Mitteilung an F schicken, so führt das mit Schwellwert 3 zu insgesamt 26 Nachrichten: A schickt an B und C eine Nachricht mit Zählerwert 1. Mit Wert 2 werden Nachrichten von B an C,D,E und von C an B,D,E gesendet. Schließlich ergeben sich folgende 18 Nachrichten mit Zählwert 3: B → A,D,E ; C → A,D,E ; D → B,C,2∗E,2∗F ; E → B,C,2∗E,2∗F.

**Selbsttestaufgabe 8.1-1:**

*Nennen Sie einige Gründe, warum das Problem der Wegeauswahl in Kommunikationsnetzen nicht durch die einfache Vorschrift gelöst werden kann, dass immer ein kürzester Weg ausgewählt werden soll.*

## 8.2 Algorithmen zur Bestimmung kürzester Wege

Das in diesem Abschnitt behandelte Thema hat zunächst einen rein mathematischen Aspekt: hier geht es um die Algorithmen zur effektiven Berechnung der kürzesten Wege.

Die konkrete Anwendung hat dann zwei weitere Aspekte. Erstens geht es darum festzulegen, an welcher Stelle im Netz (zentral oder verteilt) die kürzesten Wege berechnet und Informationen darüber bereitgehalten werden. Zweitens muss das Kommunikationsprotokoll genau festlegen, nach welchen Regeln Routing-Informationen (z. B. über den Verlauf kürzester Wege oder Netzänderungen aufgrund von Ausfällen) ausgetauscht werden.

Im weiteren Verlauf dieses Abschnitts werden die grundlegenden Algorithmen betrachtet, wobei der Aspekt der zentralen oder verteilten Realisierung automatisch mit ins Spiel kommt. Die anderen angesprochenen Themen werden dann in Abschnitt 8.3 und Abschnitt 8.3 behandelt.

Zunächst soll an die im ersten Teil des Kurses verwendete Terminologie erinnert werden.

**Netz**

Ist ein gerichteter Graph $G = (E, K)$ mit einer Kantenbewertung $w : K \to \mathbb{R}$ gegeben, so wird auch von dem **Netz**$(G, w)$ gesprochen. Der Einfachheit halber ist im folgenden $E = \{1, \ldots, n\}$ vorausgesetzt. Die Bewertung $w$ induziert eine Matrix $(w_{ij})$ mit

$$w_{ij} := \begin{cases} 0 & , \text{ falls } i = j \text{ ist} \\ w(ij) & , \text{ falls } G \text{ die Kante } ij \text{ enthält} \\ \infty & , \text{ sonst.} \end{cases}$$

**Algorithmus von Floyd-Warshall**

Wir beginnen mit dem **Algorithmus von Floyd-Warshall**, der bereits Kapitel 6 behandelt wurde, und zwar wählen wir zunächst die dort dargestellte Form.

Gegeben sei ein Netz $(G, w)$. Es werden kürzeste Wege zwischen je zwei Ecken bestimmt. Der Algorithmus bricht ab, sobald ein gerichteter Kreis negativer Länge gefunden wird.

**Algorithmus**: Algorithmus von Floyd-Warshall

```
begin
   for i = 1 to n do
         for j = 1 to n do
                  setze d^0 (i, j) := w^ij;
                  if i ≠ j then
                        if w_ij < ∞ then setze v_ij := i
                           else setze v_ij := ∞
                        else setze v_ij := 0;
   setze k := 0;
   while d^k (i,i) ≥ 0 für alle i und k < n
               do
         begin
               setze k := k + 1;
               for i = 1 to n do
                     for j = 1 to n do
                        if d^{k-1} (i, k) + d^{k-1} (k, j) < d^{k-1} (i,j)
                           then setze v_ij := v_kj
                           und d^k (i, j) := d^{k-1} (i, k) +
                           d^{k-1} (k, j)
                           else setze d^k (i, j) := d^{k-1} (i, j);
         end
end
```

Da es von den hier betrachteten Anwendungen her ohnehin nahegelegt ist, werden wir der Einfachheit halber im folgenden stets voraussetzen, dass in einem gegebenen Netz keine Zyklen negativer Länge existieren.

Der Algorithmus von Floyd-Warshall vereinfacht sich dann zu folgender Form:

**Algorithmus**: Algorithmus von Floyd-Warshall

```
begin
      for i = 1 to n do
         for j = 1 to n do
            setze d^0 (i,j) := w_ij;
            if i ≠ j then
                     if w_ij < ∞ then setze v_ij := i
               else setze v_ij := ∞
                  else setze v_ij := 0;
      for k = 1 to n do
         for i = 1 to n do
            for j = 1 to n do
               if d^{k-1} (i, k) + d^{k-1} (k, j) < d^{k-1} (i,j)
                  then setze v_ij := v_kj
                     und d^k (i, j) := d^{k-1} (i, k) + d^{k-1} (k, j)
                  else setze d  (i, j) := d^{k-1} (i,j);
end
```

Wir gehen noch einen Schritt weiter und verzichten darauf, die kürzesten Wege im Algorithmus mitzunotieren. Es werden also nur die kürzesten Abstände bestimmt. Dies hat hier wie auch in den weiteren Ausführungen den Vorteil, dass die Algorithmen einfacher und ihre unterschiedlichen Strukturen deutlicher werden.

Der Algorithmus von Floyd-Warshall stellt sich nun wie folgt dar (vgl. in Kapitel 6):

**Algorithmus**: Algorithmus von Floyd-Warshall

```
begin
      for i = 1 to n do
            for j = 1 to n
                  setze d^0 (i, j) := w_ij;
      for k = 1 to n
            for i = 1 to n do
                  for j = 1 to n
                     setze
                     d^k (i, j)    :=    min { d^{k-1} (i, j), d^{k-1} (i, k) +
d^{k-1} (k, j) };
end
```

Der Beweis der Korrektheit des Algorithmus von Floyd-Warshall wurde in Satz 6.1-5 geführt. Grundidee dieses Beweises ist eine Iteration über die Ecken, die als Zwischenknoten für kürzeste Wege verwendet werden dürfen. Im Gegensatz dazu iteriert der **Algorithmus von Dijkstra** (von einer festen Ecke $s$ ausgehend) über die Länge kürzester Wege, die von $s$ aus zu anderen Ecken führen:

**Algorithmus von Dijkstra**

**Algorithmus von Dijkstra** Es gelte $w(k) \geq 0$ für jede (gerichtete) Kante $k$. Es werden die Abstände von 1 zu allen anderen Ecken bestimmt. (Dabei muss nicht jede Ecke von 1 aus erreichbar sein.)

**Algorithmus**: Algorithmus von Dijkstra

```
]
begin
      Setze d(1) := 0, T := E;
      for e ∈ E \ { 1 } do setze d(e) := ∞;
      while T ≠ ∅ do
      begin
            finde ein f ∈ T, für das d(f) minimal ist;
            setze T := T \ { f };
            for e ∈ T do
            setze d(e) := min { d(e), d(f) + w_fe };
      end
end
```

Bei den bisher behandelten Algorithmen von Floyd-Warshall und von Dijkstra haben wir stillschweigend angenommen, dass in einer konkreten Anwendung die den Algorithmus anwendende Instanz über alle relevanten Informationen verfügt, die das Netz betreffen. Bei dem **Algorithmus von Bellman-Ford**, der nun als nächstes vorgestellt werden soll, gehen wir zunächst auch von dieser Annahme aus. In einem zweiten Schritt werden wir dann eine dezentrale Variante dieses Algorithmus behandeln, bei der die beteiligten Instanzen nur über Teilinformationen verfügen müssen. Zuerst ist jedoch eine gründliche Analyse des Algorithmus von Bellman-Ford notwendig.

**Algorithmus von Bellman-Ford**

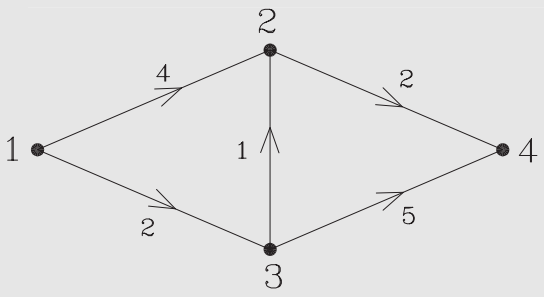
**Algorithmus von Bellman-Ford** Gegeben sei ein Netz $(G, w)$ ohne Kreise negativer Länge. Es werden die Abstände von $1$ zu allen anderen Ecken bestimmt. (Dabei muss nicht jede Ecke von $1$ aus erreichbar sein.)

**Algorithmus**: Algorithmus von Bellman-Ford

```
begin
   for e = 1 to n do setze d^1 (e) := w_1e;
   for k = 2 to n - 1 do
      for e = 1 to n do
         setze d^k (e) := min { d^{k-1} (e), min { d^{k-1}(f) + w_fe | f ≠ e } };
end
```

Beim Algorithmus von Bellman-Ford verläuft die Iteration über die Anzahl der Kanten, die in den Wegen verwendet werden dürfen. Dem Beweis der Korrektheit des Algorithmus stellen wir zwei Beispiele voraus.

**Beispiel 8.2-1:**
Es sei das im folgenden Bild dargestellte Netz gegeben:

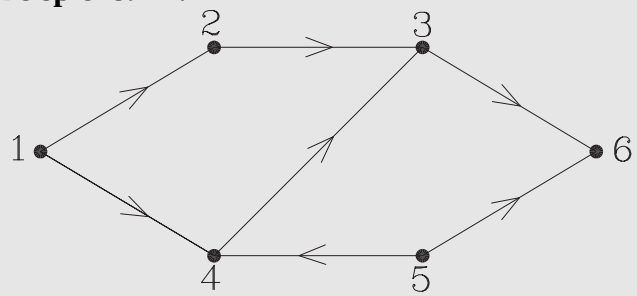


Die Initialisierung ergibt $d^1(1) = 0$, $d^1(2) = 4$, $d^1(3) = 2$, $d^1(4) = \infty$. Danach hat man in der for-Schleife nacheinander die folgenden Werte:

$$
\begin{array}{rll}
k = 2: & d^2(1) & = 0 \\
 & d^2(2) & = d^1(3) + 1 = 3 \\
 & d^2(3) & = d^1(3) = 2 \\
 & d^2(4) & = d^1(2) + 2 = 6 \\
k = 3: & d^3(1) & = 0 \\
 & d^3(2) & = d^2(2) = 3 \\
 & d^3(3) & = d^2(3) = 2 \\
 & d^3(4) & = d^2(2) + 2 = 5
\end{array}
$$

Für jedes $e \in \{2, 3, 4\}$ gibt nun $d^3(e)$ den Abstand $|1, e|$ an.

**Beispiel 8.2-2:**




Für das hier dargestellte Netz sei für alle Kanten die einheitliche Bewertung $1$ angenommen.

Man hat also zunächst
$d^1(1) = 0$, $d^1(2) = 1$, $d^1(3) = \infty$, $d^1(4) = 1$, $d^1(5) = \infty$, $d^1(6) = \infty$.

Die Iterationen liefern:

$$
\begin{aligned}
k=2: & d^2(1) && =0\\
& d^2(2)=d^1(2) && =1\\
& d^2(3)=d^1(2)+1 && =2\\
& d^2(4)=d^1(4) && =1\\
& d^2(5) && =\infty\\
& d^2(6) && =\infty\\
k=3: & d^3(1) && =0\\
& d^3(2)=d^2(2) && =1\\
& d^3(3)=d^2(3) && =2\\
& d^3(4)=d^2(4) && =1\\
& d^3(5) && =\infty\\
& d^3(6)=d^2(3)+1 && =3
\end{aligned}
$$

Für $k=4$ und $k=5$ ändern sich diese Werte nicht mehr, so dass bereits hier die korrekten Abstände gefunden sind.

Für die späteren Betrachtungen ist es nützlich, den Algorithmus von Bellman-Ford in der folgenden Form zu verwenden. Man beachte, dass sich die Zeile mit der Bestimmung von $d^k(e)$ dabei vereinfacht.

**Algorithmus**: Algorithmus von Bellman-Ford

```
begin
      for k = 0 to n − 1 do setze d^k(1) := 0;
      for e = 2 to n do setze d^0(e) := ∞;
      for k = 1 to n − 1 do
            for e = 2 to n do
                  setze d^k(e) := min_f {d^{k−1}(f) + w_fe };
end
```

Der Unterschied zu dem vorigen angegebenen Ablauf liegt erstens in der Zuweisung für $d^k(e)$; dies ist jedoch nicht relevant, da hier für $f=e$ gilt $d^{k-1}(f)+w_{fe}=d^{k-1}(e)$.
Ein zweiter Unterschied ist, dass $d^k(1)=0$ für alle $k$ von vornherein festgelegt ist - für die vorige Fassung in muss dies erst bewiesen werden.
Drittens ist es schließlich so, dass sich die Anfangsbedingungen $d^1(e)=w_{1e}$ hier erst nach dem ersten Iterationsschritt ergeben.

Es soll nun die Korrektheit des Algorithmus von Bellman-Ford nachgewiesen werden:

**Satz 8.2-1:** *Der Algorithmus von Bellman-Ford bestimmt die Abstände der Ecke* $1$ *zu allen anderen Ecken. Die Komplexität ist* $0(n^3)$.

**Beweis:** Wir führen den Beweis für den zuletzt beschriebenen Ablauf. Es wird mit Induktion gezeigt, dass $d^k(e)$ die Länge eines kürzesten Weges von $1$ nach $e$ ist, der höchstens $k$ Kanten enthält. Für $k = 0$ ist dies offensichtlich. Es gelte nun die Induktionsvoraussetzung, dass für ein beliebiges festes $k \geq 0$ für jede Ecke $e$ die Zahl $d^k(e)$ die Länge eines kürzesten Weges von $1$ nach $e$ mit höchstens $k$ Kanten angibt. Eine Ecke $e$ sei jetzt ebenfalls fest gewählt. Es ist zu zeigen, dass unter diesen Voraussetzungen

$$d^{k+1}(e) = \min_f \{d^k(f) + w_{fe}\}$$

gleich der Länge $L$ eines kürzesten Weges von $1$ nach $e$ mit höchstens $k+1$ Kanten ist.
Es sei $1, \ldots, g, h, e$ ein (kürzester) Weg der Länge $L$ von $1$ nach $e$ mit höchstens $k+1$ vielen Kanten. Offensichtlich gilt

$$L = L' + w_{he},$$

wobei $L'$ die Länge des Weges $1, \ldots, g, h$ bezeichnet. Da dieser Weg höchstens aus $k$ vielen Kanten besteht, folgt nach Induktionsvoraussetzung

$$L' \geq d^k(h),$$

also

$$L \geq d^k(h) + w_{he} \geq \min_f \{d^k(f) + w_{fe}\} = d^{k+1}(e).$$

Es muss also nur noch die Ungleichung $d^{k+1}(e) \geq L$ gezeigt werden. Dazu sei die Ecke $x$ so gewählt, dass $d^k(x) + w_{xe}$ minimal wird, also dass

$$d^{k+1}(e) = d^k(x) + w_{xe}$$

gilt. $1, \ldots, y, x$ sei ein kürzester Weg von $1$ nach $x$ mit höchstens $k$ Kanten, seine Länge ist nach Induktionsvoraussetzung gleich $d^k(x)$. Ist nun $1, \ldots, y, x, e$ ein Weg (d. h. $e$ liegt nicht schon auf dem Weg $1, \ldots, y, x$), so hat dieser Weg die Länge $d^{k+1}(e)$, und es folgt

$$d^{k+1}(e) \geq L.$$

Liegt aber $e$ schon auf dem Weg $1, \ldots, y, x$, so liegt insgesamt ein Weg der Form

$$1, \ldots, z, e, \ldots, x, e$$

vor, wobei $e, \ldots, x, e$ einen gerichteten Kreis $K$ der Länge $0$ bilden muss. $x$ kann nun in der Argumentation gegen $z$ ausgetauscht werden, denn es ist auch

$$d^{k+1}(e) = d^k(z) + w_{ze},$$

und mit Induktion kann schließlich vorausgesetzt werden, dass $e$ nicht schon auf dem Weg $1, \ldots, z$ liegt, womit dann wie oben argumentiert werden kann.

Es bleibt die Komplexitätsaussage zu zeigen. Da $(G, w)$ keine negativen Kreise enthält, kann ein kürzester Weg höchstens $n-1$ viele Kanten enthalten. Nach dem Ablauf des Algorithmus geben demnach die Werte $d^{n-1}(e)$ die kürzesten Abstände an. Da der Algorithmus aus zwei ineinandergeschachtelten Schleifen besteht, in deren innerer $0(n)$ viele Vergleiche nötig sind, ist die Komplexität $0(n^3)$.

Es sind zwei weitere Bemerkungen zum Ablauf des Algorithmus von Bellman-Ford angebracht.

Wenn sich für ein $k \leq n-3$ ergibt, dass $d^{k+1}(e) = d^k(e)$ für alle $e$ gilt, so wird auch für die noch folgenden $j$ mit $k+1 < j \leq n-1$ stets $d^j(e)$ jeweils denselben Wert liefern, m. a. W. werden durch $d^k(e)(e = 2, \ldots, n)$ bereits die gesuchten Abstände angegeben. Dass dies so ist, kann im Algorithmus unmittelbar abgelesen werden. In Beispiel 8.2-2 tritt dieses Phänomen auf.

Der Algorithmus von Bellman-Ford kann beschleunigt werden, wenn in der Zeile

$$d^k(e) := \min_f \{d^{k-1}(f) + w_{fe}\}$$

statt $d^{k-1}(f)$ schon der Wert $d^k(f)$ verwendet wird, sofern dieser zuvor bereits berechnet wurde (sofern also $f < e$ ist). In diesem Fall stimmt allerdings nicht mehr die Aussage, dass $d^k(e)$ stets die Länge eines kürzesten Weges mit höchstens $k$ Kanten von $1$ nach $e$ ist. Löst man sich auch noch von der Numerierung der Ecken, so kommt man zu der folgenden eleganten Formulierung des Algorithmus von Bellman-Ford:

**Algorithmus**: Algorithmus von Bellman-Ford

```
begin
      Setze d(1) := 0;
      for e ∈ E \ { 1 } do setze d(e) := ∞;
      repeat
            for e ∈ E do setze d'(e) := d(e);
            for e ∈ E do setze
                        d(e) := min_f { d(f) + w_fe };
      until d(e) = d'(e) für alle e ∈ E;
end
```

Die Korrektheit dieser Variante wollen wir jetzt nicht beweisen, da sie eine Konsequenz aus den im folgenden betrachteten Eigenschaften der dezentralisierten Form des Algorithmus von Bellman-Ford sein wird. Es ergibt sich dann auch folgende Aussage: Sind die Bewertungen aller Kanten positiv, so kann bei der Initialisierung für jeden der Werte $d(e)$ statt "$\infty$" sogar jede beliebige nichtnegative Zahl genommen werden.

**Beispiel 8.2-3:**
Der abgewandelte Algorithmus von Bellman-Ford wird auf das Beispiel 8.2-1 angewendet.
Der erste Schritt ergibt $d(1) = 0$, $d(2) = 4$, $d(3) = 2$, $d(4) = \infty$.
2. Durchlauf der repeat-Schleife:

$$d(1) = 0 \quad d(2) = d(3) + 1 = 3 \quad d(3) = 2 \quad d(4) = d(2) + 2 = 5$$

Da sich diese Werte beim 3. Durchlauf der repeat-Schleife nicht mehr ändern, sind die gesuchten Abstände bereits gefunden.

**Bellmansche Gleichungen**

An der zuletzt präsentierten Form des Algorithmus von Bellman-Ford lässt sich sofort die Gültigkeit der **Bellmansche Gleichungen** ablesen: wenn obiger Algorithmus abbricht, so gilt für die zuletzt erhaltenen d-Werte

$$d(1) = 0 \qquad \text{und} \qquad d(e) = \min_f \{d(f) + w_{fe}\}.$$

In Satz 6.1-2 wurde umgekehrt gezeigt: enthält $G$ nur Kreise positiver Länge, und gelten für Werte $\tilde{d}(e)$ $(e \in E)$ die Bellmanschen Gleichungen, so gibt zwangsläufig $\tilde{d}(e)$ für jedes $e \in E$ den Abstand von 1 nach e an. Dies ergibt unter diesen speziellen Vorraussetzungen an $G$ einen Beweis der Korrektheit des oben in Alg. dargestellten Algorithmus freilich nur, wenn man schon weiß, dass der Algorithmus stets abbricht.

Wir gehen jetzt von dem folgenden Szenario aus: Ein Datennetz, modelliert durch ein Netz $(G, w)$, ist gegeben. Es wird das Problem des Routens von Nachrichten zum Knoten 1 betrachtet, und zwar sollen grundsätzlich kürzeste Wege benutzt werden. Die Komplikation liegt nun darin, dass sich die Kantenbewertungen $w_{ij}$ laufend ändern dürfen. Dies ergibt z. B. dann einen Sinn, wenn ein Wert $w_{ij}$ die augenblickliche Auslastung einer Leitung widerspiegelt; die Wahl eines kürzesten Weges stellt so eine Möglichkeit dar, eine derzeit wenig belastete Route zu nutzen.

Für die soeben skizzierte Situation lauten die Bellmanschen Gleichungen, wenn $d(e)$ den (kürzesten) Abstand der Ecke $e$ zu $1$ darstellt,

$$d(1) = 0 \qquad \text{und} \qquad d(e) = \min_{f \in N(e)} \{w_{ef} + d(f)\}.$$

(Wie an früherer Stelle bezeichnet $N(e)$ die Menge der Nachfolgerecken von e.)

Man bekommt nun für den Algorithmus von Bellman-Ford die Iteration

$$d^{k+1}(1) = 0 \qquad \text{und} \qquad d^{k+1}(e) = \min_{f \in N(e)} \{w_{ef} + d^k(f)\}.$$

Diese Iteration kann natürlich parallel ausgeführt werden in dem Sinne, dass jeder Knoten $e$ den Wert $d^{k+1}(e)$ nach obiger Formel berechnet. Freilich muss $e$ dazu die Bewertungen $w_{ef}$ (für jeden Nachfolger $f \in N(e)$) sowie die Werte $d^k(f)$ kennen. Eine Lösung könnte also so aussehen, dass die Information über $d^k(f)$ von $f$ an $e$ weitergegeben wird; mit der Art dieser Weitergabe wollen wir uns erst in Abschnitt 8.4 beschäftigen. Die Bewertungen $w_{ef}$ – davon gehen wir aus – mögen von $e$ selbst ablesbar sein (z. B. durch Messungen).

Ein Vorteil einer derart dezentralen Vorgehensweise ist es auch, dass einem Knoten $e$ außer seinen Nachfolgern $f \in N(e)$ (und den Werten $w_{ef}$) nur die Existenz des Knotens 1 bekannt sein muss, nicht jedoch die gesamte Netztopologie (Graphenstruktur). In einer konkreten Anwendung muss das gesamte Verfahren natürlich für jeden Zielknoten (nicht nur für 1) zum Einsatz kommen; ein einzelner Knoten muss dann demnach neben der Kenntnis seiner unmittelbaren Umgebung nur von der Existenz aller anderen Knoten ausgehen.

Bei einer Realisierung einer wie soeben beschriebenen dezentralen Variante des Algorithmus von Bellman-Ford ergibt sich allerdings die Schwierigkeit der

**Synchronität**

**Synchronität**: einerseits muss (von den Anfangsbedingungen $d^0(1) = 0$ und $d^0(e) = \infty$ ausgehend) für das gesamte Netz ein gemeinsamer Start des Algorithmus organisiert werden; andererseits muss bei Änderung eines $w_{ij}$-Wertes ein Abbruch und gemeinsamer Neustart erfolgen. Zwar lassen sich die Schwierigkeiten unter gewissen Bedingungen bewältigen, jedoch hat man es dann insgesamt mit einem wesentlich komplexeren Algorithmus zu tun.

Eine einfachere Alternative besteht darin, auf die Synchronität zu verzichten und beliebige Anfangswerte $d^0(e)$ zuzulassen. Das Problem des Starts bzw. Neustarts des Algorithmus tritt dann nicht mehr auf.

**verteilter asynchroner Bellman-Ford Algorithmus**

Bevor der **verteilte asynchrone Bellman-Ford Algorithmus** genau beschrieben und seine Korrektheit nachgewiesen wird, soll die ihm zugrundeliegende Idee grob umschrieben werden:

Der Algorithmus läuft im Prinzip unbeschränkt lange so ab, dass von Zeit zu Zeit in jedem Knoten $e \neq 1$ die Iteration

$$d(e) = \min_{f \in N(e)} \{w_{ef} + d(f)\} (*)$$

ausgeführt wird. Dabei werden die zuletzt von den Nachbarn $f \in N(e)$ mitgeteilten Werte $d(f)$ und die zuletzt festgehaltenen Werte $w_{ef}$ benutzt. Eine Notwendigkeit ist, dass jeder Knoten $e$ von Zeit zu Zeit seinen aktuellen Schätzwert $d(e)$ an seine Nachbarn weitergibt. Es müssen weder die Iterationen $(*)$ noch die gegenseitigen Benachrichtigungen über die $d\ (e)$-Werte in irgendeiner Weise synchron für verschiedene Knoten ablaufen.

Wie gezeigt werden wird, ist dieser völlig asynchrone Algorithmus korrekt, und zwar in dem folgenden Sinne:

Gibt es nach einem Zeitpunkt $t_0$, zu dem in den Knoten $e \neq 1$ beliebige nichtnegative Schätzwerte $d\ (e)$ vorliegen, keine weiteren Änderungen der $w_{ij}$-Werte mehr, dann hat nach endlicher Zeit (von $t_0$ aus gerechnet) jeder Knoten $e$ als $d\ (e)$ den korrekten kürzesten Abstand zum Knoten 1 bestimmt.

Dies bedeutet auch, dass bei einer $w_{ij}$-Änderung kein neuer Start des Algorithmus nötig ist, denn man hat nichts anderes als einen neuen Start-Zeitpunkt $t_0$ und weiß mit dem obigen Argument, dass die weiteren Iterationen in allen Knoten wieder zum korrekten Ergebnis laufen.

Nun soll der verteilte asynchrone Bellman-Ford Algorithmus präzise formuliert und seine Korrektheit nachgewiesen werden.

Man geht davon aus, dass ein beliebiger Knoten $e \neq 1$ zu einem beliebigen Zeitpunkt $t$ die folgenden Werte verfügbar hat:

| | |
|---|---|
| $d_f^e(t)$ : | Schätzwert für kleinsten Abstand des Nachbarn $f \in N(e)$ zum Knoten 1, der zuletzt von $f$ an $e$ übermittelt wurde |
| $d_e(t)$ : | Schätzwert für kleinsten Abstand von $e$ zum Knoten 1,der zuletzt mit der Bellman-Ford Iteration im Knoten $e$ berechnet wurde. |

Alle Schätzwerte für den Zielknoten 1 sind gleich Null, also

$$d_1(t) = 0 \qquad \text{für} \quad t \geq t_0$$
$$d_1^e(t) = 0 \qquad \text{für} \quad t \geq t_0 \qquad \text{und} \quad 1 \in N(e)$$

Ferner wird davon ausgegangen, dass ein Knoten e auch alle Bewertungen $w_{ef}$ mit $f \in N(e)$ kennt, wobei $w_{ef}$ für $t \geq t_0$ als positive Konstante angenommen wird.

Eine weitere grundsätzliche Annahme ist nun, dass sich die Schätzwerte für die Abstände nur zu bestimmten Zeitpunkten $t_0, t_1, t_2, ...$ mit $t_{m+1} \geq t_m$ und $t_m \to \infty$ für $m \to \infty$ ändern. Zu diesen Zeitpunkten findet in jedem Knoten eines der folgenden drei Ereignisse statt:

1. Knoten $e$ aktualisiert $d_e(t)$ nach der Formel

   $$d_e(t) = \min_{f \in N(e)} \{w_{ef} + d_f^e(t)\}$$

   und lässt die Schätzwerte $d_f^e(t) \quad (f \in N(e))$ unverändert.

2. Knoten $e$ empfängt von einem oder mehreren Nachbarn $f \in N(e)$ den dort zu einem früheren Zeitpunkt berechneten Wert für $d_f$, aktualisiert den Schätzwert für $d_f^e$ und belässt die übrigen Schätzwerte.

3. Knoten $e$ bleibt inaktiv und lässt alle in $e$ verfügbaren Schätzwerte unverändert.

Während die soeben beschriebene Annahme im wesentlichen dazu dient, das Problem zu "diskretisieren", muss schließlich noch sichergestellt werden, dass in den Knoten die Schätzwerte gegen die wirklichen Abstände konvergieren. Dazu dienen die folgende Annahmen:

**Annahme 1:** Ein Knoten $e$ hört niemals auf, seine Schätzwerte zu aktualisieren; mit anderen Worten: zu unendlich vielen der obigen Zeitpunkte $t_k$ führt $e$ die wie in 1. beschriebene Aktualisierung aus. ($T^e$ sei die Menge dieser Zeitpunkte.)

**Annahme 2:** Knoten $e$ empfängt von jedem Nachbarn $f \in N(e)$ zu unendlich vielen der obigen Zeitpunkte $t_k$ (wie in 2. beschrieben) den dort berechneten Wert für $d_f$. (Die Menge dieser Zeitpunkte sei $T_f^e$.)

**Annahme 3:** Alle Initialschätzwerte $d_e(t_0)$ und $d_f^e(t_0) \quad (f \in N(e))$ sind nicht negativ. Dies gilt auch für alle vor $t_0$ verschickten, jedoch erst nach $t_0$ erhaltenen Schätzwerte von Nachbarn.

**Annahme 4:** Ein "alter" Schätzwert kann sich nicht unendlich lange im System aufhalten, anders gesagt: für jeden Zeitpunkt $\bar{t} \geq t_0$ gibt es ein $\tilde{t} \geq \bar{t}$ derart, dass vor dem Zeitpunkt $\bar{t}$ im Knoten $f$ berechnete Schätzwerte $d_f$ nach dem Zeitpunkt $\tilde{t}$ von keinem Nachbarn $e(f \in N(e))$ mehr empfangen werden.

Nachdem alle Annahmen präzise formuliert sind, sind wir nunmehr in der Lage, die Korrektheit des verteilten asynchronen Algorithmus von Bellman-Ford nachzuweisen, d. h. dass die Schätzwerte $d_e(t)$ nach endlicher Zeit die korrekten Abstandswerte annehmen:

**Satz 8.2-2:** *Es gibt einen Zeitpunkt $t_m$, so dass für jeden Knoten $e$ gilt*

$$d_e(t) = d_e \text{ für } t \geq t_m$$

*Dabei ist mit $d_e$ der korrekte kürzeste Abstand von $e$ zum Knoten 1 bezeichnet.*

Dem Beweis des Satzes müssen einige Hilfsüberlegungen vorangestellt werden.

Zunächst halten wir fest, dass die Bellman-Ford Iteration offenbar die folgende Monotonieeigenschaft hat:

gelten für einen Knoten $e$ und reelle Zahlen $\bar{d}_f$ bzw.

$\tilde{d}_f \quad (f \in N(e))$ die Ungleichungen
$\bar{d}_f \geq \tilde{d}_f \quad (f \in N(e))$,
so bleibt die Ungleichung durch die Iteration erhalten,
d. h.
$\min\limits_{f \in N(e)} \{w_{ef} + \bar{d}_f\} \geq \min\limits_{f \in N(e)} \{w_{ef} + \tilde{d}_f\}$.

Als nächstes stellen wir uns vor, es würde der in obene beschriebene synchrone Bellman-Ford Algorithmus mit der Iterationsregel

$$d^{k+1}(1) = 0, \qquad d^{k+1}(e) = \min_{f \in N(e)} \{w_{ef} + d^k(f)\}$$

zweimal ablaufen, und zwar

1. Einmal mit den Initialwerten
   $d_1^0 = 0$ und $d_e^0 = \infty \quad (e \neq 1)$,
2. einmal mit den Initialwerten
   $d_e^0 = 0$ für alle Knoten $e$.

Die im ersten Fall erhaltenen Werte $d^k(e)$ bezeichnen wir mit $\bar{d}_e^k$, die beim zweiten Ablauf erhaltenen mit $\underline{d}_e^k$.

Im weiteren wird nun gezeigt, dass die aus diesen beiden extremen Anfangsbedingungen abgeleiteten Folgen alle anderen solchen Folgen –auch eines asynchronen Ablaufs– sozusagen "einfangen". Zunächst benötigen wir den folgenden

**Hilfssatz 8.2-1:** *Für die soeben definierten Folgen $(\bar{d}_e^k)$ und $(\underline{d}_e^k)$ in einem beliebigen Knoten $e$ gilt*

$$\underline{d}_e^k \leq \underline{d}_e^{k+1} \leq d_e \leq \bar{d}_e^{k+1} \leq \bar{d}_e^k$$

*für jedes $k$, und ab einem gewissen $k_0$ hat man für $k \geq k_0$ sogar*

$$\underline{d}_e^k = d_e = \bar{d}_e^k.$$

*(Mit $d_e$ ist wieder der korrekte Abstand von $e$ nach 1 bezeichnet.)*

**Beweis:** Die Gültigkeit der Ungleichungskette

$$\underline{d}_e^k \leq \underline{d}_e^{k+1} \leq d_e \leq \bar{d}_e^{k+1} \leq \bar{d}_e^k$$

folgt leicht mit Induktion, wenn man die Monotonieeigenschaft der Bellman-Ford Iteration und die gewählten Anfangsbedingungen ausnutzt.
Weiter stellen wir fest, dass aus dem Beweis des Satzes Satz 8.2-1 sofort
$\bar{d}_e^k = d_e$ für $k \geq n-1$
geschlossen werden kann (n ist die Gesamtzahl der Knoten), denn die dortige Initialisierung entspricht derjenigen für die $\bar{d}$-Werte ($d_1^0 = 0$ und $d_e^0 = \infty$ für $e \neq 1$).
Somit bleibt nur noch zu zeigen, dass

$$\underline{d}_e^k = d_e$$

für genügend großes $k$ gilt. Zu diesem Zweck macht man sich zunächst klar, dass (für jedes $e$ und $k$) $\underline{d}_e^k$ die Summe von höchstens $k$ vielen Kantenbewertungen $w_{ij}$ ist. Dies ist mit Induktion leicht einzusehen und soll hier nicht detailliert begründet werden. Wegen der Ungleichung

$$\underline{d}_e^k \leq d_e$$

kann man aber allgemein sagen, dass auch (für beliebiges $k$) $\underline{d}_e^k$ die Summe von nicht mehr als

$$\frac{\max\limits_e d_e}{\min\limits_{(i,j)} w_{ij}}$$

vielen Kantenlängen sein kann. Daher ist die Anzahl aller möglichen Werte für $\underline{d}_e^k$ (mit beliebigen $e$ und $k$) nur endlich. Da (für jedes $e$) die Folge $(\underline{d}_e^k)$ monoton fällt, muss für ein $k'$ gelten $\underline{d}_e^{k'} = \underline{d}_e^{k'+1}$ für alle $e$.

Damit erfüllen die Zahlen $\underline{d}_e^{k'} (e \in E)$ die Bellmanschen-Gleichungen, die die korrekten Abstände $d_e$ als einzige Lösung haben. Nun folgt natürlich auch $\underline{d}_e^k = d_e$ für $k \geq k'$, und der Hilfssatz ist gezeigt.

**Beweis:** ( von Satz 8.2-2)

Der Satz wird nun gezeigt, indem durch Induktion nachgewiesen wird, dass für jedes $k$ ein Zeitpunkt $t(k)$ mit den folgenden Eigenschaften existiert:
für alle $t \geq t(k)$ gilt

$$\begin{array}{lll} \underline{d}_e^k \leq & d_e(t) \leq & \bar{d}_e^k(e \in E), \\ \underline{d}_f^k \leq & d_f^e(t) \leq & \bar{d}_f^k(e \in E, f \in N(e), \end{array}$$

und für alle $t \in T_f^e$, $t \geq t(k)$ gilt

$$\underline{d}_f^k \leq d_f(\tau_f^e(t)) \leq \bar{d}_f^k(e \in E,\ f \in N(e));$$

dabei ist $\tau_f^e(t) \leq t$ der späteste Zeitpunkt, zu dem der im Knoten $e$ verfügbare Schätzwert $d_f^e(t)$ im Knoten $f$ gemäß der Iterationsvorschrift berechnet wurde. M.a.W. ist $\tau_f^e(t)$ der späteste Zeitpunkt in $T^f$, der nicht später als $t$ ist und $d_f(\tau_f^e(t)) = d_f^e(t)$ erfüllt.

Für $k = 0$ hat $t(0) = t_0$ die verlangten Eigenschaften. Dies liegt an der Annahme der Nichtnegativität für die Initialschätzwerte bzw. die später als $t_0$ empfangenen Schätzwerte (Annahme 3).

Jetzt wird angenommen, dass die Behauptung für ein gewisses festes $k$ richtig ist, d. h. ein $t(k)$ mit den obigen Eigenschaften existiert. Es muss daraus die Existenz eines entsprechenden $t(k+1)$ nachgewiesen werden.

Wegen $\underline{d}_f^k \leq d_f^e(t) \leq \bar{d}_f^k \quad (e \in E, f \in N(e))$ für $t \geq t(k)$ und der Monotonieeigenschaft der Bellman-Ford Iteration hat man nun zunächst, dass für $t \geq t(k)$ und $t \in T^e$ gilt

$$\underline{d}_e^{k+1} \leq d_e(t) \leq \bar{d}_e^{k+1}.$$

Ist demnach $t'(k)$ der früheste Zeitpunkt $t \in T^e$ mit $t \geq t(k)$, so hat man

$$\underline{d}_e^{k+1} \leq d_e(t) \leq \bar{d}_e^{k+1} \qquad (**)$$

für alle $t \geq t'(k)$ und $e \in E$.
Da (aufgrund der separat formulierten Annahmen) für $t \to \infty$ auch $\tau_f^e(t) \to \infty$ strebt, gibt es einen Zeitpunkt $t(k+1) \geq t'(k)$ derart, dass

$$\tau_f^e(t) \geq t'(k)$$

gilt für alle $e \in E\ ,\ \ f \in N(e)$ und $t \geq t(k+1)$.
Mit $(**)$ folgt nun für $t \geq t(k+1)$

$$\underline{d}_f^{k+1} \leq d_f^e(t) \leq \bar{d}_f^{k+1}(e \in E, f \in N(e)),$$

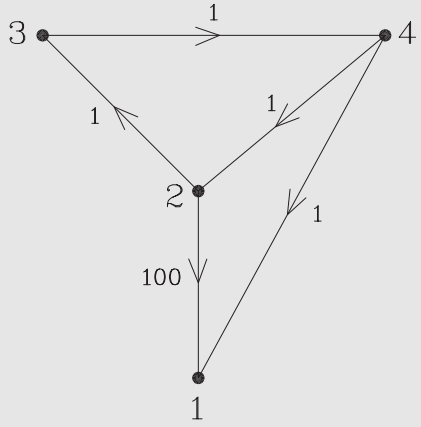
und für alle $t \in T_f^e$ mit $t \geq t(k+1)$ auch

$$\underline{d}_f^{k+1} \leq d_f(\tau_f^e(t)) \leq \bar{d}_f^{k+1}(e \in E,\ f \in N(e))$$

Damit ist der Beweis vollständig.

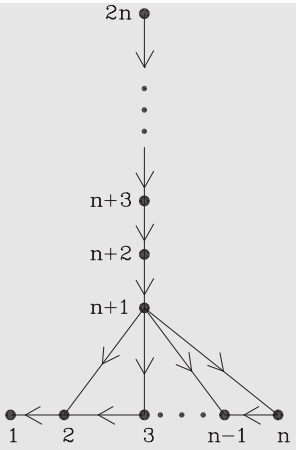
An späterer Stelle werden wir ein real existierendes Netz ansehen, das sich des verteilten asynchronen Bellman-Ford Algorithmus bedient, und auf seine spezifischen Stärken und Schwächen eingehen.
An dieser Stelle sollen zwei grundsätzliche Schwächen der Methode aufgezeigt werden. Es handelt sich allerdings um ganz spezielle (fast "exotische") Beispiele, die über die durchschnittliche Leistungsfähigkeit des Algorithmus wenig aussagen.

**Beispiel 8.2-4:**




Nimmt man an, dass zum Zeitpunkt $t_0$ die Kante $(4,1)$ ausfällt $(w_{41} = \infty)$ und davor in jedem Knoten der kürzeste Abstand zum Zielknoten $1$ bekannt war (dies sind also nun bei $t_0$ die Anfangsbedingungen), so wird Knoten 2 ca. 30 Iterationen benötigen, um festzustellen, dass die direkte Kante den kürzesten Weg nach $1$ darstellt. In diesem Beispiel wird also eine unnötig groß erscheinende Anzahl von Iterationsschritten benötigt.

**Beispiel 8.2-5:**
Bei diesem Beispiel ist eine große Anzahl von Nachrichten zwischen den Knoten nötig; dies liegt an einer ungünstigen Reihenfolge der relevanten Ereignisse in den beteiligten Knoten.




Alle Kantenlängen seien 1, alle Anfangsschätzwerte für die Abstände zum Knoten 1 gleich $\infty$.
Die folgende Folge von Ereignissen mit insgesamt $n^2 - 2$ vielen Nachrichten ist möglich:
Knoten 2 aktualisiert seinen Abstand (zum Knoten 1) und teilt das Ergebnis an 3 mit, dieser aktualisiert ebenfalls und meldet das Ergebnis an 4 etc., schließlich aktualisiert $n$ und gibt das Ergebnis an $n + 1$. Dies macht insgesamt $n - 1$ Nachrichten aus. Jetzt aktualisiert $n + 1$ und informiert darüber $n + 2$ etc., bis Knoten $2n$ aktualisiert. Auch das ergibt $n - 1$ viele Nachrichten. Nun informiert Knoten $e$, für $e = n - 1, n - 2, ..., 2$, jeweils $n + 1$ über seine Aktualisierung und löst die "Kettenreaktion" aus, dass dann $n + 1$ den Knoten $n + 2$ informiert usw.; das ergibt $n(n - 2)$ viele Nachrichten.

**Selbsttestaufgabe 8.2-1:**

*Erläutern Sie einige Vorteile des verteilten asynchronen Bellman-Ford Algorithmus.*

## 8.3 Das Stabilitätsproblem bei der Nutzung kürzester Wege

Wird in einem Kommunikationsnetz auf kürzesten Wegen "geroutet", und stellen die Kantenbewertung des Graphen die aktuellen Verkehrsbelastungen der einzelnen Verbindungsleitungen dar, so ergibt sich das folgende grundsätzliche Problem: Die Nutzung eines Weges hat die Tendenz, seine Länge zu vergrößern. Ist das Routing adaptiv (z. B. wie im vorigen Abschnitt beschrieben), so kann dies zu **Oszillationen** führen, d. h. es wird dauernd zwischen verschiedenen Wegen hin- und hergeschaltet, ohne dass der dazu benötigte Aufwand sich in einer erhöhten Transportleistung des Netzes niederschlagen würde.

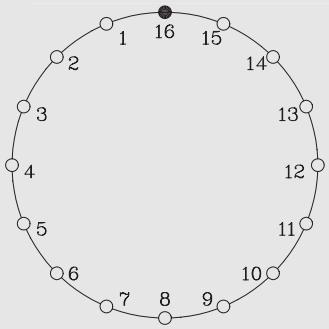
**Oszillation**

Im weiteren Verlauf dieses Abschnitts wird dieses Problem zunächst verdeutlicht, anschließend werden mögliche Gegenmaßnahmen skizziert. Es ist hier angezeigt, zwischen verbindungsloser und verbindungsorientierter Datenübermittlung zu unterscheiden.

**Verbindungslose Datenübermittlung**

In dem folgenden Beispiel wird der gesamte für ein gewisses Ziel bestimmte Verkehr ständig zwischen zwei möglichen Wegen hin- und hergeschaltet, obwohl das Verkehrsaufkommen für das Ziel nahezu völlig homogen über das Netz verteilt erzeugt wird.

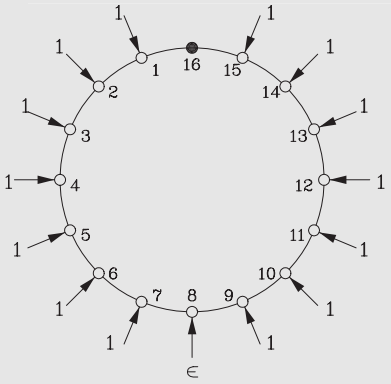
**Beispiel 8.3-1:**
Für das folgende ringförmige Netz mit 16 Knoten möge (paketförmiger) Verkehr nur für das Ziel 16 auftreten. Jede Kante kann in beiden Richtungen Verkehr tragen.

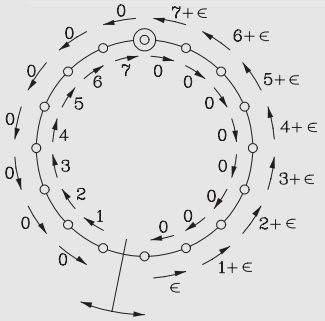



Wir machen folgende Annahmen:

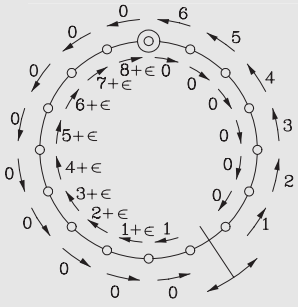
- In jedem der Knoten $1, 2, ..., 7, 9, ..., 15$ wird pro Sekunde eine Datenmenge gleicher Größe für das Ziel $16$ erzeugt, diese Größe sei auf "1" normiert. Im Knoten $8$ entsteht nur Verkehr von $\epsilon > 0$, wobei $\epsilon$ nahe bei $0$ liegt.
- Die Länge $w_{ij}$ einer Kante $ij$ ist gleich $V_{ij}$, wobei $V_{ij}$ den Datenverkehr auf der Kante $ij$ bezeichnet; dieser Verkehr kann von $i$ erzeugt sein, oder es kann sich um Transferverkehr handeln. (Man beachte: i.a. ist $w_{ij} \neq w_{ji}$).
- Jeder Knoten berechnet alle $T$ Sekunden seinen kürzesten Weg zum Ziel $16$, wobei als Kantenlängen die Werte $V_{ij}$ aus den vergangenen $T$ Sekunden genommen werden, und routet für die nächsten $T$ Sekunden entsprechend diesem kürzesten Weg. Eine weitere realistische Annahme besagt: $T$ ist mindestens so groß, dass innerhalb von $T$ Sekunden mehrere Pakete von einem Knoten verschickt werden, der eine Datenmenge von $1$ erzeugt.
- Zu Beginn routen die Knoten $1$ bis $7$ im Uhrzeigersinn, $8$ bis $15$ entgegen dem Uhrzeigersinn. (Dieses Routing ist sicher nicht das schlechteste, da es eine gewisse Balance zwischen den beiden Seiten des Rings beinhaltet.)

Das erste der folgenden Diagramme zeigt noch einmal das Netz mit seinem Muster des Verkehrsaufkommens. Die weiteren Bilder zeigen dann die Belastungen der Kanten (in jeweils beiden Richtungen) im Initialrouting und den nächsten Schritten, wobei jedes Routing aufgrund der obigen Regeln aus der zuletzt beobachteten Verkehrsstruktur abgeleitet wurde.

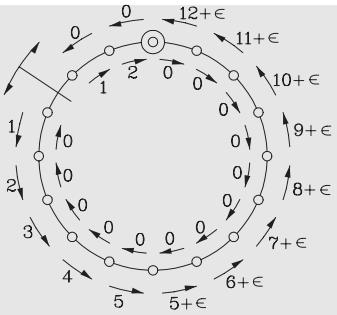




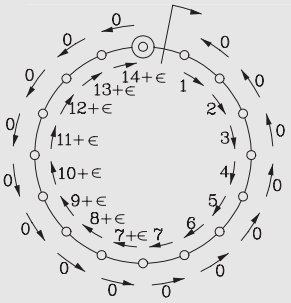
Initialrouting



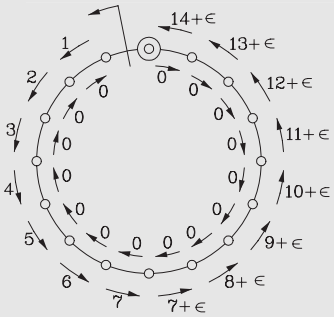
2. Routing



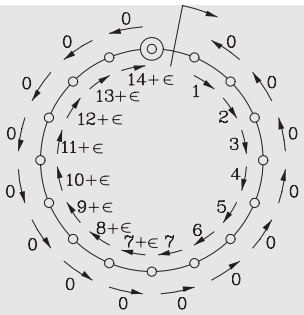

3. Routing




4. Routing




5. Routing




Wie man sieht, stellt sich nach drei Aktualisierungen ein ständiges Oszillieren zwischen zwei völlig gegensätzlichen Verkehrsmustern ein.

Bei dem hier vorliegenden Problem hat man es mit einem **Rückkopplungseffekt** (Feedback-Effekt) zu tun: die Verkehrsbelastungen der Kanten hängen vom gewählten Routing ab, und dieses richtet sich wiederum nach den aktuellen Verkehrsbelastungen. Es würde hier zu weit führen, tiefer in die Theorie solcher Phänomene einzusteigen, wie sie in der **Regelungs- und Steuerungstheorie** (Control theory) betrieben wird. Es soll an dieser Stelle nur erwähnt werden, dass das hier betrachtete Stabilitätsproblem gemildert werden kann, wenn ein **Biasfaktor** eingeführt wird: dies ist einfach eine Konstante $\alpha > 0$, die zu allen Kantenlängen dazuaddiert wird, so dass auch eine Kante, die keinen Verkehr trägt, immer noch die Länge $\alpha$ (statt $0$) hat.

**Rückkopplungseffekt**

**Regelungs- und Steuerungstheorie**

**Biasfaktor**

**Beispiel 8.3-2:**
Setzt man in dem oben (in Beispiel 8.3-1) betrachteten Beispiel einen Biasfaktor von $\alpha = 1$ an, so wird beim 2. Routing Knoten $9$ nicht auf den Uhrzeigersinn umschalten, denn hier ergäbe sich jetzt eine Weglänge von $37$ gegenüber $35 + 7\epsilon$ entgegen dem Uhrzeigersinn. Im weiteren pendelt nur Knoten $8$ mit seinem Verkehr zwischen den Nachbarn $7$ und $9$, im wesentlichen bleibt es also beim Initialrouting.

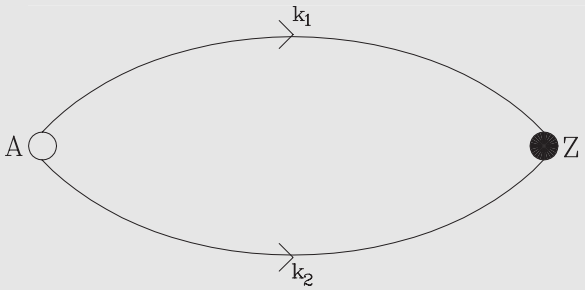
Es liegt auf der Hand, dass die Wahl eines sehr großen $\alpha$ dazu führt, dass der aktuelle Verkehr für das Routing nur wenig oder gar keine Bedeutung hat und nur noch auf Wegen mit möglichst wenigen Kanten geroutet wird. Das Stabilitätsproblem ist dann vermieden, jedoch sind auch die Vorteile der Anpassung an den aktuellen Verkehr damit vergeben. Bei der Beschreibung des Routing im Internet werden wir diesen Punkt noch einmal aufgreifen.

**Verbindungsorientierte Datenübermittlung**

Das soeben geschilderte Stabilitätsproblem bei der verbindungslosen Datenübermittlung (und adaptivem Routing mit kürzesten Wegen) tritt auf, weil nach einer Aktualisierung der kürzesten Wege sofort der gesamte Verkehr auf die neuen (nun kürzeren) Wege umgestellt wird. Da bei der verbindungsorientierten Datenübermittlung eine einmal aufgebaute virtuelle Verbindung für die Dauer der "Sitzung" (engl.

Session) bestehen bleibt und nicht etwa geändert wird, wenn sich im Netz andere kürzere Wege ergeben, leuchtet sofort ein, dass hier das Stabilitätsproblem nicht so gravierend sein wird. Dadurch, dass nach einer Routing-Aktualisierung nur neue virtuelle Verbindungen die neuen Wege nutzen, wird die Gefahr von Oszillationen von vornherein gemildert. Inwieweit das Problem trotzdem auftritt, hängt allerdings vom Verhältnis einiger relevanter Parameter zueinander ab. Wir wollen dies anhand eines Beispiels genauer erläutern.

**Beispiel 8.3-3:**
Gegeben sei das folgende Netz, in dem Knoten A auf zwei Kanten Nachrichten zum Knoten Z schicken kann. (Ausnahmsweise betrachten wir hier also einen Multidigraphen.)




Adaptives Routing mit kürzesten Wegen läuft dann folgendermaßen ab, sowohl im verbindungslosen wie im verbindungsorientierten Fall:
Die Zeitachse wird in Intervalle von jeweils T Sekunden Länge eingeteilt. In jedem Intervall wird auf beiden Kanten die Verkehrsbelastung (in bit/sec) beobachtet, und der gesamte Verkehr (also entweder Datagramme oder neue virtuelle Verbindungen) wird in einem T-Sekunden-Intervall derjenigen Kante zugeordnet, die im vorhergehenden Intervall geringer belastet war. Es wird nun angenommen, dass vom Punkt A aus virtuelle Verbindungen aufgebaut werden, und zwar nach einem Poisson-Prozess mit einer Rate von $\lambda$ pro Sekunde. Die Dauer einer virtuellen Verbindung sei exponentiell verteilt mit einem Mittelwert von $1/\mu$ Sekunden. Aus der Warteschlangentheorie weiß man, dass dann die Anzahl der bestehenden virtuellen Verbindungen Poisson - verteilt ist mit $\lambda/\mu$ als Mittelwert. Ist nun $\gamma$ die durchschnittliche Bitrate, die eine virtuelle Verbindung in Anspruch nimmt, und ist $r$ die insgesamt in Punkt A (für das Ziel Z) erzeugte Bitrate, so muss

$$r = (\frac{\lambda}{\mu})\gamma \quad \text{oder} \qquad\qquad \gamma = \frac{r\mu}{\lambda}$$

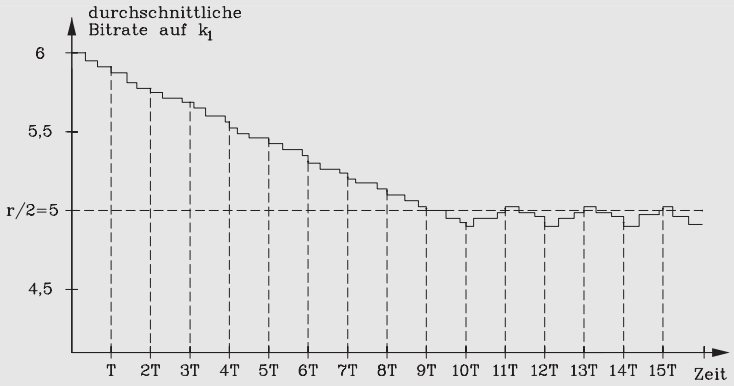
gelten.
Eine weitere Annahme ist jetzt, dass die Zeit T klein ist gegenüber der mittleren Dauer $1/\mu$ einer virtuellen Verbindung. Dann wird ein Anteil von etwa $\mu T$ der virtuellen Verbindungen, die eine Kante zu Beginn eines T-Intervalls beherbergte, am Schluss des Intervalls beendet sein. Im Mittel werden der zuletzt

weniger belasteten Kante $\lambda T$ viele virtuelle Verbindungen in einem Intervall neu zugeordnet; dies ergibt eine zusätzliche Belastung von $\gamma\lambda T$ bit/sec bzw. $r\mu T$ bit/sec.
Mithin werden die mittleren Bitraten $x_1^k$ und $x_2^k$ auf den beiden Kanten im $k-ten$ Intervall folgendem Gesetz gehorchen:

$$x_i^{k+1} = \begin{cases} (1-\mu T)x_i^k + r\mu T, & \text{falls } x_i^k \leq x_j^k \text{ und } \{i,j\} = \{1,2\}, \\ (1-\mu T)x_i^k & \text{sonst.} \end{cases}$$

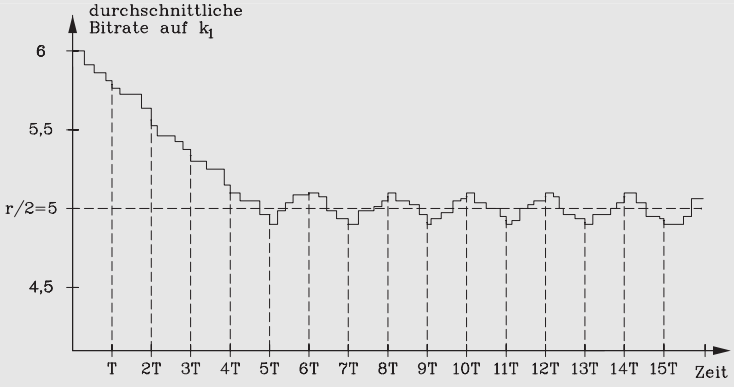
Die folgenden Diagramme, bei denen $r$ = 10 kbit/sec gewählt wurde, zeigen die zeitliche Entwicklung der mittleren Belastung von Kante $k_1$ in den jeweiligen T-Intervallen. Entscheidend ist die Größe von $\mu T$ bzw. das zwischen $\mu$ und $T$ bestehende Verhältnis.

a.



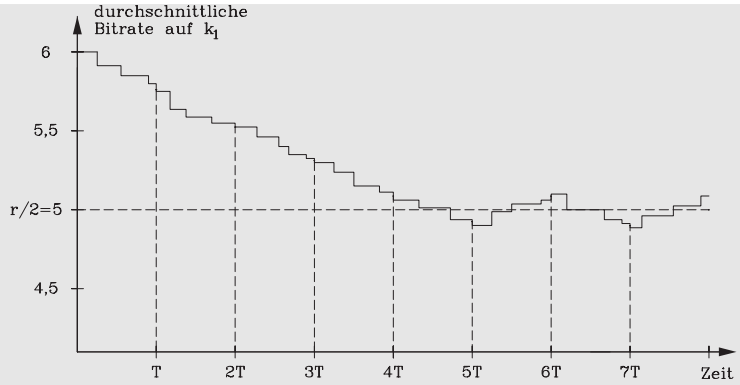

$T = 1$ sec, $1/\mu = 50$ sec
Hier ist $\mu T$ als klein angenommen: virtuelle Verbindungen bleiben lange bestehen, das Routing wird oft aktualisiert.

b.



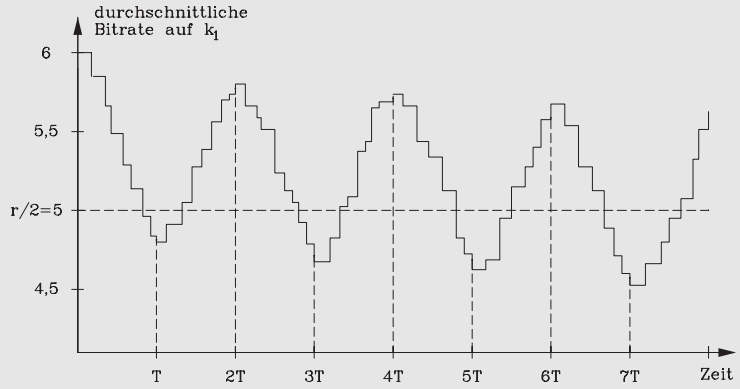

$T = 1$ sec, $1/\mu = 25$ sec
$\mu T$ hat einen mittleren Wert mit großem $\mu$ und kleinem $T$.

c.




$T = 5$ sec, $1/\mu = 125$ sec
$\mu T$ hat einen mittleren Wert mit kleinem $\mu$ und großem $T$.

d.




$T = 5$ sec, $1/\mu = 25$ sec
Hier ist $\mu T$ groß: virtuelle Verbindungen dauern nur kurz, es wird selten aktualisiert.

Man sieht schon an der oben für $x_i^{k+1}$ angegebenen Formel, dass die Raten $x_1$ und $x_2$ für $k \to \infty$ jeweils um $r/2$ oszillieren, wobei die Größe der Schwankung bei $r\mu T$ liegt. Fall a) im 1. Bild bestätigt: Ist $\mu T$ klein (also das Verhältnis von $1/\mu$ zu $T$ groß), so wird der Verkehr fast gleichmäßig auf die beiden Kanten aufgeteilt. Fall d) liefert wie erwartet das Ergebnis, dass bei größer werdendem $\mu$ sich die Oszillationen denen bei der verbindungslosen Übermittlung annähern.

Das sich im obigen Beispiel zeigende Verhalten tritt in ähnlicher Form in jedem Netz mit verbindungsorientierter Datenübermittlung auf. An einem tieferen Einstieg interessierte Leser seien auf die im Literaturverzeichnis zitierte Arbeit von E. M. Gafni und D. P. Bertsekas verwiesen. Allgemein stellt sich heraus: ist die mittlere Dauer $1/\mu$ einer virtuellen Verbindung groß, so bewegt sich das Routing nur langsam auf eine optimale Aufteilung hin; ist jedoch $1/\mu$ klein (d. h. $\mu$ groß), so muss auch $T$ klein sein, damit sich die Oszillationen ebenfalls nur in einem kleinen Bereich bewegen.

**Selbsttestaufgabe 8.3-1:**

*Erläutern Sie für die verbindungslose Datenübermittlung einige der Nachteile, die durch Oszillationen beim Routing entstehen.*

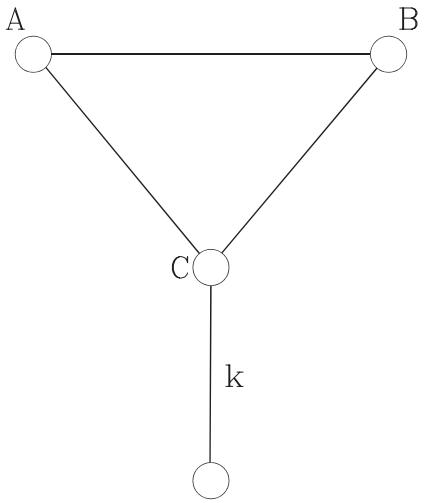
## 8.4 Zur Übertragung von Routing-Informationen

Bei jeder Art von Wegeauswahl in einem Netz ist man stets mit dem folgenden zusätzlichen Grundproblem konfrontiert: es müssen die Informationen über ausgefallene oder übermäßig belastete Subsysteme zu den Stellen gelangen, wo sie für die nächsten anstehenden Routing-Entscheidungen benötigt werden. Für die Übertragung dieser **Routing-Informationen** wird jedoch das gleiche Netz genutzt, so dass es mitunter unmöglich ist, die Informationen wie gewünscht zu verteilen. Solche und ähnliche Schwierigkeiten müssen von den verwendeten Protokollen möglichst gut abgefangen werden.

**Routing-Informationen**

Arbeitet ein Netz mit zentralem Routing, so werden alle Netzänderungen an eine Zentrale Z gemeldet, wo die Entscheidungen über die zu wählenden Routen gefällt werden. Z muss jedem Knoten die ihn betreffenden Informationen zukommen lassen. Eine Schwierigkeit besteht nun darin, dass Z ausfallen oder unter Umständen gewisse Teile des Netzes nicht mehr erreichen kann. Wenn das Problem des Ausfalls von Z auch gemildert werden kann (z. B. durch Redundanz, etwa Dopplung relevanter Komponenten), so dürfte doch deutlich sein, dass die zweitgenannte Schwierigkeit gegebenenfalls zu einer hohen Leistungseinbuße im Netz führen kann.

Ein anderes Problem liegt bei einem Netz mit relativ häufigen Netzänderungen (und folglich häufigem Austausch von Routing-Informationen) in der Unterscheidung zwischen aktueller und bereits veralteter Routing-Information. Um das Problem zu verdeutlichen, stellen wir uns ein Netz vor, in dem jeder Knoten alle von ihm wegführenden Kanten beobachtet und im Falle einer Änderung die Information durch "Flooding" (s. Abschnitt 8.1) im ganzen Netz verbreitet. Folgendes Beispiel zeigt die Problematik auf, die sich im Falle mehrerer Änderungen ergeben kann:




Angenommen, die Kante k ist zunächst intakt, fällt dann für eine kurze Weile aus und ist anschließend wieder funktionsfähig. Weiter sei angenommen, dass die beiden von C erzeugten Nachrichten über den Ausfall bzw. die Wiederherstellung von k den Weg CBA schneller zurücklegen als die erste Nachricht den Weg CA, und dass die Kante CA ausfällt, nachdem A diese Nachricht von C (über den Ausfall von k) erhalten hat. In diesem Falle kann A über einen längeren Zeitraum der falschen Ansicht sein, dass k nicht funktionsfähig ist. Wir werden uns in Kürze Mechanismen ansehen, die solche Situationen weitgehend verhindern.

In dem ersten betrachteten Szenario wurde davon ausgegangen, dass eine Routing-Information jeweils von einem Knoten an alle anderen Knoten im Netz (und nicht nur an einen bestimmten) weitergegeben werden soll. Man spricht hier von **Broadcasting**- oder **Rundspruch**-Verfahren. Es bietet sich natürlich an, dafür eine Form des Flooding zu nutzen.

**Broadcasting**
**Rundspruch**

Wenn nicht explizit anders vermerkt, werden wir im folgenden stets davon ausgehen, dass die Routing-Informationen durch Broadcasting im Netz bekanntgegeben werden sollen.

**Nachricht**

Ferner wird folgende Sprachregelung eingeführt: wir sprechen von einer **Nachricht**, wenn damit eine allgemeine für das Routing relevante Information (z. B. über den Ausfall einer bestimmten Kante) gemeint ist. Physikalisch schlägt sich eine Nachricht in (u. U. mehreren) **Meldungen** nieder – z. B. erzeugt beim Flooding eine Nachricht zahlreiche Meldungen.

**Meldung**

An dieser Stelle muss erwähnt werden, dass in der Praxis beim Broadcasting-Verfahren oft auch unter Verwendung eines minimalen Gerüsts "geroutet" wird, und zwar insbesondere dann, wenn ein zentraler Knoten regelmäßig per Rundspruch Nachrichten im Netz zu verteilen hat; es werden dann dazu nur die Kanten eines minimalen Gerüsts mit Z als Wurzel genutzt. Auch bei anderen Anwendungen sind Gerüste von Interesse, z. B. wenn Knoten eine Nachricht nur zu einigen anderen senden wollen - man spricht hier im Englischen von "multicast communications". Es gibt hierzu zahlreiche theoretische Untersuchungen. Wir wollen hierauf jedoch

nicht näher eingehen und stellvertretend auf die Arbeit von L. Berry (siehe Literaturverzeichnis) verweisen.

Das Verfahren zur Übertragung von Routing - Informationen und der Routing - Algorithmus (zur Wegeauswahl) müssen so aufeinander abgestimmt sein, dass der Routing - Algorithmus neue Informationen, die während seiner Ausführung eintreffen, in sinnvoller Weise integriert und irgendwann zum Tragen kommen lässt. Bei dem in Abschnitt 8.2 behandelten verteilten asynchronen Bellman-Ford Algorithmus ist dies durch die periodische Anwendung des Algorithmus in den einzelnen Knoten und unter Verwendung der aktuellsten vorliegenden Informationen gewährleistet.

Ein weiteres Problem besteht schließlich darin, dass auch bei der Reparatur einer zuvor ausgefallenen Kante die Abläufe garantieren sollten, dass möglichst schnell die neue Situation netzweit bekannt gemacht wird, damit das Netz optimal genutzt werden kann. Es liegt auf der Hand, dass dieses Problem z. B. dann besonderes Gewicht hat, wenn durch die Reparatur der Kante zwei zuvor getrennte Teile des Netzes wieder zusammenhängen - jeder Teil wird über den Zustand des anderen völlig veraltete Informationen haben.

Die soeben geschilderten Probleme der Übertragung von Routing-Informationen beziehen sich sowohl auf den Fall, dass diese Informationen die reine Netztopologie betreffen (d. h. welche Knoten und Kanten derzeit intakt sind), als auch darauf, dass mit diesen Informationen Verkehrsbelastungen von Kanten oder ähnliche Größen ausgetauscht werden. Im zweiten Fall stellen sich die obigen Probleme nicht ganz so gravierend, da sie u. U. nur zu einer nicht optimalen Ausnutzung der Netzressourcen führen. Da jedoch in jedem Fall Reaktions- und Übertragungszeiten mitspielen, muss eines grundsätzlich festgehalten werden: Es ist unmöglich, dass jeder Knoten des Netzes zu jedem beliebigen Zeitpunkt über alle korrekten für ihn relevanten Informationen verfügt. Man kann bestenfalls erwarten, dass wenn endlich viele Netzänderungen eintreten und sich ab einem Zeitpunkt $t_0$ nichts mehr ändert, es stets einen Zeitpunkt $t_1 > t_0$ gibt, zu dem alle Knoten einer Zusammenhangskomponente des Netzes den richtigen Zustand jeder Kante in ihrer Komponente kennen. (Man vergleiche dazu nochmals den Beweis der Korrektheit des verteilten asynchronen Bellman-Ford Algorithmus, bei dem eine anologe Forderung aufgestellt wurde.)

Um überhaupt Aussagen über einige Verfahren zur Übertragung von Routing-Informationen machen zu können, wird meist von einigen Annahmen über das Netz und dessen Protokolle ausgegangen. In der Praxis sind bei einem Netz, das die Gültigkeit dieses Annahmen "im Prinzip" garantiert, immer Fehlersituationen (theoretisch) konstruierbar, bei denen eine Annahme doch nicht greift. Im konkreten Beispiel muss natürlich untersucht werden, wie häufig mit dieser Art von Fehlern gerechnet werden muss und wie gravierend die Konsequenzen sind.

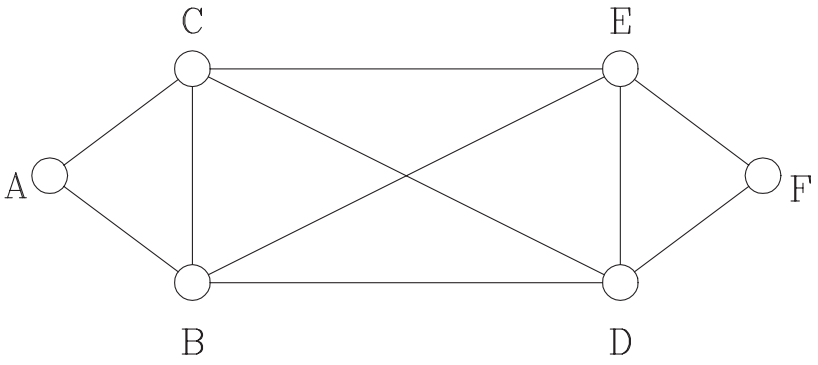
**Annahme 1:** Auf den Netzkanten werden die Nachrichten korrekt und in der richtigen Reihenfolge übertragen. In den Knoten gespeicherte Nachrichten werden nicht verfälscht.

**Annahme 2:** Der Ausfall einer Kante wird (nicht notwendig gleichzeitig) von beiden Endknoten bemerkt. Erst danach kann die Kante von einem der Knoten als wieder repariert erkannt werden.

**Annahme 3:** Mit Hilfe eines Schicht-2-Protokolls können beide Seiten eine Kante als funktionsfähig erkennen. Falls eine Seite die Kante für intakt erklärt, so tut dies nach endlicher Zeit die andere Seite ebenfalls, es sei denn, die erste Seite erklärt sie wieder für defekt.

**Annahme 4:** Fällt ein Knoten aus, so wird nach endlicher Zeit jede an ihn angeschlossene Kante von der anderen Seite für defekt erklärt.

Als nächstes wollen wir uns das bereits in Abschnitt 8.1 eingeführte "Flooding" mit verschiedenen Varianten bzw. Verfeinerungen ansehen. Schon an früherer Stelle wurde als Beispiel das folgende Netz betrachtet:




Es wurde dort davon ausgegangen, dass Knoten A eine Mitteilung an F schicken möchte. Flooding mit Zählern und Schwellwert 3 führte dazu, dass insgesamt 26 Nachrichten entstanden. Durch die Einführung des Zählers hat man zwar ein unendliches Kreisen der Nachricht im Netz verhindert, jedoch sendet z. B. immer noch Knoten D zwei zu der gleichen Nachricht gehörende Meldungen an seinen Nachbarn E (ausgelöst durch Empfang einer Meldung von B bzw. C). Dies kann verhindert werden, indem statt mit den Zählern mit **Folgenummern** gearbeitet wird.

**Folgenummern**

Die Folgenummern werden wie folgt gehandhabt. (Es ist dabei unerheblich, ob nur eine Mitteilung zwischen zwei Partnern ausgetauscht oder ein Rundspruch gemacht werden soll.) Jeder Knoten versieht die Routing-Nachrichten, die er initiiert, mit fortlaufenden Folgenummern. Empfängt ein Knoten Y eine Meldung, die zu einer in X erzeugten Nachricht gehört, so prüft Y nach, ob die in die Meldung eingetragene Folgenummer größer ist als die Folgenummer der Nachricht, die Y zuletzt von X bekam. Ist dies der Fall, so speichert Y die Nachricht mit ihrer Folgenummer und sendet außerdem entsprechende Meldungen an alle Nachbarn mit Ausnahme desjenigen, von dem Y diese Nachricht bekommen hat (und mit Ausnahme von X). Im anderen Fall wird die Nachricht von Y verworfen.

Wir betrachten wieder das obige Beispiel, bei dem A eine Nachricht nach F schicken möchte. Es wird angenommen, dass jeder Knoten von A zuletzt eine Nachricht mit Folgenummer 4 erhalten (und diese abgespeichert) hat. A sendet nun für die neue hier betrachtete Nachricht Meldungen mit Folgenummer 5 an B und C. Man sieht, dass die nun folgenden Aktionen von der zeitlichen Reihenfolge gewisser Ereignisse (Ankünfte von Meldungen) abhängen. Beispielhaft werden für eine mögliche Reihenfolge die Ereignisse dargestellt. Sowohl B als auch C mögen nahezu gleichzeitig die Meldung von A erhalten, was dazu führt, dass B Meldungen an C, D, E und C solche an B, D, E sendet. Danach haben B und C die Nachrichten mit Folgenummer 5 von Knoten A abgespeichert und ignorieren folglich die nun vom jeweils anderen erhaltenene. Es sei nun angenommen, dass D zuerst die Meldung von B (und danach die von C) erhält, E erhalte zunächst die von C und dann die von B. Resultat davon ist, dass D Meldungen an C, E, F und E solche an B, D, F sendet, womit die "Flut" stoppt. Insgesamt wurden 14 Meldungen versendet.

Man sieht, dass sich nur ein geringfügiger Unterschied ergibt, wenn A eine Routing-Nachricht per Broadcasting im Netz verteilen will: in diesem Fall sendet noch zusätzlich F eine Meldung an denjenigen seiner beiden Nachbarn, von dem er die Nachricht nicht zuerst bekommen hat.

Der Gebrauch von Folgenummern stellt sicher, dass jeder Knoten nur einmal eine zu einer Nachricht gehörenden Meldungsflut an seine Nachbarn schickt. Auch das Problem der Unterscheidung aktueller und veralteter Routing-Informationen stellt sich so nicht mehr - der zu Anfang dieses Abschnitts geschilderte Fall kann nun nicht mehr auftreten. Allerdings gibt es auch eine Reihe von Schwächen, die nun kurz geschildert werden.

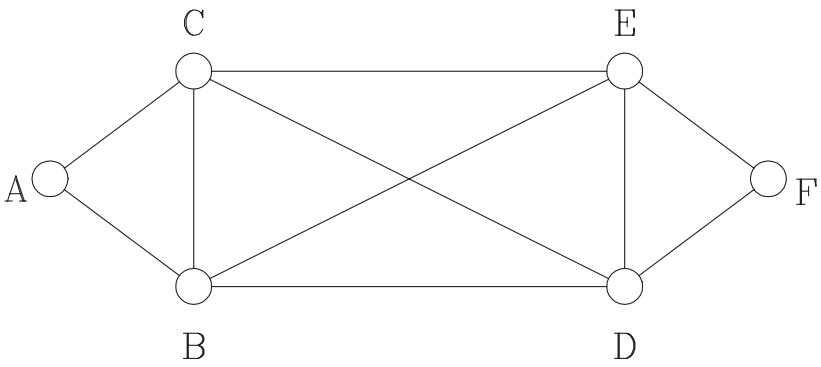
Das erste Problem liegt schon darin, dass das in den Meldungen enthaltene Datenfeld für die Folgenummmer nur eine begrenzte Länge hat und irgendwann der maximal mögliche Eintrag erreicht ist, so dass bei der Addition von 1 der Wert wieder auf 0 umschlägt. (Dies bedeutet: stehen k Bits für die Folgenummer zur Verfügung, so wird modulo $2^k$ gerechnet.) Will man diese Situation nicht extra im Protokoll berücksichtigen, so muss man die Festlegung treffen, dass die Folgenummer 0 größer ist als die Nummer $2^k - 1$.

**Beispiel 8.4-1:**
Es sei ein beliebiges Netz gegeben. Wir nehmen an, dass die Routing-Nachrichten von den einzelnen Knoten per Broadcasting gestreut werden und dazu Flooding mit Folgenummern verwendet wird, für die in den Nachrichten 3 Bits vorgesehen sind. A sei ein beliebiger Knoten, der zuletzt die Folgenummer 7 (also binär in Bits 111) verwendet hat. Initiiert A nun eine weitere Nachricht (bzw. Flut von Meldungen), so tragen diese Meldungen die Folgenummer 000; ein Nachbar von A muss diese Nachricht natürlich akzeptieren und weiterreichen, mithin 0 als größer als 7 ansehen.

In der Praxis wird die Anzahl k der für die Folgenummer reservierten Bits oft so groß gewählt, dass das obige Problem ohne Bedeutung ist. Ernster ist jedoch der Fall, wenn eine teilweise Reinitialisierung des Netzes stattfinden muss. Z.B. könnte die Situation eintreten, dass einige Knoten nach einem Ausfall wieder in Betrieb gehen. Sinnvollerweise sollten diese Knoten bei den von ihnen erzeugten Nachrichten wieder mit der Folgenummer 0 beginnen, jedoch könnte dies in anderen Netzteilen zu unerwünschten Konsequenzen führen. Ein weiteres Problem kann dadurch entstehen, dass (z. B. aufgrund eines Hardware- oder eines Übertragungsfehlers) eine "falsche" Folgenummer übertragen wird; eine mögliche Folge (etwa wenn diese Nummer zu groß war) ist, dass die nächsten von demselben Knoten erzeugten Nachrichten im Netz ignoriert werden. Im nächsten Abschnitt werden anhand des ARPANET Möglichkeiten aufgezeigt, wie diese Schwierigkeiten weitgehend in den Griff zu bekommen sind.

**Selbsttestaufgabe 8.4-1:**

*Es sei das folgende Netz gegeben:*




*Betrachtet wird die folgende Situation: A möchte eine Nachricht per Broadcasting verschicken. Es wird Flooding mit Folgenummern angewendet, wobei vorausgesetzt sei, dass jeder Knoten von A zuletzt die Folgenummer 4 erhalten hat.*

*Erläutern Sie, was geschieht, wenn die von A nach B gesendete Nachricht irrtümlich mit der Folgenummer 2 versehen ist.*

## 8.5 Das Routing im Internet

Das ARPANET als Vorläufer des Internet war eines der ersten großen Netze zur Datenkommunikation, die geographisch verteilte Rechner mit Nutzern und Peripheriegeräten verbinden. Es ging um 1970 in den USA in Betrieb. Das ARPANET wurde ursprünglich konzipiert von der US Defense Advanced Research Project Agency (DARPA, später nur ARPA), zunächst als reines Forschungsnetz. Zu Anfang noch basierend auf konventionellen Festverbindungen zwischen Endpunkten, hat das Netz eine Vorreiterrolle für Paketnetze gespielt. Später (um 1980) wurde das ARPANET zum Backbone des neuen Internets und für die frühen Experimente mit TCP/IP benutzt. Das ARPANET selbst wurde später in zwei eigenständige Netze aufgeteilt: Der Forschungsteil behielt den Namen ARPANET, der Militärabschnitt wurde unter dem Namen MILNET bekannt. Der militärische Ursprung des ARPANET ist kein Zufall: gerade das Bedürfnis nach besonders zuverlässigen Netzen hat zur Entwicklung von Paketnetzen (mit ihren sehr flexiblen Möglichkeiten des Routing) geführt. Auf diese Aspekte kommen wir an späterer Stelle zurück. Mittlerweile ist das Internet weltweit zum dominierenden Kommunikationsnetz geworden, über das nicht nur geschäftliche Kommunikation abgewickelt wird – auch für die privaten Nutzer sind E-Mail-Kommunikation und das "Surfen" im World Wide Web zur Selbstverständlichkeit geworden. Die Dominanz der TCP/IP-Protokolle wirkt heute auch in die Unternehmensnetze hinein (unter dem Begriff *Intranet*), und selbst das konventionelle Telefonnetz wird zunehmend mit der Technik *Voice over IP* durch das Internet ersetzt.

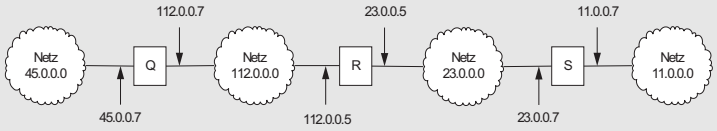
Wie funktioniert das Routing im Internet?

Das für das Routing auf Schicht 3 zuständige *IP*-Protokoll arbeitet grundsätzlich im Datagramm-Modus. Dies bedeutet, dass ein Rechner im Internet ein Datenpaket, welches nicht für ihn selbst bestimmt ist, entsprechend der in dem Paket eingetragenen Zieladresse an denjenigen von ihm direkt erreichbaren Rechner weitersendet, der ihm für dieses Ziel am günstigsten erscheint. Im allgemeinen wird das ein Rechner sein, der auf dem (bzw. allgemeiner: einem) kürzesten Weg zum Ziel liegt. Da jeder Zwischenrechner auf dem Weg vom ursprünglichen Absender zum endgültigen Empfänger seine Routing-Entscheidung (also: An wen schicke ich das Paket weiter?) aufgrund der ihm vorliegenden Informationen (auch über den aktuellen Netzzustand) selbständig trifft, "weiß" der Absender nicht, welchen konkreten Weg sein Paket zu dem von ihm gewünschten Ziel nehmen wird. Auch ist es durchaus möglich, dass zwei von demselben Absender kurz hintereinander verschickte Pakete zum selben Ziel trotzdem auf unterschiedlichen Wegen laufen.[3] In den ersten zehn Jahren des Betriebes wurde im ARPANET der verteilte asynchrone Bellman-Ford Algorithmus für die Ermittlung kürzester Wege verwendet. Die Länge einer Kante steht dabei zu ihrer aktuellen Belastung in Beziehung, genauer: ein Knoten A bemisst die Länge der Kante zum Nachbarn B mit Hilfe der Länge der

3 Auf das optional mögliche *Source Routing*, bei dem der Versender eines Datenpakets die genaue Route im vorhinein festlegen kann, gehen wir nicht ein. Es wird z. B. für den Test von Routen verwendet.

Warteschlange von Nachrichten, die im Punkt A auf die Übertragung längs dieser Kante warten. (Die Kanten haben folglich verschiedene Längen für die beiden Richtungen.) Benachbarte Knoten tauschen bei dieser Realisierung alle 625 msec ihre geschätzten Abstände zu allen Zielen aus. Das Hauptproblem, das dann zur Änderung des Routing-Verfahrens im ARPANET führte, lag hier bei der mangelnden Stabilität (vgl. Abschnitt 8.3): dadurch, dass die Warteschlangenlängen sich laufend ändern, wurde ein optimaler "Endzustand" durch den Algorithmus so gut wie nie erreicht, das Resultat waren ständige Oszillationen. Die Addition eines Biasfaktors zu den Kantenlängen verhinderte dies, hatte aber den Effekt, dass das Netz nun nicht mehr sensibel genug auf Verkehrsstaus reagieren konnte. Im Jahre 1980 wurde ein neues Routing-Verfahren ins ARPANET eingeführt. Zur Bestimmung der kürzesten Wege gibt es keine "Kooperation der Knoten" mehr, vielmehr berechnet jeder Knoten für sich die kürzesten Wege zu allen anderen (mit dem Algorithmus von Dijkstra). Jeder Knoten beobachtet die Längen der von ihm ausgehenden Kanten und teilt diese per Rundspruch mit dem Flooding-Verfahren allen anderen Knoten mit, und zwar mindestens einmal alle 60 Sekunden. Doch dies ist alles "Schnee von gestern". Wie im folgenden erläutert wird, erlaubt die heutige komplexe Struktur des Internet die Anwendung mehrerer Routing Verfahren auf unterschiedlichen Ebenen. Für die folgenden Abschnitte wird beim Leser die Kenntnis der einfachsten Grundlagen der *TCP/IP*-Protokolle vorausgesetzt, insbesondere sollte die Form der *IP*-Adressen bekannt sein.

**Beispiel 8.5-1:**
Das Grundprinzip des aktuell verwendeten Routing im Internet wird am besten vorab durch ein Beispiel illustriert. Im folgenden Bild sind vier Kommunikationsnetze dargestellt, die durch *Router* miteinander verbunden sind und so einen typischen kleinen Ausschnitt des Internet darstellen. Unter den vier Netzen kann man sich z. B. jeweils ein auf der *Ethernet-Technik* basierendes Unternehmensnetz vorstellen.




Zwischen den vier Netzen sind Router *Q*, *R* und *S* eingezeichnet. Ferner sind *IP*-Adressen eingetragen, deren vier Bytes (entsprechend Version *IPv4* von *IP*) wie üblich dezimal angegeben sind. Beispielsweise ist dem links eingezeichneten Netz die Netzadresse 45.0.0.0 zugeordnet, was bedeutet, dass die zu diesem Netz gehörenden Hosts Adressen der Form 45.0.0.1, 45.0.0.2 etc. besitzen. Die Router haben mehrere *IP*-Adressen, da sie in beiden Netzen beheimatet sind, die von ihnen verbunden werden - beispielsweise hat *Q* als *IP*-Adressen sowohl 45.0.0.7 als auch 112.0.0.7.

Im folgenden Bild ist die *Routing-Tabelle* des Routers *R* gezeigt:

| Ziel in Netz | Routing-Entscheidung |
|---|---|
| 23.0.0.0 | direkte Zustellung |
| 112.0.0.0 | direkte Zustellung |
| 45.0.0.0 | an 112.0.0.7 |
| 11.0.0.0 | an 23.0.0.7 |

Da *R* zu den beiden Netzen 112.0.0.0 und 23.0.0.0 gehört, kann dieser Router Pakete für Hosts in diesen beiden Netzen direkt zustellen (entsprechend den dort vorhandenen Protokollen und Adressen, z. B. nach Ethernet-Standard). Empfängt *R* jedoch ein Datagramm mit Zieladresse 45.0.0.3 bzw. will selbst ein solches verschicken (was hinsichtlich des Routing keinen Unterschied macht), so sagt ihm seine Tabelle, dass er dieses Paket an die Adresse 112.0.0.7 weiter schickt, denn dies ist der zuständige nächste Router, den er direkt erreichen kann (weil dieser wie er selbst zum Netz 112.0.0.0 gehört).

Zunächst eine Begriffsklärung: In dem Beispiel 8.5-1wird zwischen *Hostrechnern* und *Routern* unterschieden. Ein Hostcomputer ist in der Regel direkt an ein einzelnes physisches Netz angeschlossen, wie z. B. ein Rechner in einem Ethernet-LAN. Es gibt jedoch auch sogenannte *Multi-Homed-Hosts*, die an mehrere Netze angeschlossen sind. Ein Router ist immer an zwei oder mehr Netze angeschlossen, um seinen Zweck erfüllen zu können: Pakete von einem Netz in ein anderes zu übertragen. Da auch ein Host Routing-Entscheidungen treffen muss (nämlich an welchen der im gleichen Netz angeschlossenen Router ein Datenpaket zu schicken ist), ist die Trennung zwischen Hosts und Routern nicht ganz scharf. Auf den Punkt gebracht kann man jedenfalls sagen: Ein Host überträgt keine Pakete von einem Netz in ein anderes.

Das Beispiel 8.5-1 wirft eine Reihe von Fragen auf, die direkt mitten in das Thema des Routing im Internet führen:

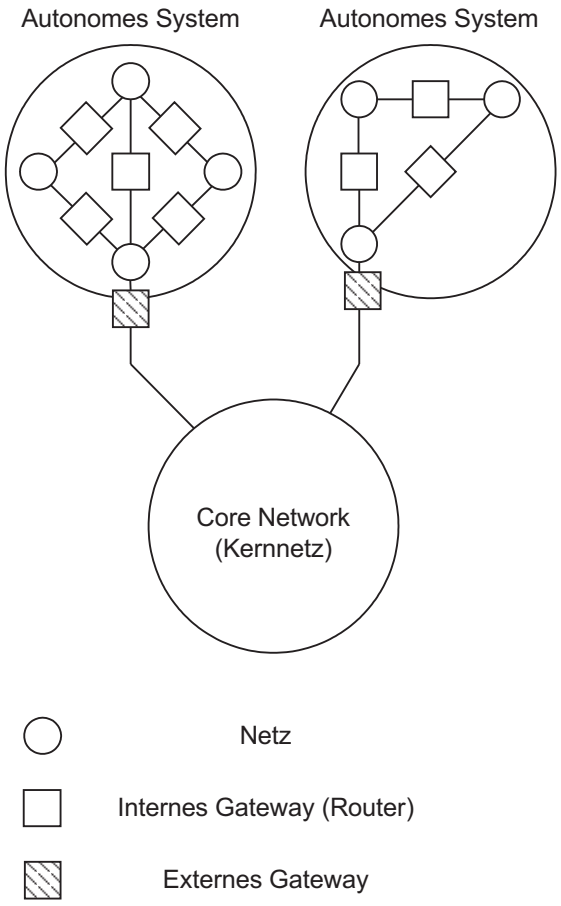
1. Muss jeder Host und jeder Router alle an das Internet angeschlossenen Netze kennen, um über vollständige Routing-Tabellen zu verfügen?

2. Wie werden die Routing-Tabellen erstellt? Wie werden sie aktualisiert (z. B. bei Ausfällen von Routern oder Verbindungen)?

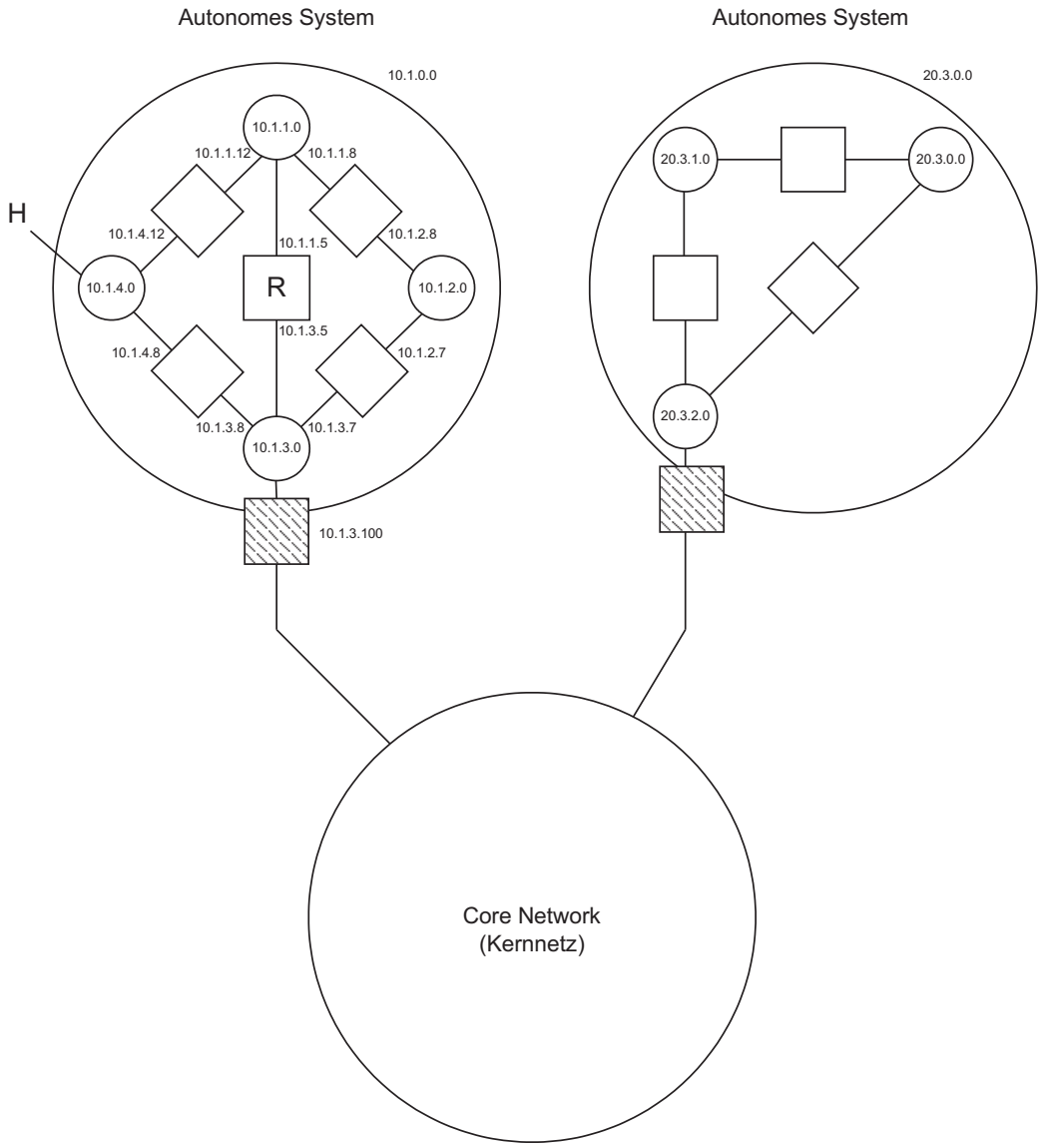
Es leuchtet ein, dass es unmöglich wäre, Routing-Informationen für viele Millionen Zieladressen in jeder Routing-Tabelle eines beliebigen Rechners zu speichern und diese laufend zu aktualisieren. Hier hilft die *hierarchische Struktur* des Internet sowie die Möglichkeit, Hierarchien durch die Bildung von *Subnetzen* in den *IP*-Adressen abzubilden – darauf gehen wir in Kürze ein. Zunächst müssen wir uns den Aufbau des heutigen Internet noch einmal genauer ansehen.

Das Internet besteht aus einer Reihe von *Autonomen Systemen* (Teilnetzen), die sich im Laufe der Zeit entwickelt haben, und dem sogenannten *Core Network* (Kernnetz), das für die Verbindung der Autonomen Systeme untereinander sorgt.

Ein Autonomes System, das in der Regel aus mehreren Netzen besteht, unterliegt einer lokalen Administration. Beispiele für Autonome Systeme sind das Netz von T-Online oder das Deutsche Forschungsnetz, an dem alle Universitäten angeschlossen sind. Die Netze eines Autonomen Systems sind untereinander mit Hilfe von *internen Gateways* verbunden, die auch *Router* heißen. Ferner verfügt ein Autonomes System über ein *externes Gateway*, das die Verbindung zum Core Network herstellt. In der folgenden Grafik ist ein fiktiver Ausschnitt des Internet mit zwei Autonomen Systemen dargestellt:




Mit Hilfe der *Subnetz-Adressierung*, durch die *IP*-Adressen flexibler gehandhabt werden können, ist es nun möglich, die hierarchische Struktur des Internet in den *IP*-Adressen abzubilden. Das Prinzip ist im nächsten Bild am demselben fiktiven Beispiel wie oben dargestellt.




Die Routing-Tabelle des Routers *R* könnte nun folgendermaßen aussehen:

| **Ziel in Netz** | **Routing-Entscheidung** |
|---|---|
| 10.1.1.0 | direkte Zustellung |
| 10.1.3.0 | direkte Zustellung |
| 10.1.2.0 | an 10.1.3.7 |
| 10.1.4.0 | an 10.1.3.8 |
| nicht in 10.1.0.0 | 10.1.3.100 |

Der Host *H* könnte mit folgender Routing-Tabelle arbeiten:

| **Ziel in Netz** | **Routing-Entscheidung** |
|---|---|
| 10.1.4.0 | direkte Zustellung |
| 10.1.1.0 | an 10.1.4.12 |
| 10.1.2.0 | an 10.1.4.12 |
| sonstiges | an 10.1.4.8 |

Man sieht an diesem Beispiel: Durch die Ausnutzung der Struktur können die Routing-Tabellen recht schlank gehalten werden. Der hier betrachtete Router muss die jenseits des externen Gateways liegenden Rechner des Internet überhaupt nicht "kennen" – er muss nicht einmal wissen, aus welchen einzelnen Rechnern das zu seinem eigenen Autonomen System gehörende Netz 10.1.2.0 besteht! Host *H* muss noch weniger wissen! Wir fassen noch einmal zusammen:

- Endgeräte eines Netzes benötigen nur die Routing-Informationen, um Datagramme an andere Endssyteme oder interne Gateways des eigenen Netzes schicken zu können.
- Interior Gateways können Datenpakete nur an Endgeräte oder Gateways schicken, die innerhalb des gleichen Autonomen Systems liegen.
- Exterior Gateways besitzen Routing-Informationen, die es ihnen gestattet, Nachrichten an Interior Gateways ihres Autonomen Systems oder an andere Exterior Gateways zu versenden.

Diese Punkte haben die wichtige Konsequenz, dass innerhalb der verschiedenen Autonomen Systeme durchaus unterschiedliche Routing-Protokolle zum Aufbau und zur Pflege der Routing-Tabellen verwendet werden können - solange der Kontakt zur "Außenwelt" über das externe Gateway ordentlich funktioniert, ist der Innenverkehr eines Autonomen Systems für diese Außenwelt nicht von Belang. Man unterteilt daher die Routing-Protokolle in die folgenden Kategorien

- das *Address Resolution Protocol* (*ARP*) für Endsysteme;
- die *Interior Gateway Protocols* (*IGPs*) wie z. B.
  - das *Routing Information Protocol*(*RIP*),
  - das *Open Shortest Path First Protocol* (*OSPF*)

  für die Verwendung innerhalb Autonomer Systeme;
- die *Exterior Gateway Protocols* (*EGPs*) wie z. B.
  - das *spezielle Exterior Gateway Protocol* (*EGP*),
  - das *Border Gateway Protocol* (*BGP*)

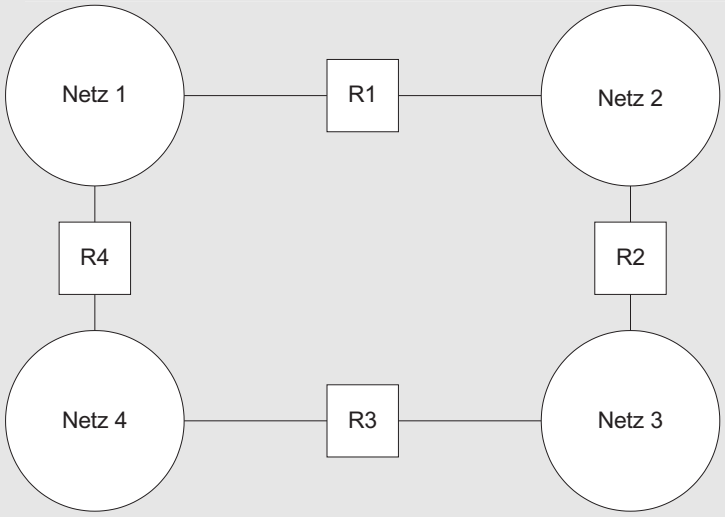
  für Exterior Gateways.

Aus Sicht der Graphentheorie sind besonders die *IGPs* von Interesse, weshalb die beiden obengenannten Vertreter *RIP* und *OSPF* im folgenden skizziert werden. Das *ARP*-Protokoll wird benötigt, damit ein Rechner ein Internet-Paket (mit *IP*-Zieladresse) an den richtigen Computer innerhalb seines eigenen Netzs senden kann - durch *ARP* wird die zu der *IP*-Adresse gehörende physische Hardwareadresse ermittelt. Bei *EGP* handelt es sich eigentlich nicht um ein Routing-Protokoll, vielmehr müsste man von einem Erreichbarkeits-Protokoll sprechen. Es wird verwendet, um dem Core Network und den Exterior Gateways anderer Autonomer Systeme Informationen über die Erreichbarkeit seiner eigenen Netze mitzuteilen. Mit der Weiterentwicklung zum *BGP* werden neben der Erreichbarkeit weitere Informationen ausgetauscht (wie z. B. der nächste Router, den man auf dem Weg zu einem

bestimmten Ziel nutzen sollte), auf deren Basis bessere Routing-Entscheidungen möglich sind. Auf Details gehen wir hier nicht ein.

*Das Routing Information Protocol (RIP)* Das *RIP* ist das am häufigsten innerhalb Autonomer Systeme verwendete Routing-Protokoll. Es handelt sich um ein verteiltes Protokoll, das einen *Distance Vector Algorithmus* (*DVA*) verwendet. Typisch für einen solchen Algorithmus ist, dass eine Metrik zur Bestimmung des Abstands zweier Router verwendet wird; die Metrik kann dabei auf dem graphentheoretischen Abstand beruhen (Anzahl der Zwischen- knoten auf einem kürzesten Weg) oder auf der Übertragungszeit zwischen den Routern. Solche *DVAs* benutzen jedoch in jedem Fall einen verteilten Algorithmus: Mit Hilfe von Broadcast-Nachrichten werden Routing-Informationen ausgetauscht, aufgrund derer jedem Router des Autonomen Systems der Aufbau seiner Routing-Tabelle ermöglicht wird, in der für jedes Ziel der günstigste direkt erreichbare Nachbarrouter eingetragen ist. Beim *RIP*-Protokoll wird als Metrik die Anzahl der Router verwendet, die bei der Übertragung von Paketen auf dem Weg zum Ziel durchlaufen werden müssen. (Man spricht auch von *Hop Counts*.) Es ist hierfür ein maximaler Wert von 15 Hop Counts vorgesehen, d. h. ein Netz mit einem größeren Wert gilt als unerreichbar. Von seiten der Netzadministration besteht die Möglichkeit, den Hop Count für einen Weg künstlich zu erhöhen und so diesen Weg anders zu gewichten - dies mag z. B. für eine Satellitenstrecke sinnvoll sein, auf der längere Übertragungszeiten entstehen. Resultat des *RIP*-Protokolls ist jedoch stets nur ein Weg zu einem bestimmten Ziel.

**Beispiel 8.5-2:**
Den Ablauf des *RIP*-Protokolls zum Aufbau und der ständigen Aktualisierung der Routing-Tabellen illustrieren wir zunächst anhand eines Beispiels für ein Autonomes System, das im nächsten Bild dargestellt ist.



Bei der Initialisierung der Routing-Tabellen trägt jeder Router für die bei ihm direkt angeschlossenen Netze die Distanz 0 ein:

$R_1$

| Netz | Distanz |
|---|---|
| 1 | 0 |
| 2 | 0 |

$R_2$

| Netz | Distanz |
|---|---|
| 2 | 0 |
| 3 | 0 |

$R_3$

| Netz | Distanz |
|---|---|
| 3 | 0 |
| 4 | 0 |

$R_4$

| Netz | Distanz |
|---|---|
| 4 | 0 |
| 1 | 0 |

Nach dieser Initialisierung beginnt der Austausch der Routing-Informationen, mit dem die Tabellen vervollständigt bzw. danach ständig aktualisiert werden. Das *RIP*-Protokoll verwendet dazu einen Broadcast-Mechanismus, mit dem ein Router den anderen Routern in regelmäßigen Abständen (alle 30 Sekunden) seine neuen Tabellen mitteilt. Jeder Router vergleicht die in den Broadcast-Nachrichten propagierten Routen mit den bisherigen eigenen und korrigiert ggf. seine Tabellen. Für unser Beispiel sind bereits nach dem ersten Broadcast die Tabellen vollständig wie folgt aufgebaut. (Dabei ist zusätzlich notiert, über welchen direkt erreichbaren Router die angegebene Distanz zu realisieren ist.)

$R_1$

| Netz | Distanz / über Router |
|---|---|
| 1 | 0 / $R_1$ |
| 2 | 0 / $R_1$ |
| 3 | 1 / $R_2$ |
| 4 | 1 / $R_4$ |

$R_2$

| Netz | Distanz / über Router |
|---|---|
| 1 | 1 / $R_1$ |
| 2 | 0 / $R_2$ |
| 3 | 0 / $R_2$ |
| 4 | 1 / $R_3$ |

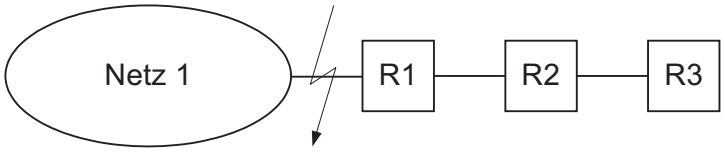
$R_3$

| Netz | Distanz / über Router |
|---|---|
| 1 | 1 / $R_4$ |
| 2 | 1 / $R_2$ |
| 3 | 0 / $R_3$ |
| 4 | 0 / $R_3$ |

$R_4$

| Netz | Distanz / über Router |
|---|---|
| 1 | 0 / $R_4$ |
| 2 | 1 / $R_1$ |
| 3 | 1 / $R_3$ |
| 4 | 0 / $R_4$ |

Wie man an dem Beispiel sieht, ist $RIP$ ein recht einfaches Protokoll. Allerdings hat es einige Schwächen, aufgrund derer zusätzliche Mechanismen eingeführt wurden. Eine Schwäche ist die der *langsamen Konvergenz*: Nach einer Veränderung im Netz - z. B. dem Ausfall einer Leitung - kann es übermäßig lange dauern, bis sich stabile neue Routing-Tabellen herausgebildet haben (siehe auch in Abschnitt 8.4, in dem derartige Probleme diskutiert werden). Das Problem kann am folgenden Bild verdeutlicht werden:



Es sind drei Router mit einer Route zu Netz 1 gezeigt. Fällt nun die Verbindung von $R_1$ zu Netz 1 aus, so bekommt $R_1$ möglicherweise noch von $R_2$ die Broadcast-Nachricht, dass er Netz 1 mit einem Hop erreichen könne, worauf $R_1$ seine Tabelle dahingehend aktualisiert, dass er nun Netz 1 mit 2 Hops erreichen kann usw. Wie man leicht sehen kann, "schaukeln" die beiden Router nun ihre Abstände zu Netz 1 gegenseitig bis zum maximalen Hop Count von 16 hoch, worauf Netz 1 schließlich als unerreichbar eingetragen wird.

Um dieses Problem zu umgehen, hat man die *Split Horizon Technik* eingeführt: Danach wird eine Routing-Information nicht an *alle* benachbarten Router geschickt. Dadurch wird verhindert, dass eine Information an einen Router (zurück) übertragen wird, von dem diese Information stammt. Für bestimmte Situationen dient Split Horizon dem schnelleren Aufbau von stabilen Routing-Tabellen. Eine andere Technik zur Vermeidung der langsamen Konvergenz ist das sogenannte *Hold Down*. Bei diesem Verfahren ignorieren Router für eine bestimmte Zeit (ca. 60 Sekunden) Nachrichten, die einen zuvor als nicht erreichbar gekennzeichneten Router für über eine andere Route erreichbar erklären. Damit soll erreicht werden, dass alle Router "in Ruhe" die Nachricht über eine ausgefallene Verbindung verarbeiten können und nicht immer wieder Wege angepriesen bekommen, die die ausgefallene Verbindung nutzen.

*Das Open Shortest Path First Protocol*

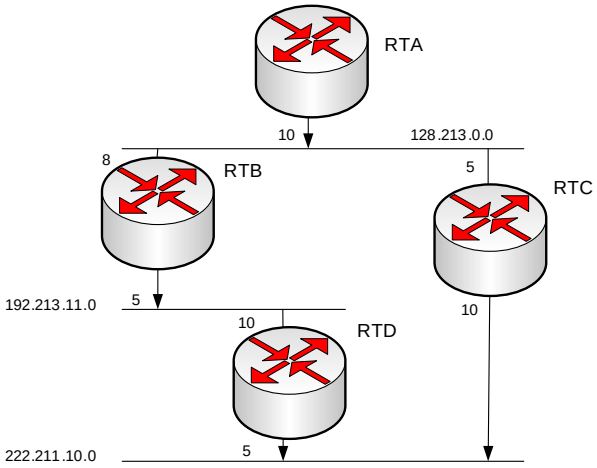
Das *OSPF* hat historisch *RIP* als Standard Interior Gateway Protocol abgelöst, insbesondere bei großen Netzen.

Im Gegensatz zu *RIP*, dass wie oben gesagt zu den *Distance Vector* Protokollen zählt, handelt es sich bei *OSPF* um ein *Link State* Protokoll. Prinzipieller Unterschied ist, dass hierbei zwar jeder Router die gesamte Netztopologie (also den ganzen Graphen) "kennen" muss, dass er als Routing-Informationen jedoch nur den Zustand der direkt an ihn angeschlossenen Verbindungen verbreitet. Aus den erhaltenen Informationen baut jeder Router dann seine Routing-Tabelle auf bzw. aktualisiert diese immer wieder.

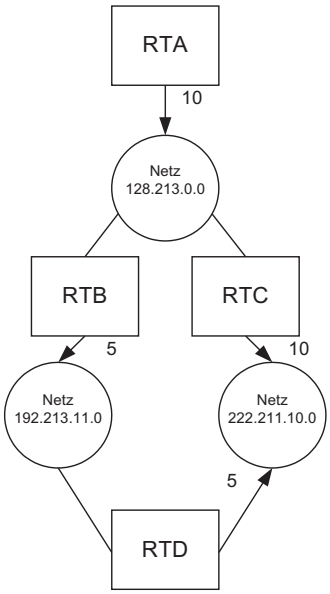
Der genauere Ablauf ist folgendermaßen:

Jeder Router testet den Status aller zu ihm benachbarten Router (genauer gesagt: untersucht die Schnittstelle aus seiner Sicht) – zwei Router sind dabei im zugehörigen Graphen benachbart, wenn sie an einem gemeinsamen Netz angeschlossen sind. Ist der Nachbar erreichbar, so ermittelt er einen dazu gehörenden Kostenwert, der zu der nutzbaren Bandbreite umgekehrt proportional ist; die entsprechende Graphenkante bekommt also eine Bewertung. Ferner verbreitet jeder Router periodisch per Broadcast die Informationen über die von ihm getesteten Links an alle anderen Router. Immer dann, wenn ein Router eine solche *Link Status Nachricht* erhält, aktualisiert er seine eigene "Karte" des Netzes (bzw. Autonomen Systems). Hat sich ein Link-Status geändert, werden die Routen neu berechnet, indem mit dem Algorithmus von Dijkstra ein Baum kürzester Wege aus Sicht des betreffenden Routers aufgestellt wird.

Wir illustrieren das Verfahren an einem kleinen Beispiel, das im folgenden Bild dargestellt ist. Es sind drei Netze (mit *IP*-Adressen, wie bei Ethernet-Netzen üblich einfach durch eine Linie dargestellt) und vier Router zu sehen, wobei wir die Situation aus Sicht des Routers RTA betrachten. (Das Beispiel ist den Internet-Seiten der Firma CISCO entnommen.)

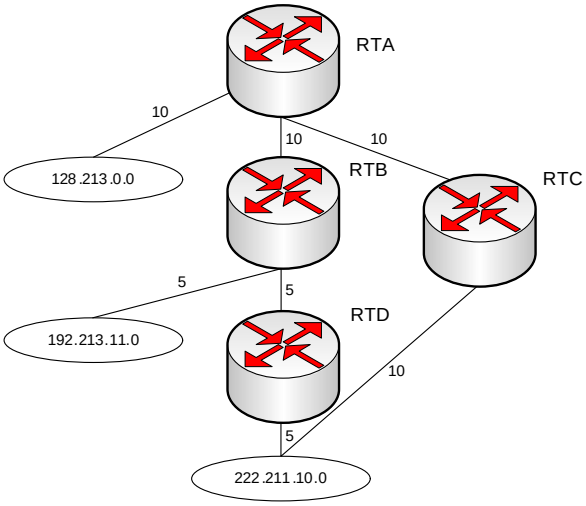



Das Bild zeigt auch die aktuellen Kosten, wobei auf die Richtung der Pfeile zu achten ist: Beispielsweise sind die Kosten der Schnittstelle von RTB nach 128.213.0.0 (mit Wert 8) für die Kosten von RTA nach 192.213.11.0 nicht von Bedeutung. Die für RTA relevante reduzierte Graphenstruktur ist im nächsten Bild noch einmal gezeigt:




Das dritte Bild zeigt nun schließlich das Ergebnis, zu dem RTA für seine Routing-Tabelle mit diesen Informationen kommt. Da RTA in diesem Beispiel das Netz

222.211.10.0 sowohl über RTC als auch über RTB und RTD mit Kosten 20 erreichen kann, werden hier beide Routen gezeigt - m. a. W. beschränkt man sich nicht wirklich auf die Konstruktion eines Baumes. Das Verfahren lässt mehrere Routen zum Ziel zu.




RTA erstellt nun abschließend seine Routing-Tabelle, wobei pro Ziel nur der (oder mehrere mögliche) direkte Nachbar(n) - sozusagen der *Next Hop* - eingetragen wird.

Das *OSPF*-Protokoll hat zunächst den Vorteil, dass jeder Router die Routen auf Basis der gleichen Daten unabhängig berechnet. Da jeder Router die Berechnungen lokal durchführt, gibt es keine Probleme mit der Konvergenz. Weil die Link-Status-Nachrichten nur Informationen über die direkten Verbindungen eines Routers enthalten, ist die Größe dieser Nachrichten auch von der Gesamtgröße des Autonomen Systems (enthaltene Netze) unabhängig. *OSPF* hat eine Reihe weiterer Vorteile, die von denen im folgenden einige aufgezählt werden, ohne dass genauer darauf eingegangen wird:

- *OSPF* schließt sogeanntes *Servicetyp-Routing* ein. Dies bedeutet: Wenn ein Datagramm geroutet wird, nutzt ein Router neben der Zieladresse auch die Servicetyp-Felder des *IP*-Headers, um eine Route zu wählen. Dadurch können mehrere Wege zu demselben Ziel genutzt werden (je nach Servicetyp).

- *OSPF* arbeitet mit *Load Balancing*. Dies bedeutet: Sind zu einer Zieladresse mehrere Routen mit den gleichen Kosten in der Routing-Tabelle festgehalten, so wird der entsprechende Verkehr von dem Router gleichmäßig über alle diese Routen verteilt.

- *OSPF* bietet den Autonomen Systemen die Möglichkeit, ihre Netze und Router in *Bereiche* (*Areas*) einzuteilen. Die Kenntnis der genauen inneren Struktur eines Bereiches bleibt den anderen Bereichen verborgen. (Dies ist analog zu den Autonomen Systemen auf höherer Ebene, deren Interna untereinander ebenfalls verborgen sind.) Dadurch sind die Netze leichter zu verwalten, insbesondere bei Wachstum.
- *OSPF* lässt zu, dass der Nachrichtenaustausch zwischen Routern authentifiziert wird. So wird sicher gestellt, dass nur vertrauenswürdige Router Routing-Informationen propagieren können.
- *OSPF* gibt Administratoren die Möglichkeit, eine virtuelle Netztopologie (also einen zugrunde liegenden Graphen) zu beschreiben, bei der auch dann eine virtuelle Verbindung zwischen zwei Routern (Kante im Graphen) definiert ist, wenn die physische Verbindung zwischen diesen Routern nicht direkt ist (sondern z. B. ein Transitnetzwerk nutzt).

**Selbsttestaufgabe 8.5-1:**

*Erläutern Sie, wieso es im Internet möglich ist, dass eine Nachricht auf einem bestimmten Weg zum Zielknoten läuft, obwohl es dazu einen kürzeren Weg gegeben hätte.*

## 8.6 Das Routing im Zeichengabesystem Nr. 7

**Zeichengabesystem Nr. 7**

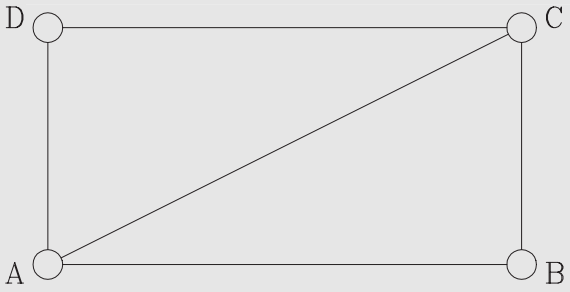
Beim **Zeichengabesystem Nr. 7** handelt es sich um ein international standardisiertes Zentralkanal-Zeichengabesystem. Es ist geeignet für den Einsatz in digitalen Kommunikationsnetzen mit speicherprogrammierten Vermittlungsstellen und arbeitet paketorientiert. Das ZGS Nr. 7 und seine Protokolle sind in den CCITT-Empfehlungen Q.700 bis Q.795 beschrieben (s. Literaturverzeichnis).

ZGS Nr. 7 wurde zunächst konzipiert für "call control signalling" für Telefon, ISDN, leitungsvermittelte Datendienste etc., es ist jedoch auch für andere Informationstypen nutzbar (z. B. zwischen speziellen Rechnern in Kommunikationsnetzen für Zwecke von Management oder Maintenance). Im ISDN-Netz der Deutschen Telekom wird ZGS Nr. 7 für die "Zwischenamtssignalisierung" benutzt. Wir werden im folgenden von dem Anwendungshintergrund weitgehend abstrahieren und ZGS Nr. 7 als eine weitere Variante der Paketvermittlung betrachten, auf deren Art von Routing wir unser besonderes Augenmerk richten.

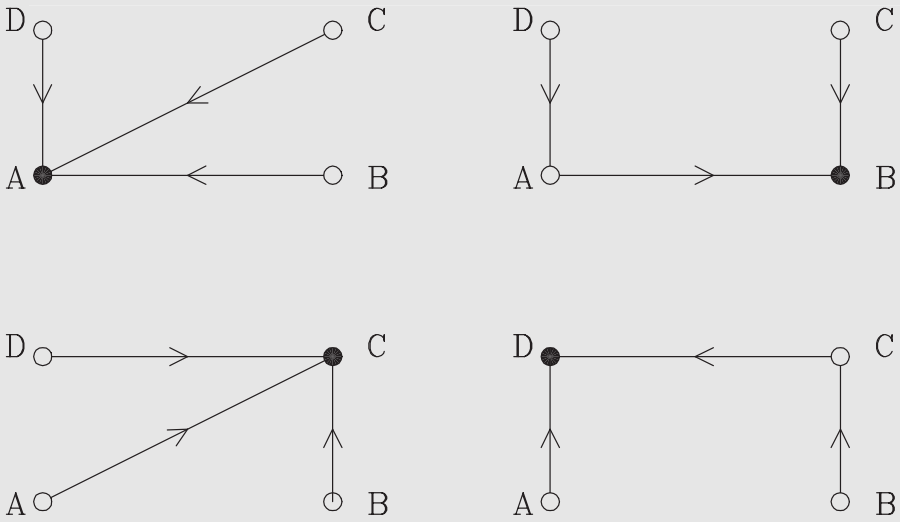
Das Routing im ZGS Nr. 7 kann als statisch bezeichnet werden. Die extremste Form statischen Routings hatten wir bereits in Abschnitt 8.1 betrachtet: für jeden Zielknoten X wird ein Baum kürzester Wege $B_X$ erstellt, und in jedem Knoten werden **Routing-Tabellen** gespeichert, die diesen Bäumen entsprechend eingerichtet sind.

**Routing-Tabellen**

**Beispiel 8.6-1:**
Wir betrachten das folgende Netz. (Alle Kanten haben die Länge 1.)




Für jeden Knoten $X$ sei ein Baum kürzester Wege (für das Ziel $X$) ausgewählt:




Die Routing-Tabellen sehen also folgendermaßen aus:

In Knoten A:

| Ziel | B | C | D |
|---|---|---|---|
| Nachbar | B | C | D |

In Knoten B:

| Ziel | A | C | D |
|---|---|---|---|
| Nachbar | A | C | C |

In Knoten C:

| Ziel | A | B | D |
|---|---|---|---|
| Nachbar | A | B | D |

In Knoten D:

| Ziel | A | B | C |
|---|---|---|---|
| Nachbar | A | A | C |

**festes Routing**

Die soeben beschriebene Art des Routing könnte man als **festes Routing** bezeichnen. In der englischen Literatur wird es als "fixed routing" oder auch "directory routing" aufgeführt. Hat man es mit einem Netz zu tun, in dem wenig Verkehr abgewickelt wird und die Knoten und Kanten recht zuverlässig sind, so ist dieses Routing einfach und leistungsfähig.

Die Nachteile des Verfahrens liegen auf der Hand. Bereits in Abschnitt 8.1 wurde die Tatsache angesprochen, dass der Ausfall einer Kante sofort gewisse Kommunikationsbeziehungen stört, da keine Ersatzwege vorgesehen sind. Eine Verbesserungsmöglichkeit, von der in einigen Netzen Gebrauch gemacht wurde, liegt darin, dass jeder Knoten mehrere Routing-Tabellen gespeichert hat, zwischen denen er umschalten kann, wenn er über gewisse Ausfälle Kenntnis erlangt. (Dies hat starke ähnlichkeit mit dem im ZGS Nr. 7 angewendeten Verfahren, welches im folgenden behandelt wird.)

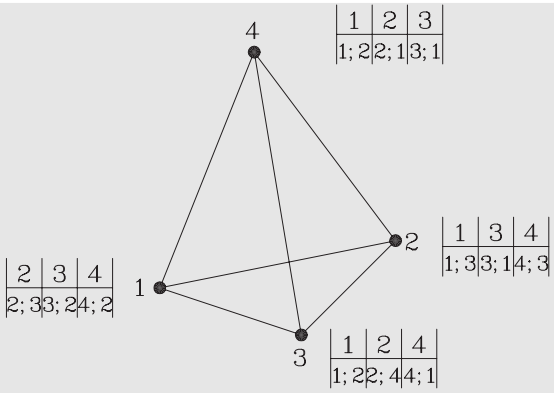
Bei homogenem Verkehrsaufkommen im Netz werden die Kanten u. U. – je nach Auswahl der Bäume – sehr ungleichmäßig belastet sein. Es ist keine leichte Aufgabe, insbesondere auch bei unterschiedlichem Verkehrsaufkommen, die Wahl der Bäume in diesem Sinne günstig zu treffen. Eine Möglichkeit besteht auch darin, den Verkehr zu einem Ziel in einem prozentual festen Verhältnis über mehrere Nachbarknoten abzuwickeln - dies bedeutet dann ein Abrücken von der starren durch die Bäume gelieferten Routing-Regel.

**Bidirektionalität**

Ein weiteres Problem liegt schließlich darin, dass die **Bidirektionalität** verletzt sein kann: fällt im obigen Beispiel die Kante AB aus, so kann (wenn nicht auf andere Tabellen umgeschaltet wird) D keine Nachrichten mehr an B versenden, jedoch B noch welche an D. Auch das Problem der Bidirektionalität werden wir beim ZGS Nr. 7 aufgreifen.

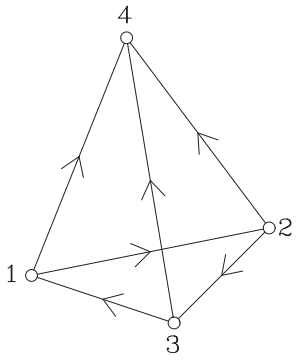
Beim Routing im ZGS Nr. 7 hat jeder Knoten ebenfalls Routing-Tabellen. Der Eintrag für einen Zielknoten besteht allerdings nicht nur aus einem Nachbarknoten, zu dem der betreffende Verkehr weitergereicht wird, sondern aus einer geordneten Liste mehrerer Nachbarknoten. Die Handhabung ist folgendermaßen: ein Paket, welches von Knoten X weitergereicht werden soll und die Zieladresse Z trägt, wird von X an den ersten ihm aktuell verfügbaren Nachbarn aus seiner Routing-Tabelle für Ziel Z geschickt.

**Beispiel 8.6-2:**
(s. auch Beispiel 6.1-8)
Das folgende Netz mit Routing-Tabellen mit jeweils zwei Einträgen sei gegeben:




Hat also z. B. Knoten 1 eine Nachricht mit Zieladresse 4 zu versenden, so schickt er diese unter normalen Umständen direkt an 4; ist dies jedoch nicht möglich (etwa wegen Ausfalls der Kante 14), so schickt er sie an den Knoten 2, usw.

Es ist offensichtlich, dass bei dieser Art von Routing die Wege zu einem Ziel Z nicht – wie bei festem Routing – einen Baum bilden. Bei Beispiel 8.6-2 bekommt man vielmehr einen gerichteten Graphen, bei dem jeden von Z verschiedenen Knoten zwei gerichtete Kanten verlassen. Für den Knoten 4 als Ziel ergibt sich beispielsweise der folgende gerichtete Graph, den wir später mit $G(\rightarrow 4)$ bezeichnen werden:




Je nach Fragestellung können auch die Prioritäten der einzelnen gerichteten Kanten mit berücksichtigt werden (etwa als Kantenbewertung).

An dieser Stelle ist es nötig, kurz einige der für das Routing relevanten Strukturen und Protokollabläufe des ZGS Nr. 7 zu betrachten.

Im obigen Beispiel 8.6-2 bestehen die Listen in den Routing-Tabellen aus jeweils zwei Einträgen. Diese Anzahl ist in den CCITT-Empfehlungen nicht festgelegt. Es gibt Implementierungen, bei denen jeweils acht Tabellenplätze vorgesehen sind, die jedoch nicht alle gefüllt werden müssen. Im Gegenteil werden unsere Überlegungen bald zeigen, dass niemals von jedem Knoten für jedes Ziel acht Möglichkeiten vorgesehen sein dürfen! (Im folgenden werden wir bei der mathematischen Formalisierung von einer Höchstzahl von $\pi$ und Mindestzahl von 2 Tabellenplätzen ausgehen. Es zeigt sich allerdings, dass schon für $\pi = 2$ die Probleme genügend komplex werden, um die meisten Effekte daran studieren zu können.)

**transfer prohibited procedure**

Von zentraler Bedeutung für das Routing ist die **transfer prohibited procedure**, die u. a. dafür sorgt, dass sich Informationen über ausgefallene Teile des Netzes verbreiten. Genauer: ein Knoten, der feststellt, dass er ein gewisses Ziel nicht mehr erreichen kann, teilt dies allen seinen Nachbarn mit; ein Nachbar wird dann – je nach Routing-Tabelle – u. U. ebenfalls zu dem Schluss kommen müssen, dass er keine Nachrichten mehr zu diesem Ziel schicken kann. Die Prozedur bedient sich einer mit TFP(Z) bezeichneten Nachricht. Ein Knoten X, der diese Nachricht von seinem Nachbarn Y empfängt, schließt daraus, dass er diesen Nachbarn nicht für Nachrichten, die für das Ziel Z bestimmt sind, nutzen darf. Umgekehrt stellt eine periodisch ausgeführte "route set test procedure" sicher, dass wiederhergestellte Routen auch wieder genutzt werden können. Nachrichten, die nicht weitergeleitet werden können (z. B. weil keiner der in Frage kommenden Nachbarknoten erreichbar ist), werden verworfen.

Die "transfer prohibited procedure" (TFP) verhindert auch das Pendeln von Nachrichten auf einer Kante. Genauer sagt die Empfehlung folgendes (s. 13.2.2 in CCITT-Empfehlungen Q.704):

> *Falls Knoten X anfängt (z. B. nach einer Netzänderung), für Knoten Z bestimmte Nachrichten an seinen Nachbarn Y zu senden, so schickt er zuvor die Nachricht TFP(Z) an Y.*

Der Empfang von TFP(Z) in Y (mit Absender X) ist –wie oben ausgeführt– von Y so zu interpretieren, dass Y nun (bis auf weiteres) seinen Nachbarn X nicht für Nachrichten mit Ziel Z nutzen darf.

Die im ZGS Nr. 7 installierten Flusskontrollmechanismen haben auf das Routing keinen direkten Einfluss. Auf der Schicht 2 können zwar Kanten überlastet sein, für das Routing in Schicht 3 stellt sich dies jedoch einfach als Nichtverfügbarkeit einer Kante zu einem Nachbarn dar. Ein anderer Mechanismus, den das Schicht 3-Routing gar nicht "mitbekommt", besteht darin, bei stark belasteten Routen die höheren Schichten von Ursprungsknoten "aufzufordern", ihren von diesen Routen zu tragenden Verkehr zu reduzieren.

**Lastteilung**

Wir lassen bei unseren Betrachtungen zum Routing im ZGS Nr. 7 eine Möglichkeit außer Betracht, die von der CCITT-Empfehlung Q.704 eingeräumt wird und in Implementierungen auch realisiert ist: die Möglichkeit der **Lastteilung**. Darunter ist zu verstehen, dass die Nachrichten für ein Ziel nicht sämtlich zu dem laut

Tabelle ersten verfügbaren Nachbarn gesendet werden, sondern dass dieser Verkehr gleichmäßig auf zwei oder mehr Nachbarn aufgeteilt wird.

Für unsere weitere Vorgehensweise wird es notwendig sein, die für das Routing im ZGS Nr. 7 relevanten Begriffe etwas weitergehender zu formalisieren, um dann interessierende Aussagen mathematisch beweisen zu können. Da wir andererseits nicht überformalisieren wollen (mit den bekannten negativen Folgen), wird es gelegentlich nötig sein, die ZGS Nr. 7 - Protokolle heranzuziehen, um über die Randbedingungen und Ziele der Untersuchungen Klarheit herzustellen.

Für den Rest dieses Abschnitts gehen wir immer davon aus, dass ein zusammenhängeder ungerichteter Graph $G$ mit $n$ Ecken gegeben ist. Der Einfachheit halber wird angenommen, dass die Ecken mit den Zahlen $1, 2, ..., n$ bezeichnet sind. Die Kanten sind nicht bewertet. Die Knoten sind alle von gleicher Art, d. h. können sämtlich Nachrichten versenden, empfangen oder weiterleiten. Letzteres stellt eine Vereinfachung gegenüber den ZGS Nr. 7 - Empfehlungen dar, die verschiedene Arten von Knoten erlauben (z. B. solche, die keine Nachrichten weiterleiten können).

**Routing-Plan**

Ein **Routing-Plan** mit $\pi$ Prioritäten für $G$ ist eine dreidimensionale Anordnung

$$C = (c(i, j, k) \mid i = 1, ..., \pi; j, k = 1, ..., n).$$

Dies ist so zu lesen: $c\ (i, j, k)$ ist der Nachbarknoten von $j$, der für Nachrichten mit Ziel $k$, die von $j$ zu versenden sind, als i-te Priorität zur Verfügung steht. (Diese Definition ist aus [KLE] übernommen und auch in [POG1, POG2] verwendet worden.)

Wir nehmen stets $c(1, j, k) \neq c(2, j, k)$ an, jedoch kann für $i \geq 2$ gelten $c(i, j, k) = c(i + 1, j, k)$. Dies bedeutet, dass dem Knoten $j$ für Ziel $k$ mindestens zwei Nachbarn zur Verfügung stehen – und möglicherweise keine weiteren. Eine weitere Annahme ist folgende: existiert in $G$ die Kante $jk$, so gilt stets $c(1, j, k) = k$ – wenn m. a. W. eine direkte Kante zum Zielknoten führt, so wird diese auch mit erster Priorität genutzt.

**Routing-Tabellen**

Wird in $C$ ein $j$ fest gewählt, so erhält man die (von $j$ verwendeten) **Routing-Tabellen**.

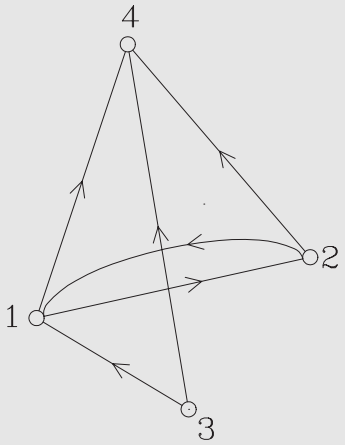
**Beispiel 8.6-3:**
In Beispiel 8.6-2 sind die Routing-Tabellen der einzelnen Knoten angegeben. Die zu dem Parameter i gehörende "dritte Dimension" ist dabei als Folge (durch Semikolon getrennt) dargestellt. Insgesamt hat man hier den folgenden Routing-Plan, in dem j mit den Zeilen und k mit den Spalten variiert:

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | - | 2;3 | 3;2 | 4;2 |
| 2 | 1;3 | - | 3;1 | 4;3 |
| 3 | 1;2 | 2;4 | - | 4;1 |
| 4 | 1;2 | 2;1 | 3;1 | - |

**Beispiel 8.6-4:**
Für den in Beispiel 8.6-2 betrachteten Graphen $K_4$ sei nun der folgende Routing-Plan gegeben:

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | - | 2;3 | 3;2 | 4;2 |
| 2 | 1;3 | - | 3;1 | 4;1 |
| 3 | 1;2 | 2;1 | - | 4;1 |
| 4 | 1;2 | 2;1 | 3;1 | - |

Für den Knoten 4 als Ziel ergibt sich der folgende gerichtete Graph:




Die Ausführungen zur "transfer prohibited procedure" machen klar, dass in diesem Beispiel der gerichtete Kreis auf der Kante 12 aufgrund der Protokolle nicht zu einem Problem führt. Anders verhält es sich mit dem sich aus Beispiel 8.6-2 ergebenden Kreis $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$, denn über mehr als zwei Stationen laufende Kreise werden von den Protokollen nicht "bemerkt". Auch wird keiner der beteiligten Knoten zu dem Schluss kommen, dass er das Ziel 4 nicht mehr erreichen kann. Wir werden auf dieses Problem ausführlich eingehen.

Für die nun folgenden Untersuchungen ist es nicht notwendig, eng bei dem Beispiel des Zeichengabesystems Nr. 7 zu verbleiben. Wir stellen noch einmal die Grundvoraussetzungen und Annahmen zusammen, von denen ausgegangen wird:

1. Es ist ein ungerichteter Graph G mit Ecken $1, ..., n$ gegeben, ferner eine natürliche Zahl $\pi \geq 2$ als Anzahl der Prioritäten in zu findenden Routing-Plänen *C* (im Sinne der obigen Definition).

2. Es wird von Netzprotokollen mit Datagramm-Modus ausgegangen, die einen Routing-Plan *C* wie oben beschrieben nutzen.

3. Die Protokolle verhindern das Pendeln von Paketen auf einer Kante, größere Kreise müssen von den Routing-Plänen ausgeschlossen werden.

4. Ausfälle von Knoten oder Kanten werden sofort an allen relevanten Stellen registriert, gleiches gilt für die Wiederherstellung von Netzkomponenten – m. a. W. es werden keine Reaktionszeiten berücksichtigt.

Der obige Punkt 4. stellt natürlich eine starke Abstrakion dar, welche die Gültigkeit einiger Aussagen wieder relativiert. Z. B. tragen Reaktionszeiten ja auch zu den Paketverzögerungszeiten im Netz bei, die ein Kriterium für ein gutes Routing darstellen. Wir kommen auf diesen Punkt später noch einmal zurück.

Für das weitere Vorgehen ist es auch nötig, die Ziele des Routing zu formulieren, die letzlich in Forderungen an die Routing-Pläne umzusetzen sind. Im nächsten Abschnitt werden allgemeine Ziele und sich daraus ergebende Forderungen betrachtet. Hier wollen wir uns zunächst damit begnügen, einige spezielle Kriterien für "gute" Routing-Pläne zu benennen und herauszuarbeiten, wie diese erfüllt werden können bzw. untereinander gegenläufig sind. Auf eines muss an dieser Stelle noch einmal hingewiesen werden: Obschon alle vorkommenden Graphen endliche Strukturen sein werden, sind die angesprochenen Probleme nicht einfach zu lösen, da wir den algorithmischen Standpunkt einnehmen und folglich "leichte Berechenbarkeit" unser Kriterium für "einfache Lösung" ist.

Die folgenden Kriterien werden zuerst betrachtet. Ein Routing-Plan $C$ soll möglichst so ausgelegt sein, dass

- niemals eine Nachricht im Netz kreisen kann (wir sagen dann, $C$ sei "kreisfrei");
- eine Nachricht stets einen möglichst kurzen Weg zum Ziel nimmt;
- stets Bidirektionalität herrscht (d. h. falls $X$ an $Y$ Nachrichten schicken kann, ist dies auch umgekehrt möglich).

Ein weiteres Kriterium ist natürlich eine hohe Zuverlässigkeit der Kommunikationswege (in Abhängigkeit von der Verfügbarkeit von Knoten und Kanten). Betrachtungen dazu werden hier zunächst ausgespart, da sie Inhalt des nächsten Kapitels sind. Auch das Ziel einer möglichst gleichmäßigen Auslastung von Netzkomponenten wird dann zur Sprache kommen.

Wir gehen hier stets von einer gegebenen Netzstruktur (sprich von einem Graphen *G*) aus. In der Realität mag es durchaus auch so sein, dass von gewissen Grundanforderungen abgesehen auch die präzise Struktur noch gewählt werden kann. Dann hat man natürlich die Möglichkeit, das im Sinne der Kriterien beste Netz auszuwählen. Auch die sich daraus ergebende Problematik des Netzdesigns wird an späterer Stelle noch einmal angesprochen.

Es sei nun ein Routing-Plan $C$ für $G$ gegeben, $k$ sei ein Knoten in $G$. Offenbar wird hierdurch ein gerichteter Graph für Knoten $k$ als Ziel induziert, den wir mit $G(\rightarrow k)$ bezeichnen wollen; es existiert in $G(\rightarrow k)$ eine gerichtete Kante $x \rightarrow y$ von Knoten $x$ nach Knoten $y$ genau dann, wenn es ein $i \in \{1, ..., \pi\}$ gibt mit $c(i, x, k) = y$.

In Beispiel 8.6-2 ist beispielsweise der zu dem angegebenen Routing-Plan gehörende Digraph $G(\rightarrow 4)$ dargestellt.

Man beachte, dass in $G(\rightarrow k)$ nicht jede Kante von $G$ (mit Orientierung versehen) verwendet werden muss. Andererseits gibt es natürlich zu jeder Kante ein $k$ derart, dass die Kante in $G(\rightarrow k)$ verwendet wird. Vernünftigerweise wird man an einen Routing-Plan auch die Forderung stellen wollen, dass jeder Knoten überhaupt jeden anderen erreichen kann – dies bedeutet, dass in jedem $G(\rightarrow k)$ von jedem Knoten $x \neq k$ ein gerichteter Weg nach $k$ führt. Interessanterweise ist dies im Falle der Kreisfreiheit automatisch erfüllt.

Man kann auch dazu übergehen, jede gerichtete Kante in $G(\rightarrow k)$ mit ihrer Priorität $i$ zu bewerten (s. [POG1]). Da dies im weiteren nicht benutzt wird, sehen wir hier davon ab.

Die im weiteren Verlauf dieses Abschnitts dargestellten Ergebnisse sind sämtlich [POG1] und [POG2] entnommen. Dabei ist die Anzahl $\pi$ der möglichen Tabelleneinträge zunächst als beliebig angenommen.

**Satz 8.6-1:** *Ein Routing-Plan $C$ für $G$ ist genau dann kreisfrei, wenn kein gerichteter Graph der Form $G(\rightarrow k)$, bei dem $k$ eine beliebige Ecke von $G$ ist, einen gerichteten Kreis durch mehr als zwei Knoten enthält.*

**Beweis:** Ist $C$ nicht kreisfrei, d. h. kann es eine Nachricht (mit Zieladresse $k_0$) geben, die im Netz kreist, so muss es im gerichteten Graphen $G(\rightarrow k_0)$ einen Kreis durch mehr als zwei Knoten geben, da sich die Nachricht ja "in diesem Graphen" längs gerichteter Kanten bewegt.

Nun gebe es umgekehrt einen Knoten $k_1$ mit einem gerichteten Kreis $x_1 \rightarrow x_2 \rightarrow ...x_r \rightarrow x_1$ $(r > 2)$ in $G(\rightarrow k_1)$. O. B. d. A. hat $x_2$ für $x_1$ bezüglich des Ziels $k_1$ eine höhere Priorität als $x_r$ (sofern die Kante $x_1 \rightarrow x_r$ überhaupt existiert). Wäre dies nicht der Fall für $x_1$ oder entsprechend eines der anderen $x_i$, so könnten wir statt dessen denselben in umgekehrter Richtung durchlaufenen Kreis betrachten. Fallen nun alle Kanten mit Ausnahme der zu dem Kreis gehörenden aus, und initiiert $x_1$ eine Nachricht für das Ziel $k_1$, so wird diese Nachricht den obigen Kreis oder einen Teilkreis mit wenigstens drei Knoten durchlaufen – mindestens so lange, bis eine weitere Netzänderung eintritt.

Der nächste Satz liefert eine wichtige notwendige Bedingung für die Kreisfreiheit eines Routing-Plans:

**Satz 8.6-2:** *Ist $C$ kreisfrei und $k$ beliebiger Knoten, so gibt es in dem gerichteten Graphen $G(\rightarrow k)$ mindestens einen von $k$ verschiedenen Knoten mit nur genau zwei Nachfolgern.*

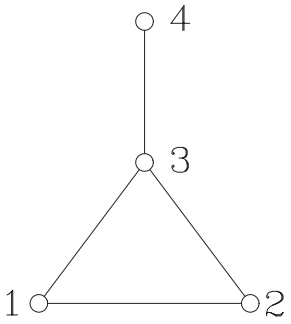
**Beweis:** Wir stellen uns vor, wir entnehmen dem gerichteten Graphen $G(\rightarrow k)$ alle Kanten mit $k$ als Endpunkt. Im Restgraphen $G'$ hat jeder Knoten $x \neq k$, wenn die Aussage des Satzes nicht gilt, immer noch mindestens zwei Nachfolger. Startet man nun mit einer beliebigen Kante, und konstruiert man dann schrittweise einen gerichteten Weg, wobei eine neue Kante niemals direkt zum Anfangspunkt der letzten Kante zurückführt, so kommt man wegen der Endlichkeit des Graphen nicht umhin, einen gerichteten Kreis mit mehr als zwei Knoten zu konstruieren.

Der soeben bewiesene Satz hat die Konsequenz, dass es, wie groß auch immer die Zahl $\pi$ der möglichen Tabellenplätze für ein Ursprung-Ziel-Paar ist, bei einem kreisfreien Routing-Plan in jeder Spalte (die einem Ziel $k$ entspricht) mindestens eine Zeile mit nur genau zwei Einträgen gibt.

Nun wissen wir bereits aus Kapitel 4: ist ein Routing-Plan $C$ gegeben (und damit auch die gerichteten Graphen $G(\rightarrow k)$), so ist es (unter Beachtung von Satz 8.6-1) algorithmisch nicht schwierig zu ermitteln, ob $C$ kreisfrei ist. In [KLE] wird auf solche Tests etwas ausführlicher eingegangen. Wir wollen uns nun jedoch der interessanteren Frage zuwenden, wie man (bei gegebenem $G$) ein kreisfreies $C$ konstruiert – und zwar auch noch ein im Sinne der weiteren Kriterien möglichst gutes! Leider können wir dieses Problem derzeit nicht umfassend lösen.

Schon die Frage, für welche Graphen $G$ (bei gegebenem $\pi$) ein kreisfreier Routing-Plan $C$ überhaupt existiert, ist nicht befriedigend zu beantworten: wir können diese Graphen bisher weder mathematisch elegant charakterisieren, noch können wir einen guten Algorithmus angeben, der diese Graphen "erkennt". Man beachte, dass es für gegebenes $G$ und $\pi$ auf jeden Fall exponentiell viele dreidimensionale Strukturen ($c(i,j,k)$) gibt, die einen kreisfreien Routing-Plan darstellen können.

Eine notwendige Bedingung an $G$ ist allerdings unschwer abzuleiten. Man sieht sofort, dass es für den folgenden Graphen G keinen kreisfreien Routing-Plan geben kann:




Da nämlich für jeden Routing-Plan in dem induzierten Digraphen $G(\rightarrow 4)$ jeden der Knoten 1, 2, 3 mindestens zwei gerichtete Kanten verlassen, entsteht auf jeden Fall ein gerichteter Kreis.

Die notwendige Bedingung lautet:

**Satz 8.6-3:** *Existiert für $G$ (und $\pi \geq 2$) ein kreisfreier Routing-Plan, so ist $G$ 2-fach-kantenzusammenhängend, d. h. zwischen beliebigen zwei Knoten von $G$ gibt es stets zwei kantendisjunkte Wege.*

**Beweis:** $C$ sei ein solcher Routing-Plan, $x$ und $y$ seinen Knoten von $G$. Verfolgt man in $G(\rightarrow y)$, von $x$ startend, lauter gerichtete Kanten von Priorität 1 bis zum Ziel $y$ und entfernt dann diese Kanten, so muss es im Restgraphen immer noch einen gerichteten Weg von $x$ nach $y$ geben.

Wir wollen nun das Problem mit einbeziehen, die Routing-Pläne so zu konstruieren, dass die Nachrichten stets auf möglichst kurzen Wegen zu ihrem Ziel laufen. Es stellt sich heraus, dass die Festlegung eines Routing-Plans eine so starke Einschränkung ist, dass dieses Ziel unmöglich zu ereichen ist.

Für die nächsten Überlegungen sei der Einfachheit halber $\pi = 2$ gesetzt. Nun sei ein 2-fach-kantenzusammenhängender Graph $G$ gegeben. Ein Routing-Plan $C$ wird auf folgende Weise konstruiert:

Für beliebige Knoten $j$ und $k$ (mit $j \neq k$) bestimme man einen kürzesten Weg von $j$ nach $k$. Ist $l$ der Nachbar von $j$ auf diesem Weg, so setze man $c\,(1, j, k) = l$. Dann entferne man die Kante $jl$ aus dem Graphen und wende dieselbe Prozedur auf den Restgraphen an, um $c\,(2, j, k)$ zu erhalten.

**SP-Routing**

Wir bezeichnen diesen Prozess als **SP-Routing** (von "shortest path"). Da es u. U. mehrere kürzeste Wege gleicher Länge gibt, führt SP-Routing nicht zu einem eindeutigen Ergebnis. Z. B. sind die in Beispiel 8.6-2 und Beispiel 8.6-4 dargestellten Routing-Pläne für den Graphen $K_4$ beide mit SP-Routing erstellt.

Die Frage ist nun, wie gut SP-Routing wirklich ist. Man sieht leicht, dass man es anwenden muss, wenn die Wege kurz werden sollen:

**Beobachtung:** Stellt ein Routing-Plan $C$ sicher, dass eine Nachricht stets einen kürzesten Weg (aus den derzeit im Graphen verfügbaren) wählt, so ist der Plan mit SP-Routing erstellt worden.

Man sieht das folgendermaßen ein. Knoten $j$ und $k(j \neq k)$ seien gegeben, das Netz sei voll intakt. Eine von $j$ nach $k$ gesendete Nachricht nimmt einen kürzesten Weg, mithin muss die Kante von $j$ nach $c\,(1, j, k)$ der Beginn eines kürzesten Weges sein. Analog schließt man für die zweite Priorität mit der Annahme, dass die Kante von $j$ nach $c\,(1, j, k)$ ausgefallen ist.

Umgekehrt ist nun aber SP-Routing nicht hinreichend dafür, dass stets kürzeste Wege genommen werden. Fallen beispielsweise im Routing-Plan in Beispiel 8.6-4 die Kanten 14 und 34 aus, und versendet Knoten 3 eine Nachricht mit Ziel 4, so wird diese Nachricht den Weg $3 \rightarrow\ 1 \rightarrow\ 2 \rightarrow 4$ nehmen, obwohl der kürzere Weg $3 \rightarrow\ 2\ \rightarrow 4$ (physikalisch) verfügbar ist.

Die Erklärung für diesen Effekt liegt darin, dass SP-Routing nicht über die Anfänge der Wege "hinausdenkt", und wegen der Beschränktheit fester Routing-Pläne lässt sich dies auch grundsätzlich nicht reparieren.

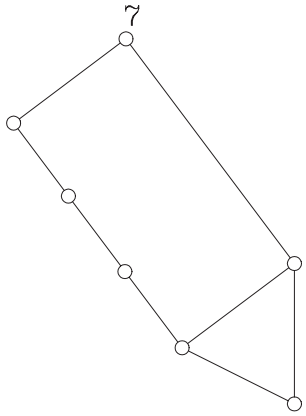
Wir wenden uns nun kurz der Frage zu, inwieweit SP-Routing wenigstens Kreisfreiheit garantieren kann.

**SP-Routing mit Präferenz**

Es wird folgende Sprechweise eingeführt. Wird SP-Routing angewendet mit der Zusatzregel, dass bei der Existenz mehrerer kürzester Wege derjenige Nachbar mit der niedrigsten Knotennummer ausgewählt wird, so sprechen wir von **SP-Routing mit Präferenz**.

Die Einführung einer Präferenz macht SP-Routing eindeutig. Man beachte: der in Beispiel 8.6-4 verwendete Routing-Plan ist nach SP-Routing mit Präferenz erstellt worden.

Schon Beispiel 8.6-2 zeigte, dass SP-Routing keine Kreisfreiheit garantiert. Auch Verwendung von Präferenz reicht nicht aus, wie folgender Graph zeigt:




Bei beliebiger Numerierung der übrigen Knoten mit 1 bis 6 wird auch SP-Routing mit Präferenz zu einem Kreis führen.

Es zeigt sich jedoch, dass eine schwache strukturelle Bedingung an $G$ ausreicht, um dies auszuschließen. Der (kürzeste) Abstand zwischen Knoten $x$ und $y$ ist im folgenden mit $d(x, y)$ bezeichnet:

**Satz 8.6-4:** *Für jeden Knoten $x$ des Graphen $G$ gelte, dass jeder Knoten $v \neq x$ mindestens zwei verschiedene Nachbarn $v_1$,$v_2$ besitzt mit*

$$d(v_1, x),\ d(v_2, x)\ \leq\ d(v, x).$$

*Dann führt SP-Routing mit Präferenz zu kreisfreiem Routing.*

**Beweis:** Angenommen, ein $G(\rightarrow x)$ enthält einen Kreis $w_1 \rightarrow w_2 \rightarrow ...w_r \rightarrow w_1$ (mit $r \geq 3$). Offensichtlich muss $d(w_j, x) = d(w_i, x)$ für alle $i, j \in \{1, ..., r\}$ gelten, und der Kreis besteht aus lauter Kanten zweiter Priorität. Es muss ein $i \in \{1, ..., r\}$ geben mit $w_{i+1} \geq w_{i-1}$, wobei die Indizes modulo $r$ zu berechnen sind. Dies bedeutet aber, dass für $w_i$ der Nachbar $w_{i-1}$ statt $w_{i+1}$ die zweite Priorität für Ziel $x$ hätte sein müssen, man hat also einen Widerspruch.

Der obige Satz ist anwendbar, wenn $G$ vollständig ist (also $G \cong K_n$). Der allgemeine Nachteil von SP-Routing mit Präferenz ist eine ungleichmäßige Belastung des Netzes, falls Ausfälle auftreten. Wir kommen auf diesen Punkt an späterer Stelle zurück.

**Bidirektionalität**

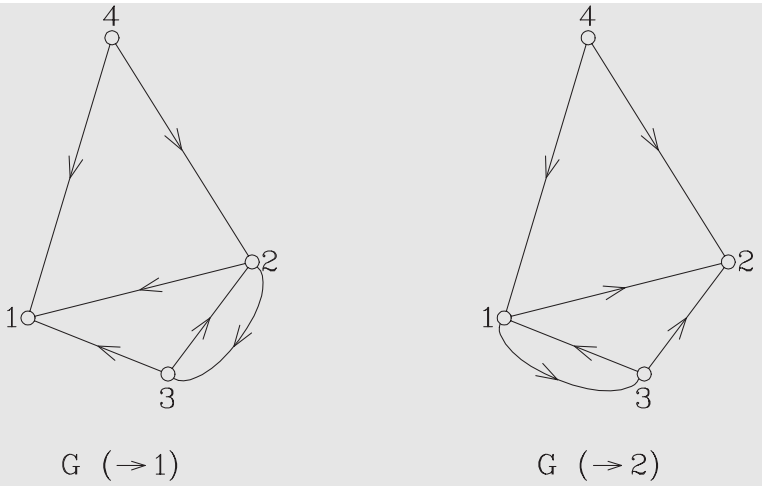
Zum Ende dieses Abschnitts wollen wir uns mit der **Bidirektionalität** beschäftigen. Hauptergebnis wird sein, dass diese sich u. U. nicht mit der Kreisfreiheit verträgt. Wir erinnern an den Begriff: ein Routing-Plan C heißt **bidirektional**, falls immer gilt – unabhängig davon, welche Knoten oder Kanten gerade ausgefallen sind (d. h. nicht benutzt werden dürfen): kann Knoten $x$ Nachrichten nach Knoten $y$ schicken, so auch $y$ nach $x$.

**bidirektional**

Folgende **Beobachtung** halten wir fest:

$C$ ist offenbar genau dann bidirektional, wenn die Menge aller Wege von $x$ nach $y$ in $G(\rightarrow y)$ gleich der Menge aller Wege von $y$ nach $x$ in $G(\rightarrow x)$ – selbstverständlich in umgekehrter Richtung durchlaufen – ist.

**Beispiel 8.6-5:**
$C$ sei der Routing-Plan für $K_4$ aus Beispiel 8.6-4. Es werden die Knoten 1 und 2 angeschaut:




In $G(\rightarrow 1)$ ist $\{2 \rightarrow 1, 2 \rightarrow 3 \rightarrow 1\}$ die Menge aller Wege von 2 nach 1. Diese stimmt (bis auf die Richtungen) überein mit der Menge aller Wege von 1 nach 2 in $G(\rightarrow 2)$, letztere ist nämlich $\{1 \rightarrow 2, 1 \rightarrow 3 \rightarrow 2\}$. Knoten 1 und Knoten 2 können sich also stets gemeinsam erreichen oder nicht erreichen.

Dieser Routing-Plan ist jedoch nicht bidirektional: $4 \rightarrow 1 \rightarrow 2 \rightarrow 3$ ist Weg von $4$ nach $3$ in $G(\rightarrow 3)$, jedoch ist $3 \rightarrow 2 \rightarrow 1 \rightarrow 4$ kein Weg in $G$ (→4). Fallen also alle Kanten bis auf die des Weges $4 \rightarrow 1 \rightarrow 2 \rightarrow 3$ aus, so kann 4 noch 3 erreichen, jedoch nicht umgekehrt.

**Satz 8.6-5:** *Für den Graphen $K_4$ gibt es keinen Routing-Plan (mit $\pi = 3$), der kreisfrei und bidirektional ist.*

**Beweis:** Wir nehmen an, es gebe einen solchen Routing-Plan $C$. Wir wissen, dass für jeden Knoten $k$ $G(\rightarrow k)$ ein azyklischer Diagraph ist, in dem mindestens ein $x \neq k$ nur zwei Nachfolger hat. (Alle $x \neq k$ haben mindestens zwei Nachfolger.) Folgende Bezeichnung wird nun verwendet: für Knoten $j, k$ ist $c(j,k)$ die Menge der von $j$ für Ziel $k$ verwendeten Nachbarn, formal

$$c(j,k) = \{x | x \text{ Knoten, und es gibt ein } i \in \{1,2,3\} \text{mit } c(i,j,k) = x\}.$$

Die Bidirektionalität von $C$ schlägt sich nun in einer Symmetrieeigenschaft dieser Mengen nieder. Anwendung der obigen Mengenaussage aus auf Wege der Länge 2 liefert

$$c(j,k) \setminus \{k\} = c(k,j) \setminus \{j\}$$

für beliebige $j$ und $k$. Anwendung der Mengenaussage aus der obigen Beobachtung auf Wege der Länge 3 ergibt die Aussage: ist $i \in c(j,k)$ und $h \in c(i,k)$, so gilt $h \in c(k,j)$ und $i \in c(h,j)$.
Wir stellen fest, dass es in jedem $G(\rightarrow k)$ einen Weg der Länge 3 geben muss. Andernfalls wären die drei Knoten außer $k$ in disjunkte Paare aufgeteilt (mit jeweils Doppelkanten dazwischen), was unmöglich ist.
O. B. d. A. nehmen wir nun an, es gebe den Weg $1 \rightarrow 3 \rightarrow 4 \rightarrow 2$ in $G(\rightarrow 2)$, also $3 \in c(1,2)$ und $4 \in c(3,2)$. Es folgt $4 \in c(2,1)$ und

$3 \in c(4,1)$. Durch Blick auf die Teilwege von Länge 2 ergibt sich auch $3 \in c(2,1)$, $4 \in c(2,3), 4 \in c(1,2)$, und $3 \in c(1,4)$. Da $G(\rightarrow 2)$ azyklisch ist, folgt nun $c(4,2) = \{2,3\}$ (und $c(2,4) = \{3,4\}$) sowie $c(3,2) = \{2,4\}$ (und $c(2,3) = \{3,4\}$). $G(\rightarrow 1)$ ist auch azyklisch, woraus folgt $c(3,1) = \{1,4\}$ (und $c(1,3) = \{3,4\}$) sowie $c(4,1) = \{1,3\}$ (und $c(1,4) = \{3,4\}$). Jetzt wird $c(4,3)$ betrachtet. Angenommen, es gilt $1 \in c(4,3)$. Wegen $4 \in c(2,4)$ haben wir den Weg $2 \rightarrow 4 \rightarrow 1 \rightarrow 3$ in $G(\rightarrow 3)$ und somit (wegen $3 \rightarrow 1 \rightarrow 4 \rightarrow 2$ in $G(\rightarrow 2)$) $1 \in c(3,2)$, einen Widerspruch. Die Annahme $2 \in c(4,3)$ kann ähnlich zum Widerspruch geführt werden, womit der Beweis erbracht ist.

Wie der obige Satz – und im Grunde schon die Charakterisierung mit Hilfe der Wege – zeigt, ist die Eigenschaft der Bidirektionalität nur sehr schwer in den Griff zu bekommen. Zum Glück ist es jedoch so, dass in einer Anwendung, bei der der zugrundeliegende Routing-Plan nicht bidirektional ist, die Protokolle der höheren Schichten immer noch die daraus resultierenden Probleme abfangen können. In der Tat ist es auch im ISDN-Netz der Deutschen Telekom so, dass durch die spezielle hierarchische Struktur des Netzes der Vermittlungsstellen die Kreisfreiheit gewährleistet ist, die Bidirektionalität jedoch nicht.

**Selbsttestaufgabe 8.6-1:**

*Für einen Graphen G mit n vielen Knoten und $\pi$ = 2 Einträgen für jede Ursprung-Ziel-Kombination soll ein Routing-Plan C mittels des Algorithmus "SP-Routing mit Präferenz" erstellt werden. Bestimmen Sie die Komplexität dieses Algorithmus (in Abhängigkeit von n).*

## 8.7 Optimales Routing

Die in den vorigen Abschnitten angesprochenen Ziele des Routing bzw. Kriterien für gutes Routing haben sich als teilweise gegenläufig herausgestellt. Will man die verschiedenen Ziele gegeneinander abwägen (z. B. Nutzung möglichst kurzer Wege gegenüber gleichmäßiger Netzbelastung), so müssen einige Größen quantifiziert werden, um in eine gemeinsame zu optimierende Zielfunktion eingehen zu können.

Als Grobziele des Routing werden oft genannt

- die Maximierung des Durchsatzes und
- die Minimierung der durchschnittlichen Paketlaufzeit.

Es liegt auf der Hand, dass diese beiden Grobziele ebenfalls gegenläufig sind – eine Verbesserung hinsichtlich eines der beiden Ziele wird stets eine Verschlechterung hinsichtlich des anderen nach sich ziehen. Es obliegt dem Zusammenspiel von Routing und Flusskontrolle, das Netz im Sinne einer Balance zwischen diesen beiden Grobzielen möglichst gut zu nutzen.

Beispielhaft soll im folgenden für ein einfaches Modell aufgezeigt werden, auf welche Art mathematischer Probleme der Versuch der Bewertung von Routing-Algorithmen führt.

Wir gehen wieder von einem ungerichteten Graphen $G$ mit den Ecken $1, ..., n$ aus. Die Belastung einer Kante $ij$ setzen wir als konstanten Wert $F_{ij}$ an (mit der Einheit Datenblöcke/sec), der sich nur aufgrund von Aktualisierungen des Routing ändert. Es wird ferner vorausgesetzt, dass sich der von den Knoten erzeugte Verkehr mit der Zeit nicht ändert.

Als zu minimierende Kostenfunktion wählen wir den Ausdruck

$$\sum_{ij} D_{ij}\ (F_{ij}),$$

wobei die Funktionen $D_{ij}$ folgendermaßen festgelegt sind:

$$D_{ij}(F_{ij}) = \frac{F_{ij}}{C_{ij} - Fij} + d_{ij}F_{ij}$$

Dabei bezeichnet $C_{ij}$ die Übertragungskapazität der Kante $ij$ (gemessen in denselben Einheiten wie $F_{ij}$) und $d_{ij}$ die Verarbeitungs- und Übertragungszeit längs der Kante $ij$.

Auf eine genaue Motivation des obigen Ausdrucks für $D_{ij}(F_{ij})$ soll an dieser Stelle nicht eingegangen werden. Man kann zeigen, dass unter gewissen Annahmen der Ausdruck $\sum_{ij} D_{ij}(F_{ij})$ gleich der durchschnittlichen im System befindlichen Anzahl von Datenblöcken ist (s. etwa [BER]). Hier soll nur festgehalten werden, dass obiges $D_{ij}(F_{ij})$ die Tatsache ausdrückt, dass die Belastung einer Kante am Rande ihrer Kapazität zu Überlast (im Englischen "Congestion") und zu hohen "Kosten" führt.

Als nächstes wird für das verwendete Modell ein Problem optimalen Routings formuliert. Es geht darum, für das vorgegebene Verkehrsaufkommen die Wege durch das Netz (und damit die $F_{ij}$) so auszuwählen, dass die Kostenfunktion minimal wird.

Für jedes Paar $w = (i, j)$ von verschiedenen Knoten gebe es ein konstantes durchschnittliches Verkehrsaufkommen $r_w$; $r_w$ (gemessen in Datenblöcken/sec) ist also die Ankunftsrate von Verkehr, der in Knoten $i$ ins Netz tritt und für $j$ bestimmt ist.

Wir führen nun folgende Bezeichnungen ein:

| | | |
|---|---|---|
| $W$ | – | Menge aller Ursprungs-Ziel-Paare |
| $P_w$ | – | Menge aller gerichteten Wege vom Ursprung zum Ziel des Paares $w$. (An dieser Stelle könnte eine Einschränkung gemacht und nicht alle Wege zugelassen werden.) |
| $x_p$ | – | Der Fluss (in Datenblöcken/sec) auf Weg $p$. |

Unser Optimierungsproblem kann nun folgendermaßen formuliert werden

Minimiere

$$\sum_{ij} D_{ij} \left( \sum_{Wege\ p\ durch\ ij} x_p \right)$$

mit den Nebenbedingungen

$$\sum_{p \in Pw} x_p = r_w \quad \text{für alle } w \in W$$

und $x_p \geq 0$ für alle p $\in P_w$ und $w \in W$.

Bevor wir zu einer Charakterisierung der optimalen Lösung dieses Optimierungsproblems kommen, wobei interessanterweise kürzeste Wege mit ins Spiel kommen, wollen wir noch einmal auf die Beschränkung der Aussagekraft hinweisen, die sich durch die Wahl unseres Modells ergibt. So ist die gewählte Kostenfunktion z. B. nicht beeinflusst von einer hohen Varianz des Verkehrsaufkommens oder von Korrelationen zwischen Paketankunftszeiten und Übertragungszeiten. Man stößt hier jedoch sehr schnell an die Grenzen des bisher analytisch Untersuchten; beispielsweise hat man keinen exakten analytischen Ausdruck für die Mittelwerte oder Varianzen der Warteschlangenlängen in einem Datennetz.

Wir benötigen den folgenden Hilfssatz, der unter sehr allgemeinen Voraussetzungen optimale Lösungen charakterisiert:

**Hilfssatz 8.7-1:** *Es sei $X$ eine konvexe Teilmenge des n-dimensionalen Raums $R^n$ über den reellen Zahlen, $f$ sei eine auf $X$ definierte differenzierbare konvexe Funktion mit Werten in $R$. Dann ist $x^* \in X$ genau dann eine optimale Lösung des Problems*

$$Minimiere\ f(x) \qquad \textit{für} \quad x \in X,$$

*wenn für alle $x \in X$ gilt*

$$\sum_{i=1}^{n} \frac{\partial f(x^*)}{\partial x_i}(x_i - x_i^*) \geq 0.$$

Dabei ist wie üblich mit $\frac{\partial f(x^*)}{\partial xi}$ die erste partielle Ableitung von $f$ nach der i-ten Komponente – ausgewertet an der Stelle $x^*$ – bezeichnet.

Der Hilfssatz gehört zum Standard einer Optimierungsvorlesung und soll hier nicht bewiesen werden. Interessierte Leser seien auf die Literatur verwiesen (z. B. auch [BER]).

Da sowohl die Kostenfunktion als auch die Definitionsmenge unseres Minimierungsproblems konvex sind, kann der Hilfssatz angewendet werden. Die Rolle der Variablen $x_i$ des Hilfsatzes wird dabei von den einzelnen Flüssen $x_p$ (mit $p \in P_w$ und $w \in W$) gespielt.

Eine Umformulierung der Charakterisierung des Hilfssatzes führt dann hier zu folgender Aussage: $x^*$ ist genau dann optimal, wenn für alle $w \in W$

$$x_p^* > 0 \qquad (p \in P_w)$$

nur dann gilt, wenn

$$\frac{\partial D(x^*)}{\partial x_{p'}} \quad \geq \quad \frac{\partial D(x^*)}{\partial x_p}$$

für alle $p' \in P_w$ ist. $D(x)$ ist dabei die oben definierte Kostenfunktion.

In Worten heißt das: eine Menge von Weg-Flüssen ist genau dann optimal, wenn der Fluss nur auf solchen Wegen positiv ist, auf denen eine Flussvergrößerung zu einem minimalen Anstieg der Kosten führt. Mit einem gewissen Recht kann man diese Wege "kürzeste Wege" nennen, und unter geeigneten Bedingungen werden diese auch den graphentheoretisch kürzesten Wegen entsprechen.

**Selbsttestaufgabe 8.7-1:**

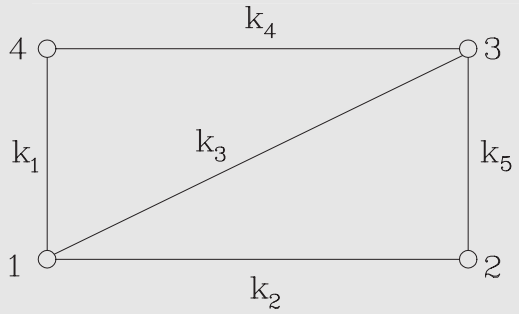
*Erläutern Sie die Schwierigkeiten, das Routing in einem Kommunikationsnetz optimal zu gestalten.*

# 9 Zuverlässigkeit von Netzen

## 9.1 Zufallsgraphen

Was ist ein Zufallsgraph?
Die Grundidee kann mit dem folgenden Beispiel verdeutlicht werden.

**Beispiel 9.1-1:**
Das im folgenden Diagramm dargestellte Netz sei gegeben:




Wir nehmen an, jede Kante $k_i$ sei mit der Wahrscheinlichkeit $p_i (0 \leq p_i \leq 1)$ verfügbar. Dies bedeutet, dass sie mit der Wahrscheinlichkeit $(1 - p_i)$ ausfällt. Die vier Stationen (Knoten) seien ausfallsicher. Sind alle möglichen Wege als Kommunikationswege zugelassen, so ist die Wahrscheinlichkeit $P\ (1 \leftrightarrow 4)$, dass 1 mit 4 (und auch 4 mit 1) kommunizieren kann, offenbar gegeben durch die Wahrscheinlichkeit, dass
- Kante $k_1$ intakt ist oder
- $k_3$ und $k_4$ intakt sind oder
- $k_2$ und $k_4$ und $k_5$ intakt sind.

Bezeichnen wir das Ereignis "$k_i$ ist intakt" kurz mit "$k_i$", und verwenden wir wie üblich das Zeichen $\vee$für "oder" und $\wedge$ für "und", so haben wir also $P\ (1 \leftrightarrow 4) \quad = \quad P\ \{k_1 \ \vee \ (k_3 \wedge \ k_4) \ \vee \ (k_2 \ \wedge \ k_4 \ \wedge \ k_5)\}$. Dabei ist mit $p\{A\}$ die Wahrscheinlichkeit des Ereignisses A bezeichnet.
Um nun die Zahlen $p_1, \ ..., \ p_5$ einsetzen zu können, muss der Ausdruck $k_1 \ \vee \ (k_3 \ \wedge \ k_4) \ \vee \ (k_2 \ \wedge \ k_4 \ \wedge \ k_5)$ so umgeformt werden, dass sich eine ODER-Verknüpfung sich gegenseitig ausschließender Ereignisse ergibt. Wir verweisen hier auf den Anhang, in dem Notwendigkeit und Möglichkeit solcher Umformungen dargelegt sind. Eine logisch äquivalente Form ist z. B. $k_1 \ \vee \ (k_1' \ \wedge \ k_3 \ \wedge \ k_4) \quad \vee \quad (k_1' \ \wedge \ k_2 \ \wedge k_3' \ \wedge \ k_4 \ \wedge \ k_5)$, wobei $k_i'$ die Negation von $k_i$ bezeichnet. Damit folgt $P\ (1 \leftrightarrow 4) \ = \ p_1 + (1 - p_1)\ p_3\ p_4 + (1 - p_1)\ p_2\ (1 - p_3)\ p_4\ p_5$. Sind insbesondere alle $p_i$ gleich einem $p$, so hat man $P\ (1 \leftrightarrow 4) \ = \ p \ + \ p^2 \ - \ 2p^4 \ + \ p^5$. Für $p \ = \ 0,99$ ergibt sich z. B. $P\ (1 \leftrightarrow 4) \ \approx 0,999898$.

Zu diesem Beispiel sind einige Bemerkungen angebracht. Es sind nur Ausfälle von Kanten betrachtet und die Stationen (Knoten) als ausfallsicher angenommen worden. Wir werden auch im weiteren meist von dieser Annahme ausgehen. Dies hat seine Rechtfertigung in folgender Überlegung: In der Regel wird unser Zuverlässigkeitsmaß auf funktionsfähigen Zuständen basieren, bei denen je zwei Stationen miteinander kommunizieren können. Letzteres kann aber ohnehin nur der Fall sein, wenn alle Knoten intakt sind. Dies bedeutet: die Zuverlässigkeit der einzelnen Knoten geht einfach als zusätzlicher multiplikativer Faktor in die zu untersuchende Zuverlässigkeit ein und hat damit für die weitere Analyse keine Bedeutung.
Ein Rechenschritt des Beispiels bestand darin, einen Booleschen Ausdruck in eine Disjunktion von Konjunktionen von (eventuell komplementierten) Variablen so umzuwandeln, dass die einzelnen Terme zueinander disjunkt sind (d. h. sich ausschließen), um daraus dann die Wahrscheinlichkeit bestimmen zu können. Über das hier vorliegende Problem, welches (algorithmisch) nicht einfach zu lösen ist, kann man sich im Anhang informieren.

Es muss auch herausgestellt werden, dass die in dem obigen Beispiel gefragte Wahrscheinlichkeit $P\ (1 \leftrightarrow 4)$ natürlich eine andere sein kann, wenn nicht alle physikalisch möglichen Wege erlaubt sind. Wie wir aus dem Abschnitt über das Routing im Zeichengabesystem Nr.7 wissen, gibt es Kommunikationsnetze mit dieser Eigenschaft. Die Zuverlässigkeit ist somit außer von der Netzstruktur auch von dem gewählten Routing abhängig.

**Beispiel 9.1-2:**
Für das Netz aus Definition 1.1-1soll nun der folgende Routing-Plan verwendet werden:

| von/nach | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | - | 2;3 | 3;2 | 4;3 |
| 2 | 1;3 | - | 3;1 | 1;3 |
| 3 | 1;2 | 2;1 | - | 4;1 |
| 4 | 1;3 | 1;3 | 3;1 | - |

Wir wollen die Wahrscheinlichkeit $P\ (1 \rightarrow 4)$ bestimmen, dass Station 1 Nachrichten an 4 (erfolgreich) schicken kann. Da nun Station 2 für Nachrichten zu Ziel 4 für Station 1 gar nicht in Frage kommt, ergibt sich hier

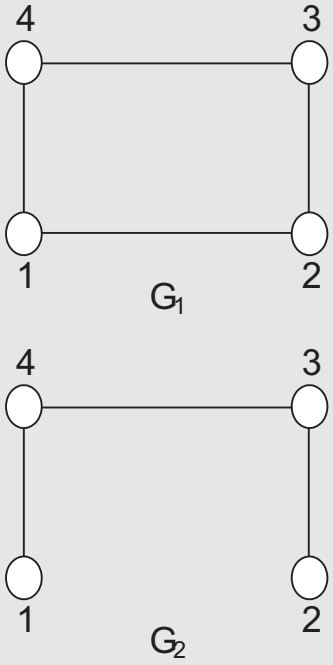
$$\begin{aligned} P\ (1 \rightarrow 4) &= P\{k_1\ \vee\ (k_3\ \wedge\ k_4)\} \\ &= P\{k_1\ \vee\ (k_1'\ \wedge\ k_3\ \wedge\ k_4)\} \\ &= p_1 + (1 - p_1)p_3p_4 \end{aligned}$$

.

Nimmt man wieder ein konstantes $p$ für alle $p_i$, so hat man $P\ (1 \rightarrow 4) = p\ +\ p^2\ -\ p^3$ und mit $p = 0,99\ P\ (1 \rightarrow 4) = 0,999801$. Dies ist, wie man sieht, etwas schlechter als der Wert in 1.1.1. (Dort war selbstverständlich $P\ (1 \leftrightarrow 4)\ =\ P\ (4 \rightarrow 1)$.)

Was ist also ein Zufallsgraph $G$?
Man kann sich das intuitiv folgendermaßen vorstellen: Für alle Eckenpaare $k = (e, f) \in E \times E$ wird unabhängig entschieden, ob $k$ eine Kante von $G$ sein soll oder nicht; die Wahrscheinlichkeit, dass $k$ eine Kante sei, soll dabei jeweils gleich einem festen Wert $p$ sein ($0 \leq p \leq 1$). Mit anderen Worten machen wir im folgenden die Annahme, dass nur die Kanten "dem Zufall unterliegen" und jede Kante mit der gleichen Wahrscheinlichkeit $p$ vorhanden ist. Mit dieser Festlegung wird die Menge $\mathfrak{G}(n, p)$ aller Graphen auf der Eckenmenge $E = \{1, 2, ..., n\}$ zu einem Wahrscheinlichkeitsraum, wobei die Elementarereignisse den einzelnen Graphen $G = (E, K)$ entsprechen. Genauer gesagt verhält es sich so:
Ist $G_1 = (E, K)$ ein konkreter Graph auf der Eckenmenge $E$, und enthält $K$ $m$ viele Kanten, so hat das Elementarereignis $\{G_1\}$ die Wahrscheinlichkeit $p^m \cdot q^{\binom{n}{2}-m}$, d. h. ein Zufallsgraph ist mit dieser Wahrscheinlichkeit gerade genau dieser Graph $G_1$. (Wie üblich ist $1 - p$ durch $q$ abgekürzt.)
Zur Begründung dieser Formel muss man sich nur klar machen, dass die $m$ vielen Kanten aus $K$ vorhanden sein müssen - dies führt zum Faktor $p^m$ - und die restlichen möglichen $\binom{n}{2} - m$ vielen Kanten sämtlich nicht vorhanden sein sollen – dies führt zu $q^{\binom{n}{2}-m}$.

**Beispiel 9.1-3:**

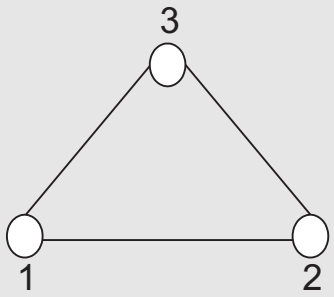



Für $n = 4$ hat der Graph $G_1$ die Wahrscheinlichkeit $p^4q^2$, $G_2$ besitzt die Wahrscheinlichkeit $p^3q^3$.

Auf einen vollständigen Beweis, dass man in der obigen Weise in der Tat einen Wahrscheinlichkeitsraum bekommen hat, wollen wir hier verzichten. (Man betrachtet dazu zu jeder möglichen Kante den "kleinen" Wahrscheinlichkeits-

raum mit nur zwei Elementarereignissen - Kante existiert oder nicht - und dann den Produktraum all dieser Wahrscheinlichkeitsräume. Dieser kann mit dem uns interessierenden Raum identifiziert werden.) Wenn im folgenden von der Wahrscheinlichkeit die Rede ist, dass ein beliebiger Graph $G \in \mathfrak{G}(n,p)$ eine gewisse Eigenschaft (*) hat (z. B. dass er zusammenhängend ist), so ist dies immer eine abkürzende Sprechweise für das Wahrscheinlichkeitsmaß der Menge $\{G \in \mathfrak{G}(n,p) \mid G \text{ hat die Eigenschaft (*)}\}$ . Man kann sich (siehe folgendes Beispiel) leicht klar machen, dass diese formale Festlegung mit der intuitiven Vorstellung übereinstimmt.

**Beispiel 9.1-4:**




Sei $G \in \mathfrak{G}(3,p)$. Wie groß ist die Wahrscheinlichkeit, dass zwischen den Ecken 1 und 2 ein Weg existiert?
Da entweder die Kante 12 existieren muss, oder aber (falls nicht) 13 und 32, ergibt sich als Wahrscheinlichkeit $p + qp^2$.

In den obigen Beispielen sind Zufallsgraphen dazu verwendet worden, um Kommunikationsnetze mit ausfallgefährdeten Kanten zu modellieren. Dies wird auch im nächsten Abschnitt im Mittelpunkt der Betrachtungen stehen. Im Rest des vorliegenden Abschnitts soll beispielhaft aufgezeigt werden, welch wichtige Rolle die Zufallsgraphen heute innerhalb der Graphentheorie spielen. Die Theorie der Zufallsgraphen hat sich in den letzten Jahrzehnten zu einer eigenen Disziplin entwickelt.

Sehr bekannt ist der folgende Satz von Erdös.

**Satz 9.1-1:** *Es gibt Graphen, die zugleich eine große Taillenweite und eine hohe chromatische Zahl haben. Genauer gesagt: Zu jeder natürlichen Zahl $k$ existiert ein Graph $G$ mit $g(G) > k$ und $\chi(G) > k$*

**Taillenweite $g(G)$**
**Chromatische Zahl $\chi(G)$**

Dabei wird unter der **Taillenweite** $g(G)$ die Länge eines kürzesten Kreises in $G$ verstanden. Die **chromatische Zahl** $\chi(G)$ ist die kleinste natürliche Zahl $k$ von Farben, mit denen die Ecken von $G$ "gefärbt" werden können derart, dass benachbarte Ecken stets unterschiedliche Farben erhalten. Wenn man über diese Begriffsbildungen nachdenkt, scheint es auf den ersten Blick so, dass Graphen mit "wenigen" Kanten (relativ zur Eckenanzahl $n$) zu großer Taillenweite tendieren; ist der

Graph lediglich ein Kreis mit $n$ vielen Kanten, so ist offenbar $g(G) = n$. Auf der anderen Seite scheinen viele Kanten für eine hohe chromatische Zahl zu sprechen - man denke an den vollständigen Graphen $K_n$, der offensichtlich die größtmögliche chromatische Zahl $n$ besitzt. So gesehen, liegt die Annahme nahe, dass sich "große Taillenweite" und "hohe chromatische Zahl" widersprechen. Der Satz von Erdös besagt jedoch, dass dies nicht der Fall ist.
Wie beweist man einen solchen Satz, der der Intuition zu widersprechen scheint? Es erscheint sehr schwierig, einen Graphen mit $g(G) > k$ und $\chi(G) > k$ zu *konstruieren*; bei einer solchen konstruktiven Vorgehenswiese stützt man sich meist auf kleinere Bausteine, die zu dem gesuchten Objekt zusammen gesetzt werden - man hat aber hier keine Idee, wie diese Bausteine aussehen könnten. Nun kommen wir zu dem Punkt, auf den wir hier hinaus wollen: Zum Beweis dieses Satzes hat Erdös im Jahre 1959 die "probabilistische Methode" eingeführt, indem er (kurz gesagt) zeigt, dass für genügend großes $n$ mit geeignetem $p$ die Wahrscheinlichkeit, dass ein Zufallsgraph $G \in \mathfrak{G}(n,p)$ sowohl $g(G) > k$ als auch $\chi(G) > k$ erfüllt, größer als Null ist!

Ein ganz eigener Typ graphentheoretischer Sätze ergibt sich, wenn man *Eigenschaften fast aller Graphen* betrachtet. Wir wollen einen Satz von diesem Typ komplett beweisen, um so beispielhaft in die Theorie der Zufallsgraphen einzusteigen. Dabei gehen wir von folgender Begriffsbildung aus: Eine Grapheneigenschaft wird durch eine Klasse von Graphen modelliert, die mit jedem Graphen auch alle dazu isomorphen Graphen enthält; dadurch wird es möglich, von der Wahrscheinlichkeit zu sprechen, dass ein Graph die betreffende Eigenschaft hat.

Ist $p$ gegeben, so kann man für eine Grapheneigenschaft $\mathfrak{E}$ untersuchen, wie sich die Wahrscheinlichkeit der Menge aller $G \in \mathfrak{G}(n,p)$ mit $G \in \mathfrak{E}$ verhält, wenn $n$ immer größer wird. Strebt diese Wahrscheinlichkeit gegen 1, so sagt man, dass *fast alle* $G \in \mathfrak{G}(n,p)$ die Eigenschaft $\mathfrak{E}$ haben; geht sie gegen 0, so hat *fast kein* $G \in \mathfrak{G}(n,p)$ die Eigenschaft $\mathfrak{E}$.
Der Vorbereitung des Satzes dient:

**Lemma 9.1-1:** *Für $p \notin \{0,1\}$ ist jeder Graph $H$ ein Untergraph fast aller Graphen $G \in \mathfrak{G}(n,p)$.*
*(Präziser formuliert: Fast alle Graphen $G \in \mathfrak{G}(n,p)$ enthalten einen zu $H$ isomorphen Untergraphen.)*

**Beweis:** Ein beliebiger Graph $H$ mit $k$ vielen Ecken sei gegeben. Ist $U \subseteq \{1,2,...,n\}$ irgendeine Menge von $k$ vielen Ecken von $G \in \mathfrak{G}(n,p)$, so ist sicherlich der von $U$ innerhalb $G$ aufgespannte Untergraph $G[U]$ mit einer Wahrscheinlichkeit $r > 0$ isomorph zu $H$. Diese Wahrscheinlichkeit $r$ hängt selbstverständlich von $p$ ab, nicht jedoch von $n$. Nun denken wir uns die Eckenmenge $\{1,2,...,n\}$ von $G$ in $\lfloor n/k \rfloor$ disjunkte solche Teilmengen $U$ unterteilt. Die Wahrscheinlichkeit, dass nun *keiner* der Teilgraphen der Form $G[U]$ isomorph zu $H$ ist, beträgt offenbar $(1-r)^{\lfloor n/k \rfloor}$. Insgesamt ergibt sich damit $P(H \not\subseteq G) \leq (1-r)^{\lfloor n/k \rfloor} \longrightarrow 0$ (für $n \longrightarrow \infty$), was zu beweisen war.

Es folgt ein zweites Lemma, mit dessen Hilfe für eine Reihe von Eigenschaften gezeigt werden kann, dass fast jeder Graph sie hat. Dazu bezeichnen wir mit $P_{ij}$ für $i, j \in \mathbb{N}$ die Eigenschaft, dass der Graph mindestens $i + j$ viele Ecken hat und zu je zwei disjunkten Eckenmengen $U, W$ mit $|U| \leqslant i$ und $|W| \leqslant j$ stets eine Ecke $v \notin U \cup W$ enthält, die zu allen Ecken aus $U$, jedoch zu keiner aus $W$ benachbart ist.

**Lemma 9.1-2:** *Für gegebene $p \notin \{0, 1\}$ und $i, j \in \mathbb{N}$ hat fast jeder Graph $G \in \mathfrak{G}(n, p)$ die Eigenschaft $P_{ij}$.*

**Beweis:** In einem Zufallsgraphen $G \in \mathfrak{G}(n, p)$ seien $U$ und $W$ mit $|U| \leqslant i$ und $|W| \leqslant j$ gegeben. Die Wahrscheinlichkeit, dass nun eine Ecke $v \notin U \cup W$ zu allen Ecken aus $U$, jedoch zu keiner aus $W$ benachbart ist, beträgt offenbar $p^{|U|}q^{|W|} \geqslant p^i q^j$. Folglich gilt für die Wahrscheinlichkeit, dass für diese Mengen $U$ und $W$ kein solches $v$ existiert, die Ungleichung $(1 - p^{|U|}q^{|W|})^{n-|U|-|W|} \leqslant (1 - p^i q^j)^{n-i-j}$. Da es auf jeden Fall nicht mehr als $n^{i+j}$ viele solcher Mengen U und W in G gibt, beträgt die Wahrscheinlichkeit, dass sich darunter ein Paar ohne ein geeignetes v befindet, höchstens $n^{i+j}(1-p^i q^j)^{n-i-j}$, und dieser Wert konvergiert für $n \to \infty$ gegen Null (weil $1 - p^i q^j < 1$ ist).

Daraus ergibt sich nun die folgende interessante Folgerung.

**Folgerung 9.1-1:** *Für gegebenes $p \notin \{0, 1\}$ und $k \in \mathbb{N}$ ist fast jeder Graph in $\mathfrak{G}(n, p)$ k-zusammenhängend.*

**Beweis:** Aufgrund von Lemma 9.1-2 ist die Folgerung klar, wenn wir uns überlegt haben, dass jeder Graph mit der Eigenschaft $P_{2,k-1}$ *k*-zusammenhängend ist. Ist aber *F* eine Menge von weniger als *k* vielen Ecken, so haben je zwei weitere Ecken $x, y$ einen gemeinsamen Nachbarn $v \notin F$, also kann *F* die Ecken *x* und *y* nicht trennen.

Ebenfalls interessant – und auch etwas überraschend – ist:

**Folgerung 9.1-2:** *Für gegebenes $p \notin \{0, 1\}$ hat fast jeder Graph in $\mathfrak{G}(n, p)$ den Durchmesser 2.*

**Beweis:** Aus Lemma 9.1-2 folgt, dass es (mit $i = 4$ und $j = 0$) zu jedem Weg von Länge 3 eine Abkürzung von Länge 2 gibt.

Hiermit wollen wir die Einführung in die Theorie der Zufallsgraphen beenden.

## 9.2 Der Zusammenhang von Zufallsgraphen

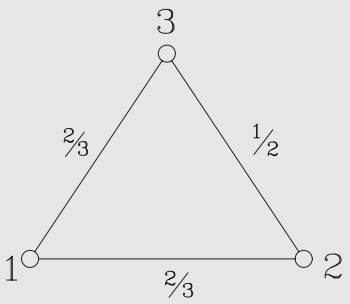
Das folgende, für Zuverlässigkeitsbetrachtungen offenbar relevante Problem wurde bereits im vorrigen Abschnitt angesprochen:

Gegeben seien mit 1, 2, ..., n bezeichnete Ecken eines Graphen. Für jede der $\binom{n}{2}$ vielen möglichen ungerichteten Kanten $ij$ sei $p_{ij}$ ($0 \leq p_{ij} \leq 1$) die Wahrscheinlichkeit, dass die Kante $ij$ existiert. Wir sprechen von einem **Zufallsgraphen** und

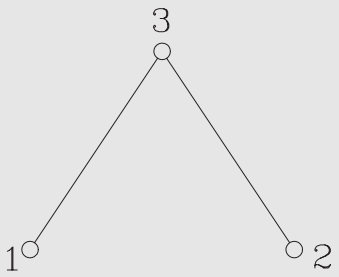
**Zufallsgraph**

fragen nach der Wahrscheinlichkeit, dass der Graph eine gewisse Eigenschaft hat (z. B. dass er zusammenhängend ist).

**Beispiel 9.2-1:**
Es sei $n = 3$. Die Wahrscheinlichkeiten $p_{ij}$ seien wie im folgenden Diagramm gegeben:




Als $G$ sei nun der Graph mit den Kanten 13 und 23 definiert:




Man erhält

$$\begin{aligned} P\{G\} &= p_{13} \cdot p_{23} \cdot (1 - p_{12}) \\ &= \tfrac{1}{9}\,. \end{aligned}$$

Dies hat folgende Interpretation: Sind 1, 2, 3 die Stationen eines Datennetzes, und stellt $p_{ij}$ die Wahrscheinlichkeit dar, dass zu einem beliebigen Zeitpunkt die Kante $ij$ intakt ist, so hat das Netz zu irgendeinem Zeitpunkt mit Wahrscheinlichkeit 1/9 die durch $G$ gegebene Struktur.

Wir fassen jeden einzelnen Graphen $G$ als Ereignis auf, welches mit einer gewissen Wahrscheinlichkeit $P\{G\}$ auftritt. Strenggenommen befinden wir uns hier in dem Produkt von $\binom{n}{2}$ vielen den Kanten zugeordneten Wahrscheinlichkeitsräumen (siehe etwa [PAP]).

Mi $\mathcal{G}_n$ bezeichnen wir alle Graphen mit der Eckenmenge $\{1, ..., n\}$. Für eine beliebige Teilmenge $\tau \subseteq \mathcal{G}_n$ sei $P\{\tau\}$ die Wahrscheinlichkeit, dass einer der Graphen aus $\tau$ auftritt; m. a. W. stellt $P\{\tau\}$ die Wahrscheinlichkeit der ODER-Verknüpfung

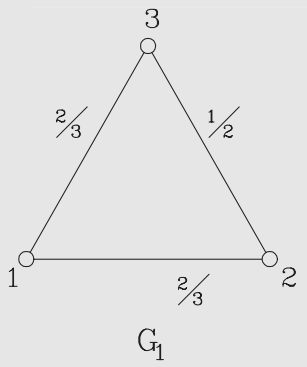
der einzelnen Graphen (Ereignisse) aus $\tau$ dar. Da sich diese verschiedenen Graphen alle gegenseitig ausschließen, gilt hier

$$P\{\tau\} = \sum_{G \in \tau} P\{G\}.$$

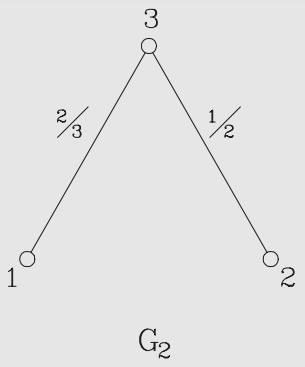
$P\{\tau\}$ ist die Wahrscheinlichkeit, dass ein zufällig herausgegriffener Graph mit Eckenmenge $\{1, ..., n\}$ zu der Menge $\tau$ gehört.

$\tau_n \subseteq \mathcal{G}_n$ sei die Menge der zusammenhängenden Graphen mit Eckenmenge $\{1, ..., n\}$. $P\{\tau_n\}$ ist also die Wahscheinlichkeit, dass ein beliebiger Graph $G \in \mathcal{G}_n$ zusammenhängend ist. Diese Wahrscheinlichkeit hängt natürlich von den einzelnen $p_{ij}$ ab, was wir jedoch der Einfachheit halber nicht in die Bezeichnung $P\{\tau_n\}$ mit aufgenommen haben.

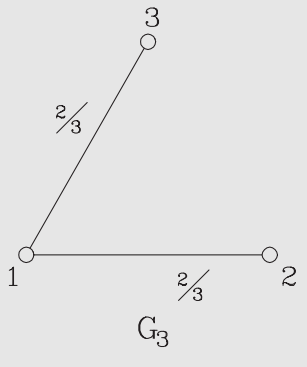
**Beispiel 9.2-2:**
Wir gehen wieder von $n = 3$ und den in Beispiel 9.2-1 verwendeten Kantenwahrscheinlichkeiten aus und wollen $P\{\tau_3\}$ bestimmen.
Ein Graph $G$ mit Ecken $1, 2, 3$ ist offenbar genau dann zusammenhängend, wenn er mindestens zwei Kanten hat. Die vier Möglichkeiten sind hier dargestellt:



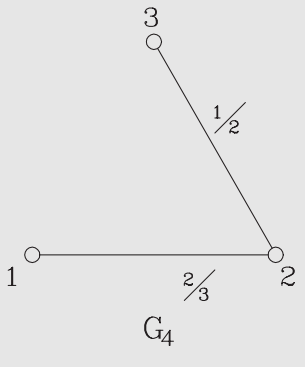
$G_1$



$G_2$



$G_3$



$G_4$

Man erhält

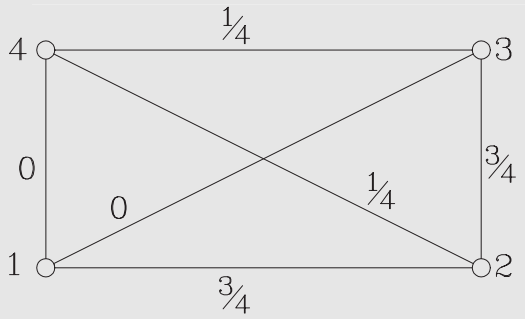
$$\begin{aligned} P\{\tau_3\} &= \sum_{i=1}^{4} P\{G_i\} \\ &= \tfrac{2}{3}\cdot\tfrac{2}{3}\cdot\tfrac{1}{2} + \tfrac{1}{3}\cdot\tfrac{2}{3}\cdot\tfrac{1}{2} + \tfrac{2}{3}\cdot\tfrac{2}{3}\cdot\tfrac{1}{2} + \tfrac{2}{3}\cdot\tfrac{1}{3}\cdot\tfrac{1}{2} \\ &= \tfrac{2}{3}. \end{aligned}$$

Ein interessanter Sonderfall liegt vor, wenn einige der möglichen Kanten $ij$ mit Sicherheit nicht auftreten (und wenn somit $p_{ij} = 0$ für diese Kanten gilt). Für einen Graphen $G$ auf der Eckenmenge $E = \{1, ..., n\}$ mit Kantenmenge $K$ kann sicher nur dann $P\{G\} \neq 0$ gelten, wenn $P(k) \neq 0$ für jedes $k \in K$ ist. Bezeichnet man die Menge aller Kanten $k = ij$ mit $P(ij) \neq 0$ mit $\bar{K}$, so kann man sagen:

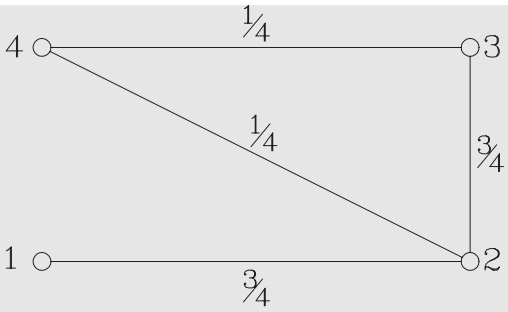
Für $G = (E, K)$ gilt $P\{G\} \neq 0$ genau dann, wenn $G$ ein Teilgraph von $\bar{G} = (E, \bar{K})$ ist.

Durch Betrachtung des Graphen $\bar{G}$ (statt $K_n$) hat man die Wahrscheinlichkeitsbetrachtungen quasi auf $\bar{G}$ "relativiert". In der Literatur stößt man oft auf diesen Ansatz, den auch wir im nächsten Abschnitt zugrundelegen werden.

**Beispiel 9.2-3:**
Die im folgenden Diagramm eingetragenen Kantenwahrscheinlichkeiten für $E = \{1, 2, 3, 4\}$ seien gegeben:

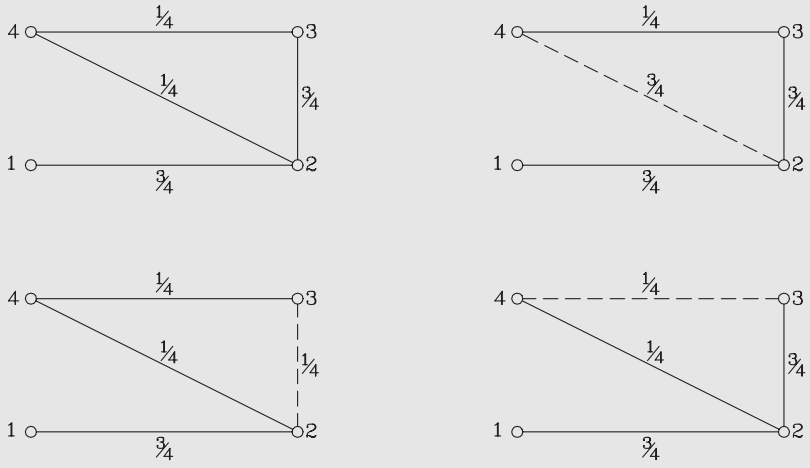



Für $\bar{G} = (E, \bar{K})$ hat man dann:




$P\{G\} \neq 0$ gilt offenbar nur für Graphen $G$, die Teilgraphen von $\bar{G}$ sind.

Die zusammenhängenden Teilgraphen von $\bar{G}$ sind in folgenden Diagrammen dargestellt:



Dies führt zu der Zusammenhangswahrscheinlichkeit

$$\begin{aligned} P\{\tau_4\} &= \tfrac{1}{4}\cdot\tfrac{1}{4}\cdot\tfrac{3}{4}\cdot\tfrac{3}{4}\cdot + \tfrac{1}{4}\cdot\tfrac{3}{4}\cdot\tfrac{3}{4}\cdot\tfrac{3}{4}\cdot + \tfrac{1}{4}\cdot\tfrac{1}{4}\cdot\tfrac{3}{4}\cdot\tfrac{1}{4}\cdot + \tfrac{3}{4}\cdot\tfrac{1}{4}\cdot\tfrac{3}{4}\cdot\tfrac{3}{4}. \\ &= \tfrac{33}{128}. \end{aligned}$$

Es ist auch üblich, dies so auszudrücken: "$\frac{33}{128}$ ist die Wahrscheinlichkeit, dass der Graph $\bar{G}$ zusammenhängend ist."

Als nächstes soll das Problem der Bestimmung von $P\{\tau_n\}$ allgemein untersucht werden für den Fall, dass alle Wahrscheinlichkeiten $p_{ij}$ gleich einem konstanten $p(0 \leq p \leq 1)$ sind.Gefragt wird also nach der Wahrscheinlichkeit, dass ein beliebig ausgewählter Graph auf $n$ Ecken zusammenhängend ist. In der Terminologie von Abschnitt 9.1 geht es also nun um die Frage, mit welcher Wahrscheinlichkeit ein Graph $G \in \mathfrak{G}(n,p)$ zusammenhängend ist.

Wir setzen abkürzend $f_n := P\{\tau_n\}$. $f_n = f_n(p)$ ist eine Funktion (genauer: ein Polynom) der Wahrscheinlichkeit $p$ und stellt die Wahrscheinlichkeit dar, dass der vollständige Graph $K_n$ zusammenhängend ist. Ein zusammenhängender spannen-

der Teilgraph von $K_n$ muss mindestens $n-1$ viele Kanten enthalten, insgesamt hat $K_n$ $\binom{n}{2}$ viele Kanten.

Mit $g(n,b)$ sei für $n-1 \leq b \leq \binom{n}{2}$ die **Anzahl aller zusammenhängenden Graphen** auf der Eckenmenge $\{1,...,n\}$ mit $b$ vielen Kanten bezeichnet.

**Anzahl aller zusammenhängenden Graphen**

**Satz 9.2-1:** *Mit den obigen Bezeichnungen gilt*

$$f_n = \sum_{b=n-1}^{\binom{n}{2}} g(n,b)\ p^b\ (1-p)^{\binom{n}{2}-b}.$$

**Beweis:** Dies folgt unmittelbar daraus, dass jeder einzelne dieser Graphen mit $b$ vielen Kanten eine Auftrittswahrscheinlichkeit von

$$p^b\ (1-p)^{\binom{n}{2}-b}$$

hat.

Leider ist die obige Formel zur Berechnung nicht sehr praktikabel, da keine guten Verfahren (im algorithmischen Sinne) auf der Hand liegen, die Koeffizienten $g(n,b)$ zu berechnen. Wir werden auf dieses Problem an späterer Stelle zurückkommen.

Wir können jedoch eine rekursive Formel für $f_n$ $(n = 1,2,...)$ angeben, die ohne die Zahlen $g(n,b)$ auskommt:

**Satz 9.2-2:**

Es gilt $\quad f_1 = 1$

und für $n \geq 1 \quad f_n = 1 - \sum_{k=1}^{n-1} \binom{n-1}{k-1}\ f_k\ (1-p)^{k(n-k)}.$

**Beweis:** Die Aussage über $f_1$ ist trivial.

Nun sei $n \geq 2$ gegeben. Um die Rekursion zu zeigen, betrachten wir für $1 \leq k \leq n$ die Wahrscheinlichkeit $z_k$, dass die den Punkt 1 enthaltende Zusammenhangskomponente aus genau $k$ vielen Ecken besteht. Wir behaupten, dass

$$z_k = \binom{n-1}{k-1}\ f_k\ (1-p)^{\ k(n-k)}$$

gilt. Da außerdem

$$\sum_{k=1}^{n} z_k = 1$$

gilt, folgt damit dann

$$f_n = z_n = 1 - \sum_{k=1}^{n-1} z_k$$

und durch Einsetzen von $z_k$ die zu zeigende Behauptung. Die Gültigkeit der obigen Formel für $z_k$ wird mit folgender überlegung klar: Es gibt $\binom{n-1}{k-1}$ viele

Möglichkeiten, aus den $n-1$ von 1 verschiedenen Ecken $k-1$ auszuwählen, die mit 1 eine $k-$ elementige Zusammenhangskomponente bilden können. dass diese $k$ Ecken eine Zusammenhangskomponente bilden, hat die Wahrscheinlichkeit $f_k\ (1-p)^{k(n-k)}$, denn diese $k$ Ecken müssen erstens einen zusammenhängenden Graphen bilden (die führt zu dem Term $f_k$) und dürfen zweitens keine zu einer weiteren Ecke führende Kante haben (dies "verbietet" $k(n-k)$ viele mögliche Kanten).

Damit ist der Satz bewiesen.

**Beispiel 9.2-4:**
Man rechnet leicht aus, dass sich bis zu $n=4$ die folgenden Polynome in $p$ für $f_n$ ergeben:

$$\begin{array}{lcll} f_2 & = & p & \\ f_3 & = & 1-(1-p)^2-2p(1-p)^2 & = & 3p^2-2p^3 \\ f_4 & = & 1-(1-p)^3-3p(1-p)^4 & - & 3(3p^2-2p^3)(1-p)^3 \\ & = & 16p^3-33p^4+24p^5-6p^6 & & \end{array}$$

Für $p=0,99$ hat man also:

$$\begin{array}{lcl} f_2 & = & 0,99 \\ f_3 & = & 0,999702 \\ f_4 & \approx & 0,999996 \end{array}$$

An dieser Stelle muss eine Bemerkung zur algorithmischen Komplexität der Berechnung von $f_n$ gemacht werden. Die in dem obigen Satz angegebene rekursive Formel liefert offenbar ein polynomiales Verfahren (gemessen in der Größe von $n$) zur Bestimmung von $f_n$. Ist das Polynom $f_n(p)$ auf diese Weise bestimmt, können über ein lineares Gleichungssystem auch die Koeffizienten $g(n,b)$ aus Satz 9.2-1 errechnet werden.

**Beispiel 9.2-5:**
Es sollen die Koeffizienten $g(4,b)$ für die Darstellung

$$f_4 = g(4,3)p^3(1-p)^3 + g(4,4)p^4(1-p)^2 + g(4,5)p^5(1-p) + g(4,6)p^6$$

berechnet werden. Aus Beispiel 9.2-4 wissen wir, dass gilt:

$$f_4 = 16p^3 - 33p^4 + 24p^5 - 6p^6.$$

Zur Bestimmung der vier Unbekannten setzen wir hier die Werte $p = 1, -1, 2$ und $-2$ ein und erhalten:

$$f_4(1) = 1, \quad f_4(-1) = -79, \quad f_4(2) = -16, \quad f_4(-2) = -1808.$$

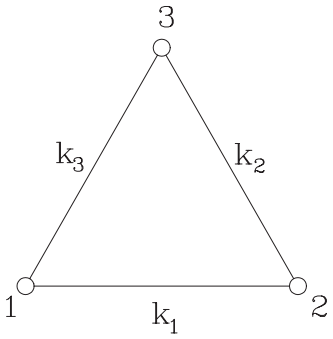
Einsetzen der so gewonnenen Wertepaare in die obere Gleichung liefert nun das Gleichungssystem

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | = | | | | | | | g(4,6) |
| -79 | = | -8g(4,3) | + | 4g(4,4) | - | 2g(4,5) | + | g(4,6) |
| -16 | = | -8g(4,3) | + | 16g(4,4) | - | 32g(4,5) | + | 64g(4,6) |
| -1808 | = | -216g(4,3) | + | 144g(4,4) | - | 96g(4,5) | + | 64g(4,6) |

Dessen eindeutige Lösung erhält man schließlich mit den üblichen Methoden (z. B. dem Gaußschen Algorithmus):

$$g(4,3) = 16\,, \quad g(4,4) = 15\,, \quad g(4,5) = 6\,, \quad g(4,6) = 1\,.$$

Nach diesem ersten Einstieg in die Eigenschaften von Zufallsgraphen wollen wir dieses Thema vorübergehend verlassen. Wir werden es bald wieder aufgreifen, um dann Verfahren zur Berechnung wie auch Abschätzungen für die Netzzuverlässigkeit zu erhalten. Der Vollständigkeit halber soll noch aufgezeigt werden, wie z. B. das Polynom $f_3 = 3p^2 - 2p^3$ durch Untersuchung Boolescher Ausdrücke gewonnen werden kann:




Wie im vorigen Abschnitt stehe die Boolesche Variable "$k_i$" für die Aussage "Kante $k_i$ ist intakt". Es ist dann

$$f_3 = P\{(k_1 \wedge k_2) \vee (k_1 \wedge k_3) \vee (k_2 \wedge k_3)\}.$$

Die zugehörige kanonische DNF (siehe Anhang) lautet

$$(k_1 \wedge k_2 \wedge k_3) \vee (k_1' \wedge k_2 \wedge k_3) \vee (k_1 \wedge k_2' \wedge k_3) \vee (k_1 \wedge k_2 \wedge k_3'),$$

folglich gilt

$$f_3 = p^3 + 3p^2\,(1-p) = 3p^2 - 2p^3.$$

**Selbsttestaufgabe 9.2-1:**

*Man bestimme den Koeffizienten* $g(5,4)$.

## 9.3 Zuverlässigkeitsmaße und -polynome

Wir gehen in diesem Abschnitt stets von einem ungerichteten Graphen $G = (E, K)$ auf der Eckenmenge $E = \{1, 2, ..., n\}$ aus, der nicht als vollständig (bzw. vollvermascht) vorausgesetzt wird.
$G$ soll die Grundstruktur eines Kommunikationsnetzes darstellen.

Es wird angenommen, dass nur Kanten (und keine Ecken) ausfallen können. Für jede Kante $k \in K$ ist dies zu einem beliebigen Zeitpunkt mit einer festen Wahrscheinlichkeit $1 - p(k)$ der Fall, und die Ausfälle verschiedener Kanten seien voneinander unabhängig.

Bei den folgenden Zuverlässigkeitsuntersuchungen wird davon ausgegangen, dass die Funktionsfähigkeit des Netzes nur von der aktuellen Topologie (d. h. welche Kanten gerade intakt sind und welche nicht) abhängt. Als Zuverlässigkeit wird also die Wahrscheinlichkeit definiert, dass sich der Graph in einem funktionsfähigen Zustand befindet.

Formaler kommt man zu folgendem Begriffsaufbau:

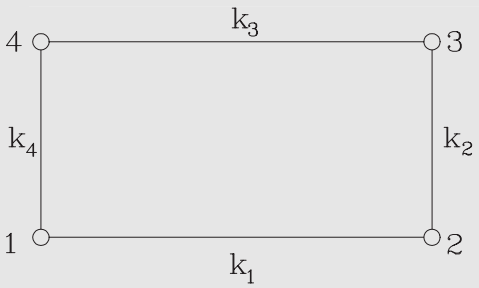
**Zustand**
**funktionsfähiger Zustand**

Eine beliebige Kantenmenge $S \subseteq K$ heißt ein **Zustand**. Nun sei eine Menge von Zuständen $\mathcal{F}(G) \subseteq \mathcal{P}(K)$ ausgezeichnet, die genau die **funktionsfähigen Zustände** enthält.

**Zuverlässigkeit**

Als **Zuverlässigkeit** $z(G)$ bezeichnen wir die Wahrscheinlichkeit, dass sich der Graph in einem funktionsfähigen Zustand befindet; $z(G)$ ist natürlich von $\mathcal{F}(G)$ abhängig.

**Beispiel 9.3-1:**
Der im folgenden Diagramm dargestellte Graph sei gegeben:




Es sei $p(k_i) = p = 0,99$ für jede Kante $k_i$.

Im ersten Fall sei $\mathcal{F}_1(G) = \{\{k_1, k_2, k_3, k_4\}\}$. Das Netz ist also nur dann funktionsfähig, wenn alle Kanten intakt sind. In der Anwendung mag sich dies aus der Forderung ergeben haben, dass sich alle Stationen gegenseitig unter Zuhilfenahme höchstens einer Zwischenstation erreichen können.
Es ergibt sich

$$\begin{aligned} z(G) &= P\{k_1 \wedge k_2 \wedge k_3 \wedge k_4\} \\ &= 0,99^4 \approx 0,960596 \end{aligned}$$

Im zweiten Fall sei $\mathcal{F}_2(G) = \{\{k_1, k_2, k_3, k_4\}, \{k_1, k_2, k_3\},$

$\{k_1, k_2, k_4\}, \{k_1, k_3, k_4\}, \{k_2, k_3, k_4\}\}$.

Funktionsfähigkeit korrespondiert hier zu dem Zusammenhang des Graphen. Für die Zuverlässigkeit folgt

$$z(G) = 0,99^4 + 4 \cdot 0,99^3 \cdot 0,01 \approx 0,999408.$$

Schon in Abschnitt 9.1 wurde darauf hingewiesen, dass die Zuverlässigkeit auch von dem im Netz angewendeten Routing (und in der Realität selbstverständlich von weiteren Protokollfestlegungen) abhängt. Die Beispiele in Beispiel 9.3-1 zeigen, dass uns dies durch den Begriff der funktionsfähigen Zustände nicht verloren gegangen ist; bei der Auswahl der Menge $\mathcal{F}(G)$ kann das Routing mit berücksichtigt werden.

Wie wir gesehen haben, ist es eine naheliegende Möglichkeit, die funktionsfähigen Zustände mit den zusammenhängenden Graphen zu identifizieren, d. h. $\mathcal{F}(G)$ als diejenigen Kantenmengen von $G$ zu wählen, die zu einem zusammenhängenden Teilgraphen führen. Die Zuverlässigkeit $z(G)$ korrespondiert dann zur Wahrscheinlichkeit, dass je zwei Knoten miteinander kommunizieren können, wobei alle Wege erlaubt sind; dieses spezielle Maß wird im folgenden mit $z_v(G)$ für **volle Zuverlässigkeit** bezeichnet.

**volle Zuverlässigkeit**

Zu einem anderen Maß kommt man, wenn nur die Forderung aufgestellt wird, dass zwei bestimmte Ecken $s, t$ des Graphen (auf beliebigen Wegen) kommunizieren können sollen: Die $s,t$-**Zuverlässigkeit** $z_{s,t}(G)$ ist dann die Wahrscheinlichkeit, dass die Ecken $s$ und $t$ in derselben Zusammenhangskomponente des Graphen liegen.

$s,t$-**Zuverlässigkeit**

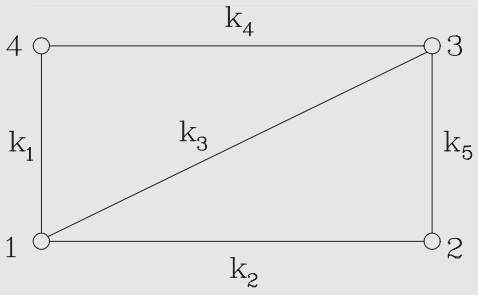
**Beispiel 9.3-2:**
Wir betrachten wieder den Graphen aus Beispiel 9.3-1 und fragen nach $z_{1,4}(G)$. Offenbar gilt

$$\begin{aligned} z_{1,4}(G) &= P\{k_4 \vee (k_1 \wedge k_2 \wedge k_3)\} \\ &= P\{k_4 \vee (k_1 \wedge k_2 \wedge k_3 \wedge k_4')\} \\ &= 0,99 + 0,99^3 \cdot 0,01 \\ &= 0,999703. \end{aligned}$$

Ein weiteres Zuverlässigkeitsmaß ergibt sich, wenn mehr als zwei Ecken $e_1, e_2, e_3, ..., e_t$ vorgegeben werden und das Ziel darin besteht, die gegenseitige Erreichbarkeit dieser Stationen sicherzustellen. Dieses Maß werden wir jedoch nicht weiter betrachten.

Wir wollen auch noch einmal darauf hinweisen, dass unser allgemeiner Ansatz über $\mathcal{F}(G)$ mit der Definition von $z(G)$ u. a. auch auf Netze anwendbar ist, in denen (wie im Beispiel 9.1-2) mit Routing-Plänen gearbeitet wird.

**Beispiel 9.3-3:**
Wir betrachten wieder den folgenden Graphen mit Routing-Plan:




| von/nach | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | - | 2;3 | 3;2 | 4;3 |
| 2 | 1;3 | - | 3;1 | 1;3 |
| 3 | 1;2 | 2;1 | - | 4;1 |
| 4 | 1;3 | 1;3 | 3;1 | - |

Soll in dem Netz jede Station zu jeder anderen Nachrichten senden können, so gilt für die funktionsfähigen Zustände

$$\begin{aligned} \mathcal{F}(G) = \{&K, K\backslash\{k_1\}, K\backslash\{k_2\}, K\backslash\{k_3\}, \\ &K\backslash\{k_4\}, K\backslash\{k_5\}, K\backslash\{k_1, k_2\}, K\backslash\{k_4, k_5\}, \\ &K\backslash\{k_1, k_5\}, K\backslash\{k_2, k_4\}\}\ . \end{aligned}$$

Sind die Kantenwahrscheinlichkeiten einheitlich gleich $p$, so hat man

$$\begin{aligned} z(G) &= p^5 + 5p^4(1-p) + 4p^3(1-p)^2 \\ &= 4p^3 - 3p^4, \end{aligned}$$

für $p = 0,99$

$$z(G) \approx 0,999408.$$

Es sei ein beliebiger (zusammenhängender) Graph $G = (E, K)$ auf der Eckenmenge $E = \{1, ..., n\}$ gegeben, $\mathcal{F} \subseteq \mathcal{P}(K)$ sei die Menge der funktionsfähigen Zustände. Es wird angenommen, dass es eine einheitliche Wahrscheinlichkeit $p(0 \leq p \leq 1)$ für die Funktionfähigkeit der einzelnen Kanten gibt, die Ereignisse "Kante $k$ ist intakt" (für $k \in K$) seien voneinander unabhängig.

Wir werden im folgenden stets von dieser Grundsituation ausgehen. Die Annahme einer einheitlichen Wahrscheinlichkeit $p$ stellt zwar eine Einschränkung dar, jedoch können damit die Zusammenhänge zwischen Graphenstruktur und Zuverlässigkeit klarer herausgearbeitet werden.

Die Zuverlässigkeit $z(G)$ ist die Wahrscheinlichkeit, dass $G$ sich in einem Zustand aus $\mathcal{F}$ befindet. $z(G)$ ist eine Funktion von $p$.

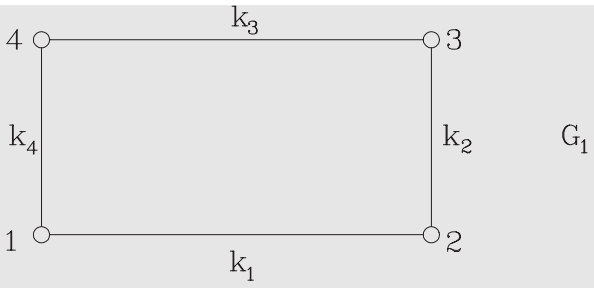
**Lemma 9.3-1:** *$z(G)$ ist ein Polynom in $p$.*

**Beweis:** Es sei $G_{\mathcal{F}} = \{(E, \tilde{K}) \mid \tilde{K} \in \mathcal{F}\}$ die Menge der funktionsfähigen Teilgraphen von $G$. Offenbar gilt

$$\begin{aligned} z(G) &= P\{G_{\mathcal{F}}\} \\ &= \sum_{T \in G_{\mathcal{F}}} P\{T\}. \end{aligned}$$

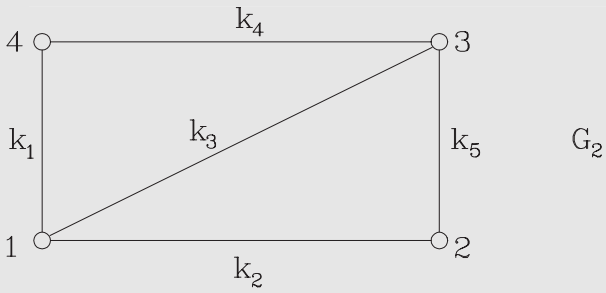
Da jede einzelne Wahrscheinlichkeit $P\{T\}$ ein Polynom in $p$ ist, gilt dies auch für $z(G)$.

**Beispiel 9.3-4:**
Wir schauen wieder die in Beispiel 9.3-1 und Beispiel 9.3-3 betrachteten Graphen an.




Soll $G_1$ nur funktionsfähig sein, wenn alle Kanten intakt sind (also $\mathcal{F}(G_1) = \{\{k_1, k_2, k_3, k_4\}\}$ ), so ergibt sich $z(G_1) = p^4$.

Soll Funktionsfähigkeit zum Zusammenhang korrespondieren, so erhält man (s. auch Beispiel 9.3-1)$z(G_1) = z_v(G_1) = 4p^3 - 3p^4$.




Da es in $G_2$ acht Teilgraphen mit drei Kanten, fünf Teilgraphen mit vier Kanten und einen Teilgraphen mit fünf Kanten gibt, die zusammenhängend sind, führt die Forderung des Zusammenhangs bei $G_2$ zu

$$z(G_2) = z_v(G_2) = 8p^3(1-p)^2 + 5p^4(1-p) + p^5$$

$$= 8p^3 - 11p^4 + 4p^5.$$

Der in Beispiel 9.3-3 verwendete Routing-Plan ergab

$$z(G_2) = 4p^3 - 3p^4.$$

**Zuverlässigkeitspolynom**

Das Polynom $z(G)$ wird **Zuverlässigkeitspolynom** des Graphen $G$ genannt. Das Polynom hängt von den als funktionsfähig definierten Zuständen ab, wie auch in den gezeigten Beispielen deutlich wird. In der Literatur wird dieser Bezug auf die Menge $\mathcal{F} \subseteq \mathcal{P}(K)$ mitunter nicht deutlich herausgestellt und implizit angenommen, dass die funktionsfähigen Zustände genau den zusammenhängenden Teilgraphen entsprechen – m. a. W. wird (in unserer Terminologie) nur $z_v(G)$ betrachtet.

Wir wollen an dieser Stelle noch einmal die beiden Methoden herausstellen, die wir bisher für die effektive Bestimmung des Polynoms $z(G)$ bei gegebenem Graphen $G$ (und gegebener Menge $\mathcal{F}$ funktionsfähiger Zustände) zur Verfügung haben.

**Boolesche Methode**

Die erste Methode könnte man die **Boolesche Methode** nennen: für jede Kante $k \in K$ wird die Aussage "$k$ ist intakt" durch den Booleschen Ausdruck "$k$" repräsentiert, und es wird ein zusammengesetzter Boolescher Ausdruck aufgestellt, dessen Gültigkeit genau der Aussage entspricht, dass man sich in einem funktionsfähigen Zustand befindet. Anschließend muss dieser Boolesche Ausdruck in eine DNF umgewandelt werden, um $z(G)$ direkt berechnen zu können. Wir haben diese Methode in einigen Beispielen angewendet.

**Teilgraph-Zählmethode**

Der andere Zugang ist die **Teilgraph-Zählmethode**: er greift auf die Beziehung

$$z(G) = \sum_{T \in G_{\mathcal{F}}} P\{T\}$$

zurück, wobei $G_{\mathcal{F}}$ die Menge der funktionsfähigen Teilgraphen ist. Hat ein $T \in G_{\mathcal{F}}$ $i$ viele Kanten, so gilt (mit $\mid K \mid = m$)

$$P\{T\} = p^i(1-p)^{m-i}.$$

Damit hat man insgesamt

$$z(G) = \sum_{i=0}^{m} F_i p^i \cdot (1-p)^{m-i},$$

wenn mit $F_i$ die Anzahl funktionsfähiger Teilgraphen mit $i$ vielen Kanten bezeichnet ist. Es gilt also, die Zahlen $F_i$ zu bestimmen.

Die Brücke zwischen den beiden Methoden stellt folgende Beobachtung her: Ist $f(k_1, ..., k_m)$ der Boolesche Ausdruck, der die Funktionsfähigkeit des Netzes beschreibt, so entsprechen die einzelnen Konjunktionen in der kanonischen DNF genau den funktionsfähigen Kantenmengen (also den Elementen von $\mathcal{F}$).

Beide Methoden unterscheiden sich daher nicht in ihrem Aufwand.

**Beispiel 9.3-5:**
Wir schauen noch einmal auf den Graphen $G_2$ aus Beispiel 9.3-4, der auf Zusammenhang überprüft werden soll. Wie in Beispiel 9.3-4 gesagt, gilt $F_3 = 8$, $F_4 = 5$ und $F_5 = 1$, also

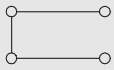
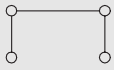
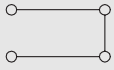
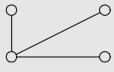
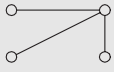
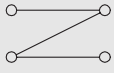
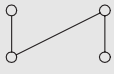
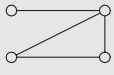
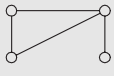
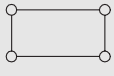
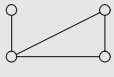
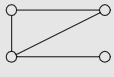
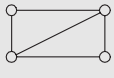
$$z_v(G_2) = 8p^3(1-p)^2 + 5p^4(1-p) + p^5.$$

Ein Boolescher Ausdruck, der den Zusammenhang beschreibt, ist z. B.

$$\begin{aligned} f(k_1, ..., k_5) \quad = \quad & (k_3 \wedge (k_1 \vee k_4) \wedge (k_2 \vee k_5)) \\ & \vee (k_3' \wedge ((k_1 \wedge k_2 \wedge k_4) \vee (k_1 \wedge k_2 \wedge k_5) \\ & \vee (k_1 \wedge k_4 \wedge k_5) \vee (k_2 \wedge k_4 \wedge k_5))) \end{aligned}$$

Zu diesem Ausdruck kommt man durch folgende Überlegung: Man unterscheidet zunächst danach, ob $k_3$ intakt ist oder nicht; im ersten Fall muss noch jeweils $k_1$ oder $k_4$ bzw. $k_2$ oder $k_5$ intakt sein; im zweiten Fall müssen mindestens drei der restlichen Kanten intakt sein.

Die zu $f$ gehörende kanonische DNF hat die folgenden 14 "Summanden", die genau den 14 zusammenhängenden Teilgraphen von $G_2$ entsprechen:

| Boolescher Ausdruck | zugehöriger funktionsfähiger Teilgraph |
|---|---|
| $k_1 \wedge k_2 \wedge k_3' \wedge k_4 \wedge k_5'$ |  |
| $k_1 \wedge k_2' \wedge k_3' \wedge k_4 \wedge k_5$ |  |
| $k_1' \wedge k_2 \wedge k_3' \wedge k_4 \wedge k_5$ |  |
| $k_1 \wedge k_2 \wedge k_3' \wedge k_4' \wedge k_5$ |  |
| $k_1 \wedge k_2 \wedge k_3 \wedge k_4' \wedge k_5'$ |  |
| $k_1' \wedge k_2' \wedge k_3 \wedge k_4 \wedge k_5$ |  |
| $k_1' \wedge k_2 \wedge k_3 \wedge k_4 \wedge k_5'$ |  |
| $k_1 \wedge k_2' \wedge k_3 \wedge k_4' \wedge k_5$ |  |
| $k_1' \wedge k_2 \wedge k_3 \wedge k_4 \wedge k_5$ |  |
| $k_1 \wedge k_2' \wedge k_3 \wedge k_4 \wedge k_5$ |  |
| $k_1 \wedge k_2 \wedge k_3' \wedge k_4 \wedge k_5$ |  |
| $k_1 \wedge k_2 \wedge k_3 \wedge k_4' \wedge k_5$ |  |
| $k_1 \wedge k_2 \wedge k_3 \wedge k_4 \wedge k_5'$ |  |
| $k_1 \wedge k_2 \wedge k_3 \wedge k_4 \wedge k_5$ |  |

**Graphenzerlegungsmethode**

Die "Boolesche Methode" lässt sich an einer **Graphenzerlegungsmethode** nachspielen. Das Verfahren leuchtet unmittelbar ein und wird anhand des Beispiels vorgeführt.

Grundidee ist, schrittweise durch Identifizierung von Ecken bzw. Streichen von Kanten kleinere Graphen zu konstruieren, die sich aus der Annahme ergeben, dass eine bestimmte Kante mit Sicherheit intakt bzw. nicht intakt ist.

( In den Zwischenschritten können sich Multigraphen ergeben.)

$$
\begin{aligned}
z_v(\ \boxtimes\ ) &= p \cdot z_v(\ \ominus\ ) + (1-p) \cdot z_v(\ \square\ ) \\
&= p \cdot (2-p^2)^2 + (1-p) \cdot (4p^3 - 3p^4) \\
&= 8p^3 - 11p^4 + 4p^5
\end{aligned}
$$

Sämtliche geschilderten Methoden zur Bestimmung des Polynoms haben den Nachteil, keine guten (d. h. effizienten) Algorithmen zu liefern.

Freilich kann in Einzelfällen (z. B. aufgrund irgendwelcher Symmetrien in der Problemstellung) die eine oder andere Methode schnell zum Ziel führen, jedoch sind für den allgemeinen Fall bei jeder Methode nur exponentielle Algorithmen bekannt.

Im nächsten Abschnitt werden wir auf die Schwierigkeit der Bestimmung der Koeffizienten $F_i$ noch einmal im Sinne der Komplexitätstheorie eingehen.

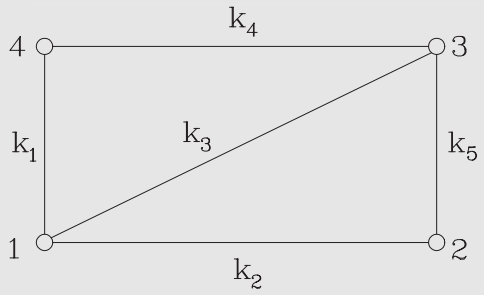
Wir wollen einen weiteren Begriff einführen, der das weitere Vorgehen erleichtert. Gegeben seien wieder $G$ und $\mathcal{F}$ wie in Lemma 9.3-1.

**kritische Kantenmenge**

$T \subseteq K$ heißt **kritische Kantenmenge**, wenn das Komplement $K \backslash T$ nicht funktionsfähig ist, d. h. wenn $K \backslash T \notin \mathcal{F}$ gilt. Anschaulich bedeutet das: $T$ ist kritisch, wenn der Ausfall von $T$ zu einem nicht funktionsfähigen Zustand führt.

Man beachte: Korrespondiert Funktionsfähigkeit zum Zusammenhang, so ist eine Kantenmenge kritisch, wenn durch ihr Entfernen der Graph in mindestens zwei Zusammenhangskomponenten zerfällt. Dies zeigt einen Zusammenhang zu dem Begriff eines Schnitts auf (vergleiche Abschnitt 1.3). Die Begriffe sind jedoch nicht identisch.

**Beispiel 9.3-6:**
Es sei wieder der folgende Graph gegeben, Funktionsfähigkeit korrespondiere zum Zusammenhang:




Funktionfähig sind z. B. $\{k_1, k_2, k_3\}$ oder $\{k_1, k_2, k_4, k_5\}$, kritisch z. B. $\{k_1, k_2, k_3\}$ oder $\{k_1, k_4\}$

Aufgrund der Definition von funktionsfähig und kritisch folgt unmittelbar:

Ist $S \subseteq K$ eine beliebige Kantenmenge, so ist

| | |
|---|---|
| entweder | $S$ funktionsfähig |
| oder | $K \backslash S$ kritisch. |

Ist mit $C_j (0 \leq j \leq m)$ die Anzahl der kritische Kantenmengen mit $j$ vielen Kanten bezeichnet, so ergibt obige Aussage die Formel

$$F_i + C_{m-i} = \binom{m}{i} \qquad \text{für } 0 \leq i \leq m.$$

Daraus folgt nun der

**Satz 9.3-1:**

$$z(G) = 1 - \sum_{j=0}^{m} C_j p^{m-j} (1-p)^j$$

**Beweis:** Man hat

$$\begin{aligned} z(G) &= \sum_{i=0}^{m} F_i p^i (1-p)^{m-i} \\ &= \sum_{i=0}^{m} (\tbinom{m}{i} - C_{m-i}) p^i (1-p)^{m-i} \\ &= \sum_{j=0}^{m} (\tbinom{m}{j} - C_j) p^{m-j} (1-p)^j, \end{aligned}$$

und wegen

$$\sum_{j=0}^{m} \binom{m}{j} p^{m-j} (1-p)^j = 1$$

folgt die Behauptung.

Man kann die behauptete Formel allerdings auch direkt einsehen, wenn man berücksichtigt, dass $1 - z(G)$ die Wahrscheinlichkeit ist, dass mindestens eine kritische Menge nicht intakt ist.

**Selbsttestaufgabe 9.3-1:**

*Zeigen Sie: Korrespondieren die funktionsfähigen Zustände zu zusammenhängenden Teilgraphen, und wird einem Graphen $G = (E, K)$ eine Kante hinzugefügt, die bisher nicht zu $K$ gehörte, so ist der neue Graph $G'$ zuverlässiger als $G$.*

## 9.4 Zur Komplexität des Zuverlässigkeitsproblems

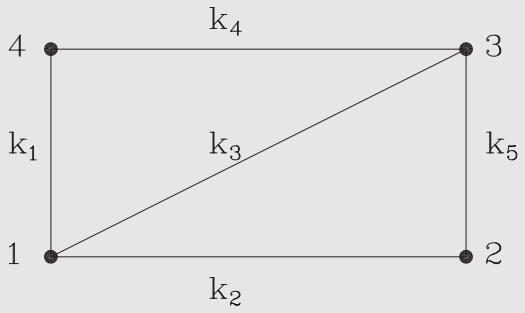
In diesem Abschnitt wird stets von folgender Situation ausgegangen: Es ist ein ungerichteter Graph $G = (E, K)$ mit einer Menge $\mathcal{F} \subseteq \mathcal{P}(K)$ funktionsfähiger Zustände gegeben. Jede Kante $k \in K$ ist mit einer einheitlichen Wahrscheinlichkeit $p$ verfügbar.

Es geht um das Problem, die Zuverlässigkeit $z(G)$ als Polynom in der Variablen $p$ effizient zu bestimmen.

An dieser Stelle ist eine Präzisierung des Zieles nötig, das Zuverlässigkeitspolynom "effizient zu bestimmen". Wir müssen hier zunächst an die Ausführungen zur Komplexität von Algorithmen in Kapitel 3 erinnern. Grundsätzlich ist man stets auf der Suche nach Algorithmen mit einer polynomialen Komplexität relativ zur Größe des gegebenen Problems; als Problemgröße wählen wir wieder stets die Eckenanzahl des Graphen.

Die Problematik verkompliziert sich nun dadurch, dass zusätzlich eine Menge $\mathcal{F} \subseteq \mathcal{P}(K)$ als "Input" in den gesuchten Algorithmus zur Bestimmung von $z(G)$ eingeht. Für die Frage nach der Komplexität spielt es natürlich auch eine Rolle, in welcher Form die Menge $\mathcal{F}$ gegeben ist, ob also z. B. $\mathcal{F}$ als Menge in Worten beschrieben ist oder etwa die Elemente von $\mathcal{F}$ in einer jederzeit greifbaren Liste abgespeichert sind.

**Beispiel 9.4-1:**
Der folgende Graph $G$ sei gegeben.




Die Menge $\mathcal{F}$ kann angegeben werden

- durch die explizite Auflistung
  $\mathcal{F} = \{\{k_1, ..., k_5\}, \{k_1, k_2, k_3, k_4\}, ...\}$ oder
- durch die Aussage "$\mathcal{F}$ korrespondiere zu den zusammenhängenden Teilgraphen".

Ein Verfahren zur Bestimmung von $z(G)$ kann im ersten Fall die Koeffizienten $F_i$ in der Darstellung

$$z(G) \;=\; \sum_{i=0}^{m} F_i \cdot p^i \cdot (1-p)^{m-i}$$

sofort ablesen, im zweiten Fall ist dies nicht möglich.

**Erkennungsalgorithmus**

Wir werden im folgenden stets die Situation vorliegen haben, dass es zu gegebenem $\mathcal{F}$ einen polynomialen **Erkennungsalgorithmus** für $\mathcal{F}$ gibt – also einen Algorithmus, der zu gegebener Kantenmenge $M \subseteq K$ feststellt, ob $M \in \mathcal{F}$ gilt oder nicht.

Man beachte, dass dies tatsächlich der Fall ist, wenn $\mathcal{F}$ zum Zusammenhang korrespondiert, denn es ist leicht zu testen, ob ein Graph zusammenhängend ist (vgl. Kapitel 3).

Man beachte aber auch: dies bedeutet nicht, dass es algorithmisch leicht ist, eine spezifische Auflistung der Elemente von $\mathcal{F}$ zu erstellen, denn es gibt schließlich exponentiell viele Kandidaten als Elemente von $\mathcal{F}$!

In dem vorliegenden Abschnitt beschränken wir uns auf die folgenden zwei Fälle:

**volles Zuverlässigkeitsproblem**

I. $\mathcal{F}$ korrespondiert zu den zusammenhängenden Teilgraphen von $G$. Wir sprechen dann vom **vollen Zuverlässigkeitsproblem** .

**s,t-Zuverlässigkeitsproblem**

II. Es sind Ecken $s$ und $t$ des Graphen ausgezeichnet, und $\mathcal{F}$ korrespondiert zu denjenigen Teilgraphen von $G$, in denen $s$ und $t$ in derselben Zusammenhangskomponente liegen. Hier sprechen wir vom **s,t-Zuverlässigkeitsproblem**.

Es geht also in I. um die Bestimmung von $z_v(G)$ und in II. um die von $z_{s,t}\,(G)$ (vgl. Abschnitt 9.3).

Wie bereits mehrfach dargelegt, gilt für das Polynom $z(G)$ stets die Darstellung

$$z(G) \;=\; \sum_{i=0}^{m} F_i \;\cdot\; p^i \;\cdot\; (1-p)^{m-i}\,,$$

wobei $F_i$ die Anzahl der $i$-elementigen Elemente von $\mathcal{F}$ ist. Hat man ein effizientes Verfahren, die Zahlen $F_0, F_1, ..., F_m$ zu berechnen, so kann das Polynom $z(G)$ effizient bestimmt und für jedes konkrete $p_0$ der Wert $z(p_0)$ berechnet werden. Zur Vereinfachung der Sprechweise wollen wir im folgenden das Problem der Bestimmung von $F_0, F_1, ..., F_m$ mit **ZUV-K** bezeichnen.

**ZUV-K**

Es ist aber auch vorstellbar, dass man einen (effizienten) Algorithmus hat, der für jeden beliebigen Wert $p_0$ $(0 \le p_0 \le 1)$ den Funktionswert $z(G)(p_0)$ liefert, ohne dass dieser Algorithmus den "Umweg" der Bestimmung der obigen Koeffizienten $F_i$ nimmt. Das Problem des Findens eines solchen Algorithmus bezeichnen wir mit **ZUV**.

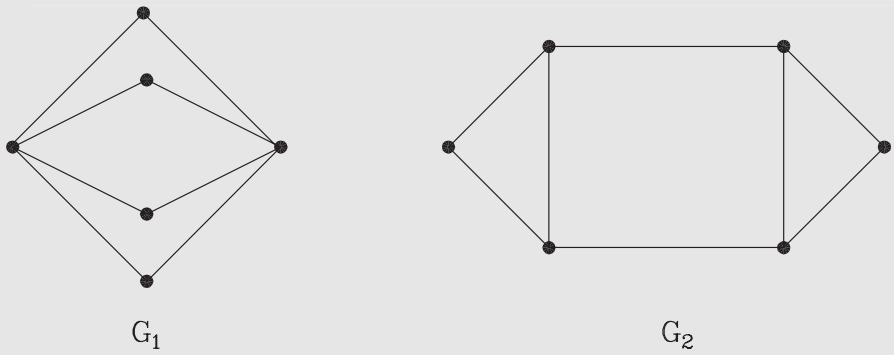
**ZUV**

Bevor wir als nächstes zeigen, dass die Probleme **ZUV-K** und **ZUV** (im algorithmischen Sinne) äquivalent sind, ist eine grundsätzliche Bemerkung zum verschiedenen Charakter dieser Problemstellungen angebracht.

Die Formulierung der Problemstellung **ZUV-K** hat den Vorteil, dass es sich bei der Bestimmung der Zahlen $F_0, F_1, ..., F_m$ um ein kombinatorisches Problem handelt, wo der Parameter $p$ keine Rolle mehr spielt. Der Nachteil ist dabei allerdings, dass die Koeffizienten $F_i$ nicht unmittelbar eine Aussage über die Werte $z(G)(p_0)$ treffen. Dies gilt umso mehr, als sich die Zuverlässigkeitspolynome verschiedener Graphen auch "kreuzen" können.

**Beispiel 9.4-2:**
(vgl. [COL])

Die folgenden Graphen $G_1$ und $G_2$ seien gegeben, es werde $z_v(G)$ untersucht.



$G_1$ $G_2$

Beide Graphen haben 6 Ecken und 8 Kanten. Die Koeffizienten $F_i$ sind im folgender Übersicht zusammengestellt:

| | $G_1$ | $G_2$ |
|---|---|---|
| $F_0$ | 0 | 0 |
| $F_1$ | 0 | 0 |
| $F_2$ | 0 | 0 |
| $F_3$ | 0 | 0 |
| $F_4$ | 0 | 0 |
| $F_5$ | 32 | 30 |
| $F_6$ | 24 | 25 |
| $F_7$ | 8 | 8 |
| $F_8$ | 1 | 1 |

Den Koeffizienten $F_i$ ist nicht sofort anzusehen, welcher Graph "zuverlässiger" ist. Tatsächlich kreuzen sich die Funktionen $z_v(G_1)(p)$ und $z_v(G_2)(p)$ bei $p = \frac{2}{3}$: für $p < \frac{2}{3}$ ist $G_1$ zuverlässiger, für $p > \frac{2}{3}$ dagegen $G_2$.

**Reduzierbarkeit Äquivalenz**

Es werden nun einige Bemerkungen zu den Begriffen **Reduzierbarkeit** und **Äquivalenz** von Problemen eingefügt.

**reduzierbar**

Man sagt, ein Problem **A** sei auf ein Problem **B reduzierbar**, wenn folgendes gilt: Hat man einen polynomialen Algorithmus, der das Problem **B** löst, so kann man daraus einen polynomialen Algorithmus zum Lösen von **A** ableiten.

Auf eine formale Definition soll an dieser Stelle verzichtet werden. Der (die) interessierte Leser(in) findet dies in jedem Informatik-Lehrbuch, das sich mit Fragen der Komplexität von Algorithmen befasst (siehe z. B. das Standardwerk [GAR]).

**Beispiel 9.4-3:**
In Kapitel 3 wurde das Problem der **minimalen Knotenüberdeckung** betrachtet: in einem gegebenen Graphen $G = (E, K)$ ist eine Menge $U \subseteq E$ mit möglichst wenigen Ecken zu finden derart, dass von jeder Kante aus $K$ mindestens eine der beiden Ecken zu $U$ gehört.

Beim Problem der **maximalen unabhängigen Menge** geht es darum, eine Menge $M \subseteq E$ mit möglichst vielen Ecken zu finden derart, dass keine Kante aus $K$ zwei Ecken von $M$ verbindet.

**Behauptung:**

Das Problem der maximalen unabhängigen Menge ist auf das Problem der minimalen Knotenüberdeckung reduzierbar.

**Beweis:** Es sei $\mathcal{A}$ ein polynomialer Algorithmus, der in einem beliebigen Graphen eine minimale Knotenüberdeckung findet. Es wird nun ein polynomialer Algorithmus $\mathcal{B}$ angegeben, der – unter Verwendung von $\mathcal{A}$ - eine maximale unabhängige Menge in einem Graphen bestimmt.

Algorithmus $\mathcal{B}$ geht folgendermaßen vor:

Ein Graph $G = (E, K)$ sei gegeben. Mit Algorithmus $\mathcal{A}$ wird zunächst eine minimale Knotenüberdeckung $U_o$ ermittelt. Sodann wird das Komplement $M_o = E \setminus U_o$ gebildet. Da offenbar eine Teilmenge $X \subseteq E$ genau dann eine Knotenüberdeckung ist, wenn $E \setminus X$ unabhängige Menge ist, muss $M_o$ eine maximale unabhängige Menge sein.

Da der Algorithmus $\mathcal{B}$ nur aus einmaligem Aufruf von $\mathcal{A}$ und der Komplementbildung besteht, ist $\mathcal{B}$ ebenfalls polynomial.

**äquivalent**

Probleme **A** und **B** werden **äquivalent** genannt, wenn sowohl **A** auf **B** als auch **B** auf **A** reduzierbar ist.

Es ist hier wichtig anzumerken, dass Reduzierbarkeit (und Äquivalenz) nur eine Aussage über das Verhältnis zweier Probleme zueinander ist und nichts über die Existenz polynomialer Algorithmen aussagt – "**A** ist reduzierbar auf **B**" bedeutet nur: wenn es einen solchen Algorithmus für **B** gibt, so auch für **A**.

Aus obigen Beweis wird sofort klar, dass die Probleme der minimalen Knotenüberdeckung bzw. der maximalen unabhängigen Menge sogar äquivalent sind. Polynomiale Algortihmen zu ihrer Lösung sind jedoch nicht bekannt.

Wie in Kapitel 3 angemerkt, ist das Problem der minimalen Knotenüberdeckung (und damit auch das der maximalen unabhängigen Menge) **NP-vollständig**. Dies bedeutet unter anderem, dass dafür bisher keine polynomialen Algorithmen gefunden wurden und es auch als unwahrscheinlich gilt, dass solche überhaupt existieren.

**NP-vollständig**

Bei der Untersuchung des Schwierigkeitsgrades eines Problems $\mathbf{A}$ wird oft in folgender Weise vorgegangen: Zunächst versucht man, einen polynomialen Algorithmus zu finden oder zu zeigen, dass kein solcher existiert - letzteres gelingt äußerst selten. Gelingt beides nicht, so versucht man, ein als NP-vollständig bekanntes Problem $\mathbf{B}$ auf $\mathbf{A}$ zu reduzieren. Wenn dies funktioniert, so weiß man, dass $\mathbf{A}$ mindestens so schwierig ist wie ein NP-vollständiges Problem, und es ist dann gerechtfertigt, nach guten approximativen Algorithmen zu suchen. Wir werden ebenfalls in dieser Weise vorgehen.

Zunächst beweisen wir jedoch den folgenden

**Satz 9.4-1:** *Die Probleme **ZUV** und **ZUV-K** sind äquivalent.*

**Beweis:** Es wurde bereits oben argumentiert, dass das Problem **ZUV** auf das Problem **ZUV-K** reduzierbar ist.

Für die Umkehrung sei nun angenommen, man habe einen polynomialen Algorithmus $\mathcal{A}$ zur Lösung des Problems **ZUV**. Die folgende Vorgehensweise liefert dann einen polynomialen Algorithmus $\mathcal{B}$ zur Bestimmung der Koeffizienten $F_o, F_1, ..., F_m$: Man wählt zunächst $m+1$ verschiedene reelle Zahlen $p_0, p_1, ..., p_m$. Durch $(m+1)$ –fache Anwendung des Algorithmus $\mathcal{A}$ können die Werte $z(G)(p_t)$ $(0 \leq t \leq m)$ berechnet werden. Für jedes $t$ gilt nun die Gleichung

$$\sum_{i=0}^{m} F_i \, p_t^i \, (1-p_t)^{m-i} \;=\; z(G)(p_t) \;,$$

in der die Potenzen von $p_t$ bzw. $(1-p_t)$ berechnet werden können, so dass daraus eine lineare Gleichung mit den Unbekannten $F_0, F_1, ..., F_m$ wird. Insgesamt hat man nun ein lineares Gleichungssystem aus $m+1$ vielen Gleichungen und mit $m+1$ vielen Unbekannten, welches mit dem Gaußschen Algorithmus eindeutig gelöst werden kann. Da der Gaußsche Algorithmus ebenfalls polynomial (in $m$) ist, bleibt auch der so beschriebene Algorithmus $\mathcal{B}$ polynomial.

Der soeben bewiesene Satz rechtfertigt, dass wir im folgenden das Problem **ZUV-K** – also die Bestimmung von $F_0, F_1, ..., F_m$ – als *das* Zuverlässigkeitsproblem ansehen. Der Einfachheit halber wird *dieses* Problem im weiteren mit **ZUV** bezeichnet. Soll spezifiziert werden, ob es sich um das volle Zuverlässigkeitsproblem oder um das s,t-Zuverlässigkeitsproblem handelt, so werden die Bezeichnungen $\mathbf{ZUV}_v$ bzw. $\mathbf{ZUV}_{s,t}$ verwendet.

Bevor wir Aussagen über die Komplexität des Problems **ZUV** ableiten, ist eine ergänzende Bemerkung zum Thema NP–Vollständigkeit nötig. In der Komplexitätstheorie wird der Begriff der NP–Vollständigkeit nur auf **Entscheidungsprobleme** angewendet. Ein Entscheidungsproblem bildet z. B. die Frage: "Gibt es in dem Graphen $G$ eine Knotenüberdeckung aus $n_o$ vielen Knoten?" Die Frage "Wieviele Knotenüberdeckungen aus $n_o$ vielen Knoten hat der Graph $G$ ?" ist ein **Zählproblem**. Für Zählprobleme gibt es eigene Begriffe, obschon natürlich Zusammenhänge bestehen zwischen "Entscheidungsversion" und "Zählversion" eines Problems.

**Entscheidungsproblem**

**Zählproblem**

Da wir den Begriff der NP–Vollständigkeit ohnehin nicht formal eingeführt haben, werden wir im folgenden so verfahren: unabhängig davon, ob es sich um Entscheidungs- oder Zählversion eines Problems handelt, werden wir von einem **schwierigen Problem** sprechen, wenn damit gesagt werden soll, dass es zur Lösung bisher kein polynomialer Algorithmus gefunden wurde. Diese Terminologie beinhaltet die Korrektheit der folgenden Schlussweise: ist Problem **A** als schwierig erkannt und auf **B** reduzierbar, so ist auch **B** schwierig.

**schwieriges Problem**

In Beispiel 9.4-3 war von dem Problem der maximalen unabhängigen Menge die Rede. Das Problem der **bipartiten unabhängigen Menge**, welches wir mit **BUM** abkürzen, lautet folgendermaßen: Man bestimme für einen gegebenen bipartiten Graphen $G$ die Anzahl der unabhängigen Knotenmengen.

**bibipartite unabhängige Menge**

**BUM-Problem**

Aus der Literatur (s. z. B. [COL]) übernehmen wir die Information, dass auch **BUM** NP-vollständig ist.

Um uns dem Zuverlässigkeitsproblem anzunähern, betrachten wir nun noch das Problem $s,t$-**Trennung**: Man bestimme für einen gegebenen (ausnahmsweise!) Multigraphen $G$ mit ausgezeichneten Knoten $s$ und $t$ die Anzahl der $s,t$-trennenden Kantenmengen minimaler Elementeanzahl.

$s,t$-**Trennung**

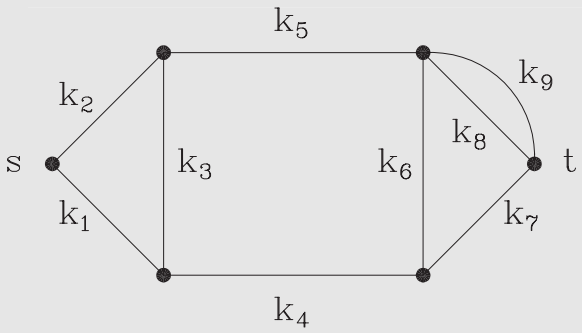
Hierbei ist eine $s,t$-trennende Kantenmenge im Sinne von Kapitel 1 zu verstehen: es ist eine Kantenmenge $S \subseteq K$, so dass nach Herausnahme der Kanten von $S$ die Knoten $s$ und $t$ in verschiedenen Zusammenhangskomponenten liegen.

Für einen zusammenhängenden Graphen $G$ sprechen wir allgemein von einer **trennenden Kantenmenge**, wenn nach deren Herausnahme aus dem Graphen dieser in mindestens zwei Zusammenhangskomponenten zerfällt. Bei gegebenen Knoten $s$ und $t$ ist also eine trennende Kantenmenge $S$ dann eine $s,t$-trennende Kantenmenge, wenn $s$ und $t$ nach der Herausnahme von $S$ in verschiedenen Komponenten liegen.

**trennende Kantenmenge**

Man beachte: Die trennenden Kantenmengen sind identisch mit den kritischen Kantenmengen des vollen Zuverlässigkeitsproblems. Die $s,t$-trennenden Kantenmengen sind die kritischen Kantenmengen des $s,t$-Zuverlässigkeitsproblems.

**Beispiel 9.4-4:**
Der im folgenden Diagramm dargestellte Multigraph sei gegeben.

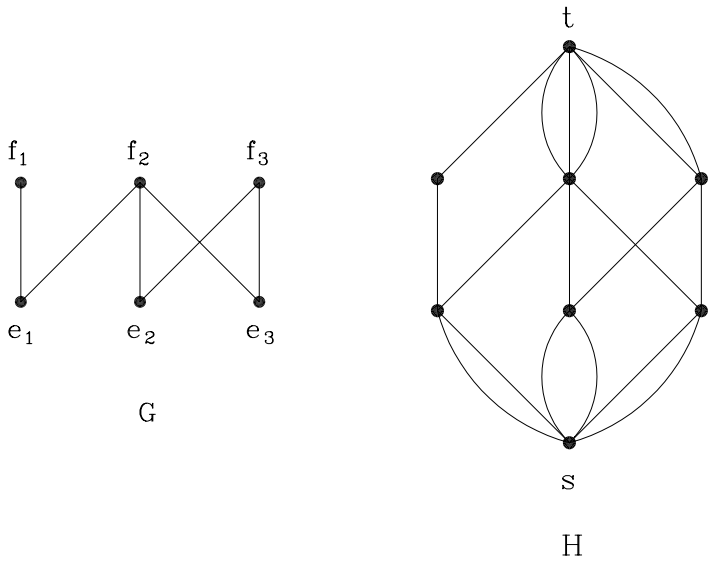



Die minimale Kardinalität einer $s,t$-trennenden Kantenmenge ist hier 2. Allgemein lässt sich diese kleinstmögliche Elementeanzahl mit dem Algorithmus von Ford und Fulkerson ermitteln (vgl. Kapitel 7). Die Anzahl der zweielementigen $s,t$-trennenden Kantenmengen (also die Lösung des Problems $s,t$-Trennung) ist hier 2, denn es sind dies nur die Mengen $S_1 = \{k_1, k_2\}$ und $S_2 = \{k_4, k_5\}$.

**Satz 9.4-2:** ***BUM*** *ist auf das Problem $s,t$-Trennung reduzierbar.*

**Beweis:** Es sei wieder vorausgesetzt, dass ein polynomialer Algorithmus $\mathcal{B}$ zur Lösung von $s,t$-Trennung existiert. Der Algorithmus $\mathcal{A}$ geht dann folgendermaßen vor:

Zu gegebenem bipartiten Graphen $G = (E \cup F, K)$ wird zunächst ein Multigraph $H = (\tilde{E}, \tilde{K})$ konstruiert. Die Knotenmenge $\tilde{E}$ von $H$ besteht aus $E \cup F$ und zwei zusätzlichen Knoten $s$ und $t$. Die Kantenmenge $\tilde{K}$ enthält die Menge $K$. Zusätzliche Kanten werden nach folgender Regel eingeführt: hat $e \in E$ in $G$ den Grad $\alpha$ , so werden $\alpha$ viele parallele Kanten zwischen $s$ und $e$ gezogen; entsprechendes gilt für $f \in F$ und $t$. Zur Verdeutlichung ist hier ein Beispiel dargestellt:




Aufgrund der Konstruktion enthält jede $s,t$-trennende Kantenmenge mindestens $|K|$ viele Kanten, und es gibt auch Schnitte mit dieser Elementeanzahl – z. B. die ursprüngliche Menge $K$. Wir zeigen nun, dass die $s,t$-trennenden Kantenmengen mit $|K|$ vielen Kanten in $H$ eindeutig den unabhängigen Knotenmengen in $G$ entsprechen.

Dazu sind zwei Vorüberlegungen nötig:

Gibt es zwischen $s$ und $e$ (oder zwischen $t$ und $f$) $\alpha > 1$ viele Kanten, so muss eine minimale trennende Kantenmenge entweder keine oder alle diese Kanten enthalten.

Zweitens:
Enthält eine minimale $s,t$-trennende Menge eine Kante zwischen $s$ und $e$ und eine zwischen $t$ und $f$, so kann $ef$ keine Kante in $G$ sein – andernfalls ergäbe sich nämlich eine $s,t$-trennende Menge mit mindestens einer Kante weniger, wenn man die Kanten zwischen $s$ und $e$ und die von $t$ nach $f$ aus diesem Schnitt entfernte und ersetzte durch alle Kanten von $G$, die bei $e$ oder $f$ enden.

Die eineindeutige Beziehung zwischen den minimalen $s,t$-trennenden Kantenmengen in $H$ und den unabhängigen Knotenmengen in $G$ lässt sich nun folgendermaßen beschreiben: Ausgehend von der minimalen $s,t$-trennenden Menge, die gleich der Menge $K$ der Kanten von $G$ ist (und der unabhängigen Knotenmenge $\phi$ entspricht), werden für eine unabhängige Teilmenge von $E \cup F$ die in Ecken dieser Teilmenge endenden Kanten durch die von den Knoten dieser Teilmenge zu $s$ bzw. $t$ führenden Kanten ersetzt. Die obigen Vorüberlegungen dienen dem Nachweis, dass man auf diese Weise alle minimalen $s,t$-trennenden Kantenmengen bekommt.

Wir kommen nun zu unserem Algorithmus $\mathcal{A}$ zurück. Nachdem $H$ konstruiert ist, wird darauf Algorithmus $\mathcal{B}$ angewendet, um die Anzahl der minimalen

$s,t$-Schnitte in $H$ zu ermitteln. Nach den obigen Überlegungen hat man damit auch die Anzahl der unabhängigen Knotenmengen in $G$.

Aus obigem Satz erhalten wir die

**Folgerung 9.4-1:** *Das Problem* $\boldsymbol{ZUV}_{s,t}$ *ist schwierig.*

**Beweis:** Es wird gezeigt, dass $s,t$- Trennung auf $\mathbf{ZUV}_{s,t}$ reduzierbar ist. Könnte man nämlich $\mathbf{ZUV}_{s,t}$ polynomial lösen, d. h. die Koeffizienten $F_0, F_1, ..., F_m$ mit polynomialem Aufwand berechnen, so hätte man auch die Zahlen

$$C_j \;=\; \binom{m}{j} \;-\; F_{m-j} \quad (0 \;\leq\; j \;\leq\; m)$$

zur Verfügung. Das erste $C_j \;\neq\; 0$ in der Folge $C_o, C_1, ..., C_m$ ist aber genau die Anzahl der minimalen $s,t$-trennenden Kantenmengen.

Abschließend wenden wir uns nun der Komplexität des Problems $\mathbf{ZUV}_v$ zu.

**Satz 9.4-3:** $\boldsymbol{ZUV}_v$ *ist schwierig.*

**Beweis:** Es wird auch hier gezeigt, dass $s,t$-Trennung auf $\mathbf{ZUV}_v$ reduzierbar ist.

Angenommen, man hätte einen polynomialen Algorithmus $\mathcal{A}$ zur Lösung von $\mathbf{ZUV}_v$. Es ergäbe sich dann der im folgenden beschriebene polynomiale Algorithmus $\mathcal{B}$ zur Lösung von $s,t$-Trennung. Gegeben sei ein Graph $G = (E, K)$ mit ausgezeichneten Ecken $s$ und $t$. Es sei $\alpha$ die minimale Elementeanzahl eines $s,t$-Schnittes. Wie oben angemerkt, kann $\alpha$ mit Hilfe von Netzflüssen mit polynomialem Aufwand berechnet werden. Wir betrachten nun in $G$ alle Schnitte (nicht nur die $s,t$-trennenden) aus $\alpha$ vielen Kanten. Anwendung des Algorithmus $\mathcal{A}$ auf $G$ liefert uns u. a. die Anzahl $a\alpha(G)$ dieser Kantenmengen.

Um die uns interessierende Anzahl von $s,t$-trennenden aus $\alpha$ vielen Kanten zu bekommen, müssen wir von $a\alpha(G)$ die Anzahl derjenigen Schnitte abziehen, die $s$ und $t$ nicht trennen.

Um die letztgenannte Anzahl zu bestimmen, bilden wir aus $G$ den Graphen $H$ durch Identifizierung der Ecken $s$ und $t$. Ein $s$ und $t$ nicht trennender Kantenmenge von $G$ bleibt dann auch in $H$ ein Schnitt; umgekehrt ist auch jede solche Menge von $H$ eine trennende von $G$, die $s$ und $t$ nicht trennt. Wir können den Algorithmus $\mathcal{A}$ nochmals anwenden, um die Anzahl $a\alpha(H)$ aller Schnitte in $H$ mit $\alpha$ vielen Kanten zu bestimmen. $a\alpha(G) \;-\; a\,\alpha\,(H)$ ist nun die gesuchte Anzahl.

Die Ergebnisse dieses Abschnitts deuten darauf hin, dass man mit hoher Wahrscheinlichkeit keine guten (also "schnellen") Algorithmen zur Lösung der betrachteten Zuverlässigkeitsprobleme finden wird. Wir werden uns daher im nächsten Abschnitt damit beschäftigen, möglichst gute (und leicht berechenbare) Abschätzungen für die Koeffizienten $F_i$ bzw. $C_i$ zu finden. Wie gezeigt werden wird, gibt es in dieser Richtung eine Reihe interessanter Ergebnisse.

**Selbsttestaufgabe 9.4-1:**

*Erläutern Sie, wieso es gerechtfertigt ist, das Zuverlässigkeitsproblem als das Problem der Bestimmung der Koeffizienten $F_0, F_1, ..., F_m$ anzusehen.*

## 9.5 Abschätzungen für das Zuverlässigkeitspolynom

Wir gehen wieder von der Darstellung

$$z(G) \;=\; \sum_{i=0}^{m} F_i \;\cdot\; p^i \;\cdot\; (1-p)^{m-i}$$

aus. Dabei ist $G$ ein Graph mit $n$ Ecken und $m$ Kanten, $F_i$ die Anzahl der $i-$elementigen funktionsfähigen Kantenmengen.

Um möglichst allgemeingültige Ergebnisse zu bekommen, machen wir in diesem Abschnitt keine Voraussetzungen darüber, welche Zuverlässigkeitsprobleme betrachtet (d. h. welche Menge $\mathcal{F}$ ausgewählt) werden – Einschränkungen werden nur dort gemacht, wo sie für die betreffende Argumentation benötigt werden.

$z(G)$ lässt sich auch darstellen als

$$1 \;-\; \sum_{j=0}^{m} C_j \;\cdot\; (1-p)^j \;\cdot\; p^{m-j} \;,$$

wobei $C_j$ die Anzahl der $j$-elementigen Schnitte ist. Es gilt der Zusammenhang

$$F_i \;+\; C_{m-i} \;=\; \binom{m}{i} \quad \text{für jedes } i \;.$$

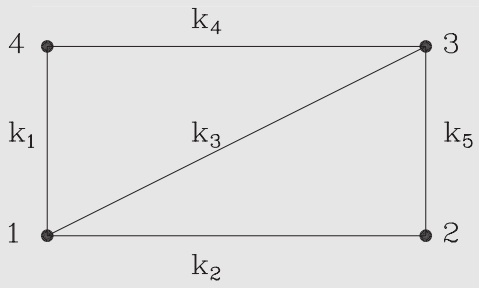
Eine dritte Darstellung, die oft verwendet wird und auch unseren weiteren Untersuchungen zugrundliegen soll, ergibt sich durch Einführung der Koeffizienten $R_i$:

für $0 \leq i \leq m$ sei $R_i$ die Anzahl derjenigen $i$-elementigen Kantenmengen, deren Komplemente funktionsfähig sind. M. a. W.

ist $R_i \;= F_{m-i}$ , und es gilt $z(G) \;=\; \sum_{i=0}^{m} R_i \;\cdot\; (1-p)^i \;\cdot\; p^{m-i}$.

(Während $F_i$ die Anzahl der funktionsfähigen und $C_i$ die Anzahl der kritischen Kantenmengen ("critical") mit $i$ vielen Kanten bezeichnen, soll $R_i$ die Restmenge suggerieren.)

**Beispiel 9.5-1:**
Es sei wieder der folgende Graph $G$ zugrundegelegt, Funktionsfähigkeit korrespondiere zum Zusammenhang:




Man hat $F_0 = F_1 = F_2 = 0\ ,\ F_3 = 8\ ,\ F_4 = 5\ ,\ F_5 = 1.$

Dies führt zu $z(G)\ =\ 8p^3\,(1-p)^2\ +\ 5p^4\,(1-p)\ +\ p^5\ .$

Weiter gilt $C_0 = C_1 = 0\ ,\ C_2 = 2\ ,\ C_3 = 10\ ,\ C_4 = 5\ ,\ C_5 = 1.$

Dies ergibt $z(G)\ =\ 1\ -\ 2\,(1-p)^2p^3\ -\ 10\,(1-p)^3p^2\ -\ 5\,(1-p)^4p\ -\ (1-p)^5.$

Schließlich ist $R_0 = 1\ ,\ R_1 = 5\ ,\ R_2 = 8\ ,\ R_3 = R_4 = R_5 = 0,$

was zu derselben Darstellung wie der ersten führt:

$$z(G)\ =\ p^5\ +\ 5\,(1-p)p^4\ +\ 8\,(1-p)^2\,p^3.$$

Die folgenden Parameter $l$ und $c$ spielen für die weiteren Überlegungen eine zentrale Rolle:

$l$ sei die minimale Größe einer funktionsfähigen Kantenmenge;
$c$ sei die minimale Größe einer kritischen Kantenmenge.

Im obigen Beispiel 9.5-1 ist offenbar $l = 3$ und $c = 2$.

Aus der Definition folgt sofort $F_i = 0$ für $i < l$ (und damit $R_j = 0$ für $j > m - l$) , ferner $C_i = 0$ für $i < c$.

Da eine $i$-elementige Kantenmenge mit $i\ <\ c$ keine kritische Kantenmenge sein kann, folgt außerdem
$R_i\ =\ \binom{m}{i}$ für $i\ <\ c$
bzw. $F_j\ =\ \binom{m}{j}$ für $j\ > m - c.$

In den Abschätzungen für $z(G)$, die hier dargestellt werden sollen, werden die Zahlen $R_c(=\ F_{m-c}\ =\ \binom{m}{c}\ -\ C_c)$ und $R_{m-l}(=\ F_l\ =\ \binom{m}{l}\ -\ C_{m-l})$ benötigt. Somit ist die Frage von Interesse, ob $l, c, R_c$ und $R_{m-l}$ leicht bestimmt werden können. Die Antwort hängt selbstverständlich davon ab, welches Zuverlässigkeitsproblem untersucht wird.

Betrachten wir zunächst das volle Zuverlässigkeitsproblem, bei dem $\mathcal{F}$ gerade alle zusammenhängenden Teilgraphen des gegebenen Graphen $G$ enthält.

Offensichtlich ist hier $l = n - 1$, denn die kleinsten zusammenhängenden Teilgraphen sind die Gerüste. $F_l$ ist folglich die Anzahl der Gerüste des Graphen $G$. Wie wir in Kapitel 6 gesehen haben (vgl. Satz 6.2-1), kann diese Anzahl mit $n$ exponentiell wachsen. Jedoch gibt es gute (polynomiale) Algorithmen, um diese Anzahl zu ermitteln; ein Verfahren ist im Anhang $G$ beschrieben.

Für die Bestimmung von $c$ und $R_c$ ist es nützlich, zunächst das $s,t$-Zuverlässigkeitsproblem zu betrachten.

Beim $s,t$-Zuverlässigkeitsproblem ist offenbar $l$ die Länge eines kürzesten Weges von $s$ nach $t$, $F_l$ also die Anzahl der Wege dieser Länge.

Wir wissen bereits aus Kapitel 6, dass $l$ leicht bestimmt werden kann. Zur Berechnung von $F_l$ erweist es sich als sinnvoll, zunächst allgemein Kantenfolgen (vgl. Kapitel 1) zuzulassen. Dazu bezeichnen wir für eine beliebige Ecke $x$ und $0 \leq i \leq m$ mit

$$K(x,i)$$

die Anzahl der Kantenfolgen aus $i$ vielen Kanten, die bei Knoten $s$ beginnen und bei $x$ enden. (Zur Erinnerung: im Gegensatz zu Wegen dürfen bei Kantenfolgen die einzelnen Kanten mehrfach vorkommen.)

Bezeichnet $N(x)$ die Menge der zu $x$ benachbarten Knoten, so gilt offenbar für $i > 0$ stets

$$K(x,i) \;=\; \sum_{y \in N(x)} K(y, i-1).$$

Diese Gleichung hat als Konsequenz, dass der folgende (polynomiale) Algorithmus alle Werte $K(x,i)$ mit $x \in E$ und $o \leq i \leq m$ bestimmt:

**Algorithmus**:

**begin**
  Setze $K(s,0) := 1$
  und $K(x,0) := 0$ für jede Ecke $x \in E$ mit $x \neq s$;
  **for** $i = 1, ..., m$ **do**
    **for** $x \in E$ **do**
      setze $K(x,i) := \sum_{y \in N(x)} K(y, i-1)$;
**end**

Da nun aber jede Kantenfolge der Länge $l$ von $s$ nach $t$ ein Weg (also auch ein kürzester Weg) sein muss, ist $K(t,l) = F_l$, m. a. W. kann $F_l$ mit dem obigen Algorithmus bestimmt werden.

Eine kritische Kantenmenge im Sinne der $s,t$-Zuverlässigkeit entspricht offenbar genau einer $s$ und $t$ trennenden Kantenmenge, wie sie in Abschnitt 7.4 eingeführt wurde. Die minimale Größe $c$ eines solchen Schnitts kann also mit dem Algorithmus von Ford und Fulkerson bestimmt werden.

Es gibt jedoch kein gutes Verfahren, $R_c(= \binom{m}{c} - C_c)$ zu bestimmen. Hier gibt es wieder das Ergebnis (vgl. [PRO]), dass es sich um ein NP-vollständiges Problem handelt – der Beweis soll an dieser Stelle nicht geführt werden.

Nun kehren wir noch einmal zum vollen Zuverlässigkeitsproblem zurück.

Um die minimale Größe $c$ einer kritischen Kantenmenge zu bestimmen, kann man nun für jede Auswahl von $s$ und $t$ das zugehörige $c_{st}$ wie soeben gezeigt bestimmen, das kleinste aller so erhaltenen $c_{st}$ $(s, t \in E)$ ist das gesuchte $c$. Der Aufwand dieses Vorgehens bleibt polynomial.

Es gibt auch ein effizientes Verfahren, $C_c$ bzw. $R_c$ zu bestimmen (vgl. [BAL]). Da es sich hierbei jedoch um einen recht komplizierten Algorithmus handelt, wollen wir auf eine genaue Beschreibung verzichten.

Wir wollen nun zwei Abschätzungen für das Polynom $z(G)$ ansteuern, wobei die zweite Abschätzung stets genauer ist als die erste. Für die erste Abschätzung sind keinerlei Voraussetzungen über die Art des betrachteten Zuverlässigkeitsproblems (also die Wahl der Menge $\mathcal{F}$) nötig.

Da $R_i = \binom{m}{i}$ für $i < c$ und $R_j = 0$ für $j > m - l$ ist, hat man

$$z(G) = \sum_{i=0}^{c-1} \binom{m}{i} \cdot (1-p)^i \cdot p^{m-i} + \sum_{j=c}^{m-l} R_j \cdot (1-p)^j \cdot p^{m-j}.$$

Zur Vereinfachung der Schreibweise setzen wir im folgenden

$$S = \sum_{i=0}^{c-1} \binom{m}{i} \cdot (1-p)^i \cdot p^{m-i} + R_c \cdot (1-p)^c \cdot p^{m-c} + R_{m-l} \cdot (1-p)^{m-l} \cdot p^l,$$

so dass also gilt

$$z(G) = S + \sum_{j=c+1}^{m-l-1} R_j \cdot (1-p)^j \cdot p^{m-j}.$$

Die erste Abschätzung bekommen wir nun, indem wir mit der Ungleichungskette $0 \leq R_j \leq \binom{m}{j}$ in die obige Gleichung gehen:

**Satz 9.5-1:** *Es gilt stets die folgende einfache Abschätzung für das Zuverlässigkeitspolynom:*

$$S \leq z(G) \leq S + \sum_{j=c+1}^{m-l-1} \binom{m}{j} \cdot (1-p)^j \cdot p^{m-j}.$$

Ein Beweis dieses Satzes erübrigt sich. Die Abschätzung ist eine leichte Abwandlung einer von Van Slyke und Frank angegebenen Fassung (vgl. [VAN]).

Man beachte, dass eine Anwendung dieser Abschätzung die Verfügbarkeit der Zahlen $c,\ R_c,\ l$ und $F_l(= R_{m-l})$ voraussetzt.

**Beispiel 9.5-2:**
Wir betrachten wieder das Beispiel 9.5-1 mit $z(G) = 8p^3(1-p)^2+5p^4(1-p)+p^5$. Wegen $c = 2$ und $l = 3$ ist hier $z(G) = S$, und die obige Abschätzung hat keine weitere Aussagekraft. Ein interessanteres (etwas größeres) Beispiel wird im Anschluss an die Herleitung der zweiten Abschätzung angesehen.

Für das Folgende müssen wir voraussetzen, dass das Hinzufügen von Kanten zu einem funktionsfähigen Zustand stets wieder einen funktionsfähigen Zustand ergibt, dass m. a. W. aus $X \in \mathcal{F}$ und $X \subseteq Y$ stets $Y \in \mathcal{F}$ folgt.

Es sei $\mathcal{F}' := \{K \setminus X | X \in \mathcal{F}\}$ die Menge der Komplemente der funktionsfähigen Zustände. Aufgrund der Definition der Koeffizienten $R_i$ gilt also

$$R_i = |\{X \in \mathcal{F}' \mid |X| = i\}|.$$

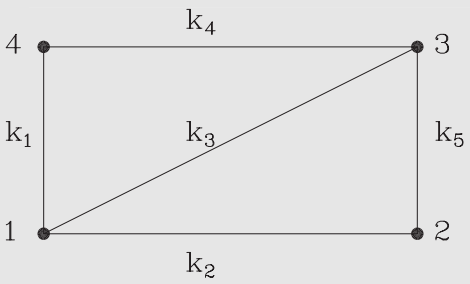
**erbliches Mengensystem**

Die oben gemachte Voraussetzung an $\mathcal{F}$ bedeutet für $\mathcal{F}'$, dass das Mengensystem $\mathcal{F}'$ **erblich** ist, d. h. es gilt:

$$X \in \mathcal{F}', Y \subseteq X \Rightarrow Y \in \mathcal{F}'.$$

Wir müssen an dieser Stelle anmerken, dass die hier eingeführte Voraussetzung an $\mathcal{F}$ bzw. $\mathcal{F}'$ bei allen von uns bisher betrachteten Zuverlässigkeitsproblemen erfüllt ist.

**Beispiel 9.5-3:**
Gegeben sei wieder der folgende Graph:




Wir stellen uns auf dieser Struktur ein Paketnetz vor, indem das Routing nach folgender Regel funktioniert: Knoten $X$ schickt ein für Knoten $Y$ bestimmtes Paket direkt an $Y$, wenn eine Kante zwischen $X$ und $Y$ verfügbar ist; ansonsten schickt $X$ das Paket stets an denjenigen unter seinen verfügbaren Nachbarn, dessen Name durch die kleinste Zahl gegeben ist.

Nun gilt: sollen die funktionsfähigen Zustände das Senden eines Pakets von $2$ nach $4$ erlauben, so gilt

$$\{k_4, k_5\} \in \mathcal{F}, \text{ aber } \{k_2, k_4, k_5\} \notin \mathcal{F}.$$

Man beachte: in der Praxis würde ein vernünftiges Netzprotokoll dafür sorgen, dass Knoten 2 Kenntnis davon erhält, wenn Knoten 1 (der über $k_2$ von 2 aus erreichbar ist) nicht zu 4 "weiterrouten" kann, und 2 würde deshalb $k_2$ dann doch nicht als verfügbar (für Ziel 4) ansehen.

Es wird nun ein allgemeiner kombinatorischer Satz über Mengensysteme gezeigt, der auf die von uns betrachteten $\mathcal{F}$ bzw. $\mathcal{F}'$ unmittelbar angewendet werden kann. Der Satz wurde im Jahre 1928 von E. Sperner veröffentlicht (siehe [SPE]).

**Satz 9.5-2: Satz von Sperner**

**Satz von Sperner**

*Es sei $K$ eine Menge mit $m$ Elementen, $\mathcal{D} \subseteq \mathcal{P}(K)$ sei ein erbliches Mengensystem. Für $0 \leq i \leq m$ sei $R_i := |\{X \in \mathcal{D} \mid \mid X \mid = i\}|$ die Anzahl der $i$-elementigen Elemente von $\mathcal{D}$. Dann gilt*

$$R_{i-1} \geq \frac{i}{m-i+1} R_i \quad \textit{für jedes } 1 \leq i \leq m.$$

**Beweis:** Die Menge $\{X \in \mathcal{D} \mid \mid X \mid = i\}$ sei mit $\mathcal{D}_i$ bezeichnet. Für $i \geq 1$ bilden wir nun $iR_i$ viele (nicht notwendig verschiedene) Elemente von $\mathcal{D}_{i-1}$, indem wir für jedes $S \in \mathcal{D}_i$ jeweils jedes Element aus $S$ herausnehmen – wegen der Erblichkeit muss jede so resultierende $(i-1)$ -elementige Menge Element von $\mathcal{D}_{i-1}$ sein. Umgekehrt kann jedes $T \in \mathcal{D}_{i-1}$ höchstens auf $(m-i+1)$ viele Weisen so aus einem Element $S \in \mathcal{D}_i$ entstehen – jeweils einmal für jedes nicht zu $T$ gehörende Element. Damit folgt die Ungleichung $iR_i \leq (m-i+1)R_{i-1}$.

Der soeben bewiesene Satz kann auf die uns interessierenden Koeffizienten $R_i$ sofort angewendet werden, als Mengensystem $\mathcal{D}$ wird hier die Menge $\mathcal{F}'$ der Komplemente funktionsfähiger Zustände gewählt.

Der Satz liefert bei gegebenem $R_i$ eine untere Schranke für $R_{i-1}$ bzw. bei gegebenem $R_{i-1}$ eine obere Schranke für $R_i$.

Es ist interessant zu bemerken, dass die Ungleichung

$$R_{i-1} \geq \frac{i}{m-i+1} R_i \ ,$$

wie man leicht nachrechnet, zu der folgenden Ungleichung äquivalent ist:

$$\frac{R_{i-1}}{\binom{m}{i-1}} \geq \frac{R_i}{\binom{m}{i}}$$

Dies ist wegen $F_j = R_{m-j}$ und $\binom{m}{m-j} = \binom{m}{j}$ wiederum gleichbedeutend mit

$$\frac{F_j}{\binom{m}{j}} \geq \frac{F_{j-1}}{\binom{m}{j-1}} \ .$$

Die letzte Ungleichung lässt sich so interpretieren: für wachsendes $j$ wächst der Anteil der funktionsfähigen $j$–elementigen Kantemengen an allen $j$ –elementigen Kantenmengen.

Als Konsequenz der Ungleichungen hat man nun, dass in den von uns betrachteten Zuverlässigkeitsproblemen stets gilt

$$\frac{\binom{m}{i}}{\binom{m}{m-l}} R_{m-l} \;\leq\; R_i \;\leq\; \frac{\binom{m}{i}}{\binom{m}{c}} R_c \;\text{ für } c < i < m-l\,.$$

Diese Ungleichungskette nutzen wir aus, um eine verbesserte Abschätzung für das Zuverlässigkeitspolynom zu bekommen:

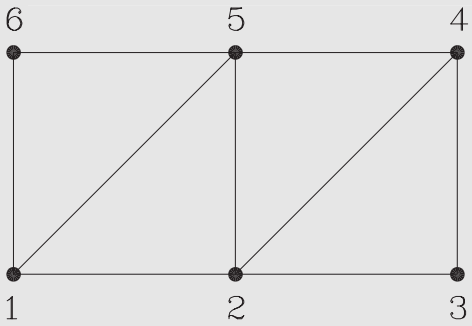
**Satz 9.5-3:** *Es gilt stets:*

$$S+\sum_{i=c+1}^{m-l-1} \frac{\binom{m}{i}}{\binom{m}{m-l}} R_{m-l}(1-p)^i p^{m-i} \leq z(G) \leq S+\sum_{i=c+1}^{m-l-1} \frac{\binom{m}{i}}{\binom{m}{c}} R_c(1-p)^i p^{m-i}$$

Wegen des oben bewiesenen Satzes und der anschließenden Anmerkungen erübrigt sich ein Beweis. Nach [COL] ist diese auf dem Satz von Sperner basierende Abschätzung zum ersten mal in [BAU] angegeben worden. Wir werden im folgenden von den **Sperner-Schranken** sprechen.

**Sperner-Schranken**

**Beispiel 9.5-4:**
Wir betrachten den folgenden Graphen $G$:




G

Es ist $n = 6$ und $m = 9$. Abgeschätzt werden soll das Polynom $z(G)$ für das volle Zuverlässigkeitsproblem.

Zunächst ist leicht zu sehen, dass $c = 2$ und $l = 5$ gilt. Durch kombinatorisches Zählen oder Anwendung des im Anhang $H$ dargestellten Rechenprogramms ermittelt man:

$$\begin{array}{lcl} F_5 = R_4 = 55 \\ F_6 = R_3 = 65 \\ F_7 = R_2 = 34 \\ F_8 = R_1 = 9 \\ F_9 = R_0 = 1 \end{array}$$

Das exakte Polynom lautet also

$$\begin{aligned} z(G) &= p^9 + 9\,(1-p)\,p^2 + 34\,(1-p)^2\,p^7 \\ &\quad + 65(1-p)^3\,p^6 + 55\,(1-p)^4\,p^5 \\ &= S + 65\,(1-p)^3\,p^6\,. \end{aligned}$$

Für die einfachen Schranken ergibt sich hier

$E_u = S$ und $E_o = S + 84(1-p)^3p^6$.

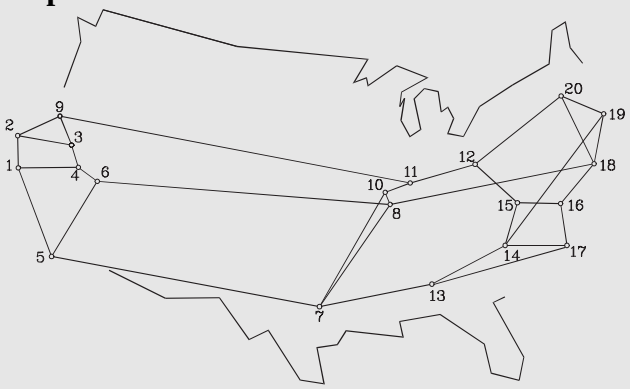
Die Sperner-Schranken liefern die bessere Abschätzung

$S_u = S + 36\frac{2}{3}\,(1-p)^3\,p^6$ und $S_o = S + 79\frac{1}{3}\,(1-p)^3\,p^6\,.$

In der folgenden Tabelle sind für zwei Werte von $p$ die entsprechenden Größen eingetragen:

| $p$ | $E_u$ | $S_u$ | $z(G)$ | $S_o$ | $E_o$ |
|---|---|---|---|---|---|
| 0,9 | 0,940710 | 0,960196 | 0,975253 | 0,982871 | 0,985351 |
| 0,99 | 0,999734 | 0,999768 | 0,999795 | 0,999809 | 0,999813 |

**Beispiel 9.5-5:**




In dem Diagramm ist das sogenannte "Red Arpa" dargestellt. Es handelt sich dabei um ein "Skelett" der 1979er Version des ARPANET, das uns bereits an früherer Stelle begegnet ist. Dieses "Skelett" wird auch in [COL] betrachtet. Es ist $n = 20$ und $m = 32$. Wir wollen auch hier $z(G)$ für das volle Zuverlässigkeitsproblem untersuchen.

Selbstverständlich ist $l = n - 1 = 19$. Auch $c = 3$ ist durch Betrachten des Diagramms leicht zu sehen, jedoch muss man $C_c = 20$ schon mit Hilfe eines Rechenprogramms ermitteln. Gleiches gilt für $F_l = R_{m-l} = 10122705$.

In der folgenden Tabelle ist eine Übersicht der relevanten $R_i$ gegeben, und zwar sind außer den exakten Koeffizienten $R_i$ aus $z(G)$ auch die entsprechenden Werte angegeben, die sich für die Abschätzungen ergeben. Die exakten Werte wurden mit dem in Anhang $H$ aufgelisteten PASCAL-Programm auf einem PC

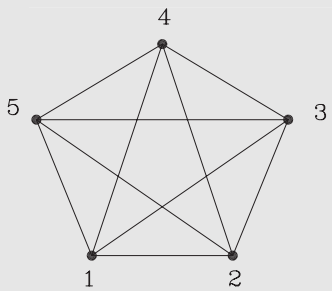
ermittelt. Die Werte für die Abschätzungen können leicht mit dem Taschenrechner ermittelt werden.

Die zweite Tabelle gibt für $p = 0,9$ und $p = 0,99$ die Werte von $z(G)$ und die der entsprechenden Schranken an:

| $R_i$ | $E_u$ | $S_u$ | z(G) | $S_o$ | $E_o$ |
|---|---|---|---|---|---|
| $R_0$ | 1 | 1 | 1 | 1 | 1 |
| $R_1$ | 32 | 32 | 32 | 32 | 32 |
| $R_2$ | 496 | 496 | 496 | 496 | 496 |
| $R_3$ | 4940 | 4940 | 4940 | 4940 | 4940 |
| $R_4$ | 0 | 1048 | 35342 | 35815 | 35960 |
| $R_5$ | 0 | 5868 | 192196 | 200564 | 201376 |
| $R_6$ | 0 | 26407 | 819228 | 902538 | 906192 |
| $R_7$ | 0 | 98083 | 2778207 | 3352284 | 3365856 |
| $R_8$ | 0 | 306510 | 7517243 | 10475888 | 10518300 |
| $R_9$ | 0 | 817361 | 16079317 | 27935700 | 28048800 |
| $R_{10}$ | 0 | 1879931 | 26517778 | 64252110 | 64512240 |
| $R_{11}$ | 0 | 3759862 | 32039959 | 128504220 | 129024480 |
| $R_{12}$ | 0 | 6579758 | 25500420 | 224882385 | 225792840 |
| $R_{13}$ | 10122705 | 10122705 | 10122705 | 10122705 | 10122705 |

| p | $E_u$ | $S_u$ | z(G) | $S_o$ | $E_o$ |
|---|---|---|---|---|---|
| 0,9 | 0,599364 | 0,611012 | 0,976683 | 0,997441 | 0,999053 |
| 0,99 | 0,999698 | 0,999706 | 0,999980 | 0,999984 | 0,999985 |

**Beispiel 9.5-6:**
Auf dem Graphen $G = K_5$ mit $n = 5$ und $m = 10$ sei der dargestellte Routing-Plan mit jeweils zwei Einträgen gegeben:



| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | - | 2;5 | 3;2 | 4;5 | 5;2 |
| 2 | 1;3 | - | 3;1 | 4;3 | 5;1 |
| 3 | 1;2 | 2;4 | - | 4;2 | 5;4 |
| 4 | 1;5 | 2;3 | 3;5 | - | 5;3 |
| 5 | 1;4 | 2;1 | 3;4 | 4;1 | - |

Bei genauerem Hinsehen erkennt man eine Rotationssymetrie hinsichtlich der Nachbarn zweiter Priorität. Z. B. benutzen sich für Zielpunkt $1$ die Punkte $2$ und $3$ und die Punkte $4$ und $5$ gegenseitig als zweite Wahl, für Ziel $2$ lauten die Paarungen $1-5$ und $3-4$ usw.

Wie in früheren Beispielen soll hier ein Zustand funktionsfähig sein, wenn er das Senden einer Nachricht von einem beliebigen Sender zu einem beliebigen Empfänger erlaubt.

Wie beim Routing im ZGS Nr. 7 (vgl. Abschnitt 8.6) soll davon ausgegangen werden, dass aufgrund der Netzprotokolle existierende zulässige Wege auch gefunden werden. Der in Beispiel 9.5-3 aufgezeigte Effekt ist also unmöglich, und die Menge $\mathcal{F}'$ ist ein erbliches Mengensystem. M. a. W. sind die Sperner-Schranken anwendbar.

Die relevanten Parameter sind hier $c = 2$ und $l = 5$, weiter ergeben sich $C_c = 15$ und $F_l = R_{m-l} = 1$.

Die erste Tabelle zeigt zunächst wieder die relevanten $R_i$ :

| $R_i$ | $E_u$ | $S_u$ | $z(G)$ | $S_o$ | $E_o$ |
|---|---|---|---|---|---|
| $R_0$ | 1 | 1 | 1 | 1 | 1 |
| $R_1$ | 10 | 10 | 10 | 10 | 10 |
| $R_2$ | 30 | 30 | 30 | 30 | 30 |
| $R_3$ | 0 | 10/21 | 30 | 80 | 120 |
| $R_4$ | 0 | 5/6 | 10 | 140 | 210 |
| $R_5$ | 1 | 1 | 1 | 1 | 1 |

Die zweite Tabelle gibt für $p = 0,9$ und $p = 0,99$ die Werte von $z(G)$ und die der entsprechenden Schranken an:

| $p$ | $E_u$ | $S_u$ | $z(G)$ | $S_o$ | $E_o$ |
|---|---|---|---|---|---|
| 0,9 | 0,865245 | 0,865517 | 0,880125 | 0,910949 | 0,933801 |
| 0,99 | 0,998502 | 0,998502 | 0,998530 | 0,998578 | 0,998616 |

Beim Aufstellen der einfachen sowie der Sperner-Schranken sind wir unausgesprochen davon ausgegangen, dass $c \leq m - l$ gilt. Jedoch kann auch der Fall $c > m - l$ in diese Schranken mit einbezogen werden, wie folgende Überlegung bestätigt:

**Hilfssatz 9.5-1:** *Ist* $c > m - l$*, so gilt* $c = m - l + 1$ *und*

$$F_l = R_{m-l} = \binom{m}{l}.$$

**Beweis:** Laut Definition von $c$ muss jede $(m - c + 1)$ -elemtige Menge funktionsfähig sein, also gilt $m - c + 1 \geq l$ und folglich im Falle $c > m - l$ bzw. $c \geq m - l + 1$ die Gleichheit $m - c + 1 = l$. Weiter ist jede $l$-elementige Menge funktionsfähig, d. h.

$$F_l = \binom{m}{l}.$$

Konsequenz dieser Überlegung ist: wenn $c > m - l$ bzw. $c = m - l + 1$ gilt, so fallen alle oben aufgeführten Schranken mit

$$z(G) = \sum_{i=0}^{c-1} \binom{m}{i} \cdot (1-p)^{m-i} \cdot p^i + R_c \cdot (1-p)^c \cdot p^{m-c}$$

zusammen.

Es ist noch einmal eine Bemerkung zur Relevanz der oben hergeleiteten Abschätzungen angebracht.

Besonders im Beispiel Beispiel 9.5-5 wurde deutlich, dass die NP-Vollständigkeit des vollen Zuverlässigkeitsproblems sich schon bei recht "kleinen" Graphen in einem großen Rechenaufwand zur Bestimmung des exakten Polynoms $z(G)$ niederschlägt. Die von uns betrachteten Abschätzungen sind allerdings nur dann von praktischem Wert, wenn die Bestimmung von $c, R_c, l$ und $R_{m-l}$ und $R_{m-l}$ (algorithmisch) einfach ist. Wie oben angemerkt, ist diese Situation beim vollen Zuverlässigkeitsproblem gegeben. Die in den Abschätzungen verwendeten Binomialkoeffizienten können rekursiv effizient berechnet werden (Stichwort: "Pascalsches Dreieck").

**Kruskal-Katona-Schranken**
**Ball-Provan-Schranken**

Es gibt eine Reihe von Abschätzungen für die Koeffizienten $R_i$ , die den Sperner-Schranken überlegen sind. Am bekanntesten sind die **Kruskal-Katona-Schranken** und die **Ball-Provan-Schranken**. Die Herleitung dieser Schranken würde allerdings den Rahmen des vorliegenden Kurses sprengen, da hierzu tiefere Ergebnisse aus Kombinatorik und Algebra verwendet werden müssen. Interessierte Leser seien auch hier auf das Buch [COL] verwiesen.

Es gibt eine Reihe weiterer Abschätzungen für den Wert $z(G)(p)$ (nicht für das ganze Polynom $z(G)$), die Voraussetzungen über den Bereich der betrachteten Werte für $p$ machen. Ist z. B. $p$ nahe bei $0$, so ist

$$F_{n-1} \cdot p^{n-1} \cdot (1-p)^{m-n+1}$$

(für das volle Zuverlässigkeitsproblem) eine gute Näherung für $z(G)(p)$. Der Term stellt die Wahrscheinlichkeit dar, dass genau die Kanten eines Gerüsts (und keine

anderen) intakt sind; jeder weitere Summand, der zur Berechnung des exakten $z(G)(p)$ dazukäme, ist um mindestens einen Faktor $p$ kleiner.

Ist $p$ nahe bei $1$ (wie bei den von uns betrachteten Anwendungen), so gilt

$$z(G) \;\approx\; 1 \;-\; C_c \;\cdot\; (1-p)^c \;\cdot\; p^{m-c}\;.$$

Dies wurde zuerst in [KEL] beobachtet. Eine ähnliche Abschätzung werden wir im nächsten Abschnitt genauer betrachen. Die Begründung für die Abschätzung ist: für die Nicht–Funktionsfähigkeit müssen die Kanten eines Schnitts ausfallen; die den Faktor $(1-p)^{c+1}$ enthaltenden Terme werden weggelassen.

**Selbsttestaufgabe 9.5-1:**

*Erläutern Sie, unter welchen Voraussetzungen die Sperner-Schranken zur Abschätzung des Zuverlässigkeitspolynoms anwendbar sind.*

## 9.6 Routing und Zuverlässigkeit

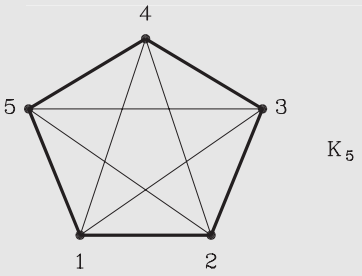
Auf den Zusammenhang von Routing und Zuverlässigkeit wurde bereits mehrfach eingegangen. Sind die Netzprotokolle so ausgelegt, dass periodisch stets die günstigsten Wege zwischen den Knoten errechnet werden (wie z. B. im Internet), so ist die Zuverlässigkeit im wesentlichen (d. h. bis auf durch die Protokolle gegebene Zeitbedingungen) durch den Zusammenhang des Netzes gegeben.

Anders verhält es sich, wenn mit Routing-Tabellen gearbeitet wird. Wie an mehreren Beispielen aufgezeigt wurde, hat das Design dieser Tabellen einen direkten Einfluss auf die Zuverlässigkeit.

Es ist auch stets das Bestreben, unter der Annahme eines homogenen Verkehrsaufkommens seitens der Netzknoten eine möglichst gleichmäßige Auslastung der Knoten und Kanten des Netzes zu erreichen. Dabei liegt es auf der Hand, dass es nicht immer einfach sein wird, diese verschiedenen Ziele in einen akzeptablen Kompromiss zu bringen.

Zur Verdeutlichung des Problemkreises wollen wir einige Beispiele betrachten.

**Beispiel 9.6-1:**
Es sei das vollvermaschte Netz mit 5 Knoten $(K_5)$ gegeben.




Im ersten Fall werde folgendermaßen "geroutet": Knoten $i$ sendet eine für $j$ bestimmte Nachricht auf der direkten Kante $ij$; ist diese Kante nicht verfügbar, so kann $i$ nicht mit $j$ kommunizieren, d. h. es sind keine Ersatzwege vorgesehen. Für das Zuverlässigkeitspolynom gilt hier offensichtlich

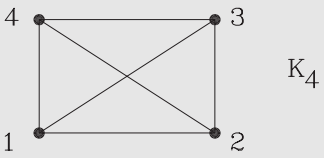
$$z(G) = p^{10}$$

Nun betrachten wir eine zweite Routing-Variante: Knoten $i$ sendet eine für $j$ bestimmte Nachricht stets längs einer in obigem Diagramm fett gezeichneten Kante, und zwar an den im Uhrzeigersinn gelegenen Nachbarn. M. a. W. wird nur ein "Ring" von Kanten ausgenutzt: 4 schickt alle Nachrichten an 3, 3 an 2 usw. Man hat hier

$$z(G) = p^5$$

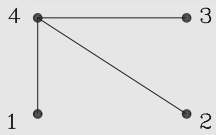
also eine größere Zuverlässigkeit als in dem ersten Routing. Nachteil ist natürlich die hohe Belastung der Kanten des Rings, während die übrigen Kanten nicht genutzt werden; die "Querkanten" brauchten gar nicht zu existieren – es kommt hier die Problematik der Netzwerksynthese mit hinein, die im nächsten Abschnitt ausführlicher untersucht wird. Ein weiterer Nachteil ergibt sich durch die längeren Wege, also die erhöhte Nachrichtenlaufzeit.

**Beispiel 9.6-2:**
Es sei das vollvermaschte Netz mit 4 Knoten $(K_4)$ gegeben.



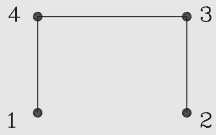
Jeder Knoten soll die Möglichkeit haben, per Broadcasting Routing-Informationen im Netz zu verteilen. Dabei soll jeweils längs der Kanten eines Gerüsts geroutet werden. Es wird nicht mit Flooding gearbeitet, d. h. ein eine Nachricht initiierender Knoten muss individuell für jeden anderen eine Meldung erzeugen.

Man kann nun z. B. einheitlich das folgende Gerüst auswählen:




Eine von 1 initiierte Nachricht führt hier zu 5 Meldungen im Netz, eine von 4 initiierte zu drei Meldungen.

Im Vergleich dazu sei dieses Gerüst ausgewählt:

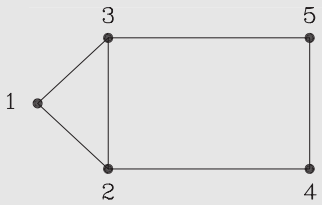



Hier führt eine von 1 initiierte Nachricht zu 6 Meldungen, eine von 4 initiierte zu 4 Meldungen.

Hinsichtlich der Anzahl erzeugter Meldungen ist das erste Gerüst also günstiger, ebenso bezüglich der entstehenden Weglängen. Interessierte Leser seien an dieser Stelle auf [HU] verwiesen, wo die Problematik der Bestimmung eines solchen optimalen Gerüsts in beliebigen Graphen zuerst betrachtet wurde.

Man könnte hier auch jedem einzelnen Knoten ein individuelles Gerüst zuordnen, in dem die zu einer von ihm erzeugten Nachricht gehörenden Meldungen geroutet werden. (Das Routing ist dann nicht mehr rein zielorientiert.) Bei der Auswahl einer solchen Kollektion von Gerüsten verkompliziert sich natürlich wieder die Bestimmung der Netzzuverlässigkeit.

**Beispiel 9.6-3:**
Das im folgenden Diagramm dargestellte Netz sei gegeben.




Es soll nach dem Prinzip geroutet werden, dass für jede Kommunikationsrichtung $i \to j$ (wo $i, j$ verschiedene Knoten sind) eine eindeutige Route $R(i \to j)$ festgelegt wird. So könnte z. B.

$$R(1 \to 4) = 1 \to 2 \to 4$$

sein. In unserem Beispiel sind also $5 \cdot 4 = 20$ solcher Routen festzulegen.

Je nach Festlegung der Menge dieser Routen

$$\mathcal{R} = \{R(i \to j) | i \neq j\}$$

können Knoten verschieden oft als Zwischenknoten in Routen auftreten. Ist $L(x)$ – die Last des Knotens $x$ – die Anzahl der Routen aus $\mathcal{R}$, in denen $x$ als Zwischenknoten fungiert, so ist es (auch im Sinne kurzer Wege) klar, dass das Maximum der $L(x)$ (gebildet über alle Knoten $x$) möglichst klein sein soll. Man ist also interessiert, $\mathcal{R}$ so festzulegen, dass

$$max\{L(x) \mid x \,\text{Knoten}\}$$

minimal wird. Der resultierende Minimalwert wird "node forwarding index" genannt (siehe [MAN]).

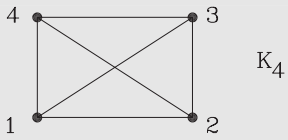
Für unser Beispiel erhält man z. B. ein optimales $\mathcal{R}$ (mit node forwarding index 2), wenn $\mathcal{R}$ folgendermaßen gewählt wird:

für benachbarte $i, j$ sei stets $R(i \to j) = i \to j$; weiter sei:

$$\begin{aligned}
R\,(1 \to 4) &= 1 \to 2 \to 4 \\
R\,(1 \to 5) &= 1 \to 3 \to 5 \\
R\,(2 \to 5) &= 2 \to 4 \to 5 \\
R\,(3 \to 4) &= 3 \to 5 \to 4 \\
R\,(4 \to 1) &= 4 \to 2 \to 1 \\
R\,(4 \to 3) &= 4 \to 5 \to 3 \\
R\,(5 \to 1) &= 5 \to 3 \to 1 \\
R\,(5 \to 2) &= 5 \to 4 \to 2
\end{aligned}$$

Die Literatur beschäftigt sich vor allem damit, zu gegebener Knotenanzahl $n$ ein Netz mit möglichst kleinem node forwarding index zu finden (s. z. B. [MAN]). Interessant wäre es auch hier, den Zusammenhang zur Netzzuverlässigkeit zu untersuchen.

**Beispiel 9.6-4:**
Es sei wieder das vollvermaschte Netz mit 4 Knoten $(K_4)$ gegeben.




Es werden zwei verschiedene Routing-Pläne mit jeweils $\pi = 2$ Einträgen betrachtet (vgl. Abschnitt 8.6), die resultierenden Netze bezeichnen wir mit $\mathcal{N}_1$ und $\mathcal{N}_2$.

$\mathcal{N}_1$ hat folgenden Routing-Plan:

| von \ nach | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | - | 2;3 | 3;2 | 4;2 |
| 2 | 1;3 | - | 3;1 | 4;1 |
| 3 | 1;2 | 2;1 | - | 4;1 |
| 4 | 1;2 | 2;1 | 3;1 | - |

Zu $\mathcal{N}_2$ gehört dieser Routing-Plan:

| von \ nach | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | - | 2;3 | 3;4 | 4;2 |
| 2 | 1;3 | - | 3;4 | 4;1 |
| 3 | 1;2 | 2;4 | - | 4;1 |
| 4 | 1;2 | 2;3 | 3;1 | - |

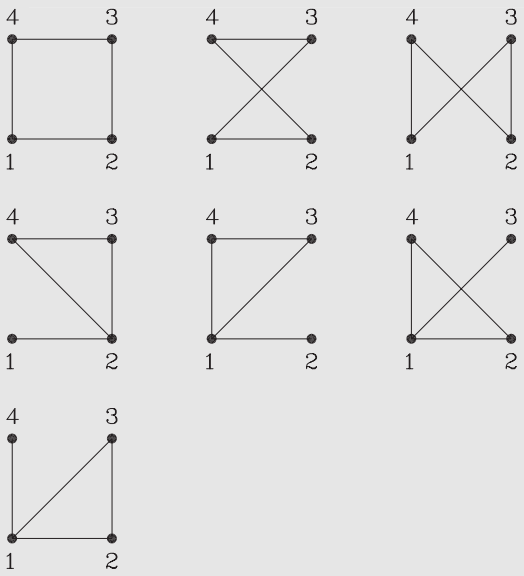
Wie man sieht, handelt es sich bei $\mathcal{N}_1$ um SP-Routing mit Präferenz. $\mathcal{N}_2$ hat eine gewisse Symmetrie: die "Knotenrotation" $1 \to 2 \to 3 \to 4 \to 1$ lässt den Routing-Plan invariant. Für die Zuverlässigkeitspolynome erhält man hier:

$$z(\mathcal{N}_1) = p^3 + 4p^4 - 5p^5 + p^6$$

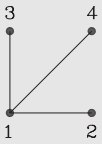
$$z(\mathcal{N}_2) = 3p^4 - 2p^6$$

Dies lässt sich folgendermaßen einsehen.

Bei beiden Netzen sind alle fünfelementigen Kantenmengen funktionsfähig. Für $\mathcal{N}_1$ sind die hier dargestellten Teilgraphen mit vier Kanten funktionsfähig:




Ferner gehört bei $\mathcal{N}_1$ folgendes Gerüst zu einem funktionsfähigen Zustand:




Dies ergibt insgesamt

$$\begin{aligned} z(\mathcal{N}_1) &= p^6 + 6p^5(1-p) + 7p^4(1-p)^2 + p^3(1-p)^3 \\ &= p^3 + 4p^4 - 5p^5 + p^6 \,. \end{aligned}$$

Das Netz $\mathcal{N}_2$ hat einige weniger funktionsfähige Zustände und ist somit weniger zuverlässig: Nur die ersten drei der obigen vierelementigen Kantenmengen sind funktionsfähig, und es gibt kein funktionsfähiges Gerüst. Dies führt zu

$$\begin{aligned} z(\mathcal{N}_2) &= p^6 + 6p^5(1-p) + 3p^4(1-p)^2 \\ &= 3p^4 - 2p^6 \,. \end{aligned}$$

Obwohl zuverlässiger als $\mathcal{N}_2$, hat $\mathcal{N}_1$ den offensichtlichen Nachteil einer ungleichmäßigen Auslastung, die desto mehr zum Tragen kommt, je größer die Kantenausfallwahrscheinlichkeit $q = 1 - p$ ist.

In der Literatur findet man auch Untersuchungen zum Verhältnis von Routing und Zuverlässigkeit, bei denen Überlegungen zu den Kommunikationsprotokollen sehr stark einbezogen sind. Als Beispiele seien [AWE] und [GAF] genannt. In [AWE]

geht es um den Entwurf absolut zuverlässiger Broadcast-Protokolle für Netze mit veränderlicher Topologie. In [GAF] werden verteilte Algorithmen vorgeschlagen, die die einzelnen Knoten befähigen sollen, ohne Kenntnis der aktuellen gesamten Netztopologie Nachrichten auf kreisfreien Routen an eine zentrale Station zu versenden.

Wir wollen diese Art von Untersuchungen hier nicht weiter verfolgen. Für den Rest dieses Abschnitts soll es vielmehr darum gehen, einige erste Ergebnisse zum Zusammenhang von Routing und Zuverlässigkeit für Netze mit Routing-Plänen herauszuarbeiten. Das zuletzt betrachtete obige Beispiel 9.6-4 weist dabei grob in die einzuschlagende Richtung. Die Ergebnisse sind sämtlich der Untersuchung [POG1] entnommen, einige wurden in [POG2] veröffentlicht.

Im folgenden werden zu Graphen $G$ Routing-Pläne $C$ mit jeweils zwei Einträgen (also mit $\pi = 2$, vgl. Abschnitt 8.6) betrachtet. Da für einen Graphen mehrere Routing-Pläne möglich sind, sprechen wir bei gegebenem $G$ und $C$ auch hier (wie bereits in Beispiel 9.6-4) von einem **Netz** $\mathcal{N} = (G, C)$. Das zugehörige Zuverlässigkeitspolynom wird dementsprechend mit

**Netz**

$$z(\mathcal{N}) \; bzw. \; z(\mathcal{N})(p)$$

bezeichnet. Wie bisher soll dabei eine Kantenmenge funktionsfähig sein, wenn sie (im Falle dass jede ihrer Kanten intakt ist) die Kommunikation von jedem Ursprungs- zu jedem Zielknoten erlaubt.

Wir gehen auch wieder von den Grundannahmen bezüglich der Netzprotokolle aus, wie sie in Abschnitt 8.6 – abgeleitet von der Situation beim Zeichengabesystem Nr. 7 – dargestellt wurden. Insbesondere müssen bei einem kreisfreien Routing-Plan gerichtete Kreise auf einer Kante nicht ausgeschlossen werden, da das Pendeln von Nachrichten auf einer Kante von den Protokollen verhindert wird.

Wir erinnern an die in Abschnitt 9.5 eingeführten Parameter $l$ und $c$: für ein gegebenes Netz $\mathcal{N} = (G, C)$ sei
$l$ die minimale Größe einer funktionsfähigen Kantenmenge,
$c$ die minimale Größe einer kritischen Kantenmenge.

Die in Abschnitt 9.5 hergeleiteten Abschätzungen für das Zuverlässigkeitspolynom basieren auf der Kenntnis von $l, c, R_{m-l} = F_l$ und $R_c = \binom{m}{c} - C_c$.

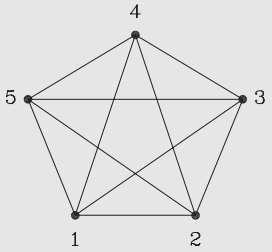
Es ist unmittelbar klar, dass aufgrund unserer generellen Voraussetzung von jeweils zwei Einträgen in den Routing-Plänen für ein Netz $\mathcal{N} = (G, C)$ stets $c = 2$ gilt. Dies liegt daran, dass bei Ausfall von nur einer Kante das Netz stets funktionsfähig bleibt, andererseits aber die beiden Kanten, die einem Knoten für ein Ziel zur Verfügung stehen, immer eine kritische Menge bilden.

An dieser Stelle ist eine Klarstellung zur Terminologie nötig. Wie üblich, nennen wir eine Kantenmenge $T \subseteq K$ einen **minimale kritische Menge**, wenn $T$ kritische Menge ist, dies jedoch für keine echte Teilmenge von $T$ gilt.

**minimale kritische Menge**

Ein kritische Menge $T \subseteq K$ mit $c$ vielen Elementen muss selbstverständlich minimal sein. Jedoch kann umgekehrt ein minimale kritische Menge aus mehr als $c$ vielen Kanten bestehen.

**Beispiel 9.6-5:**
Gegeben sei $G = K_5$ mit Routing-Plan $C$.

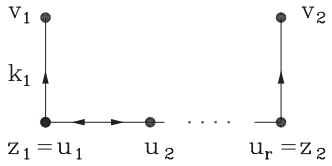



| von \ nach | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | - | 2;3 | 3;2 | 4;2 | 5;2 |
| 2 | 1;3 | - | 3;1 | 4;1 | 5;3 |
| 3 | 1;2 | 2;1 | - | 4;1 | 5;4 |
| 4 | 1;2 | 2;1 | 3;1 | - | 5;2 |
| 5 | 1;2 | 2;1 | 3;1 | 4;1 | - |

Es handelt sich um SP-Routing mit Präferenz, wobei jedoch in den Knoten 2,3 und 4 die Zweiteinträge für Ziel 5 so geändert wurden, dass in $G(\rightarrow 5)$ ein Kreis $2 \rightarrow 3 \rightarrow 4 \rightarrow 2$ entsteht. Dies führt dazu, dass die Kantenmenge $\{25, 35, 45\}$ kritisch und – wie leicht nachzuprüfen – sogar minimal ist, denn bei Ausfall von nur 2 dieser 3 Kanten bleibt das Netz funktionsfähig.

Interessanterweise kann nachgewiesen werden, dass unter der Voraussetzung der Kreisfreiheit jeder minimale kritische Menge aus 2 Kanten besteht.

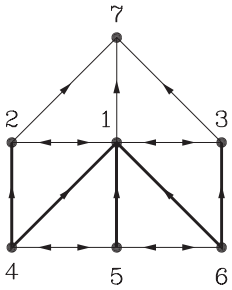
**Satz 9.6-1:** *In einem beliebigen Netz $\mathcal{N} = (G, C)$ mit kreisfreiem Routing-Plan $C$ (mit jeweils $\pi = 2$ Einträgen) besteht jede minimale kritische Menge aus zwei Kanten. Genauer gilt: ist $T = \{k_1, k_2\}$ eine minimale kritische Menge, so gibt es einen Knoten $y$ und eine Orientierung $k_1 = z_1v_1$ und $k_2 = z_2v_2$ der Kanten von $T$ sowie Knoten $z_1 = u_1, u_2, ..., u_r = z_2 (r \geq 1)$, so dass in dem Graphen $G(\rightarrow y)$ der in folgendem Diagramm dargestellte Teilgraph enthalten ist (wobei für $r \geq 2$ gelten kann $v_1 = v_2$):*




**Beweis:** Es sei $T$ eine minimale kritische Menge. $x$ und $y$ seien Knoten derart, dass der Ausfall von $T$ das Versenden von Nachrichten von $x$ nach $y$ unmöglich macht. Wir setzen voraus, dass innerhalb des gerichteten Graphen $G(\rightarrow y)$ der Knoten $x$ unter den Knoten mit dieser Eigenschaft ein solcher ist, für den der gerichtete Abstand von $x$ nach $y$ minimal ist. Mindestens einer der beiden Nachfolger von $x$ in $G(\rightarrow y)$, sagen wir $u$, hat einen kleineren gerichteten Abstand zu $y$. Daraus folgt $xu \in T$. $v$ sei der zweite Nachfolger von $x$ in $G(\rightarrow y)$. Ist $xv \in T$, so folgt wegen der Minimalität von $T$ sofort $T = \{xu, xv\}$, und die Aussage des Satzes trifft zu (mit $r = 1$).

Es sei nun $xv \notin T$ angenommen, $w_1$ und $w_2$ seien die Nachfolger von $v$ in $G(\rightarrow y)$. Aufgrund der Kreisfreiheit, und da jeder gerichtete Weg von $v$ nach $y$ auf die minimale Menge $T$ treffen muss, folgt o. B. d. A. $w_1 = x$. Gilt nun $vw_2 \in T$, so ist der Beweis erbracht mit $T = \{xu, vw_2\}$ (und $r = 2$). Andernfalls liefert die Iteration dieser Argumentationskette (mit $w_2$ anstelle $v$ usw.) den angestrebten Schluss nach endlich vielen Schritten.

Ein entsprechender Satz ist nicht mehr gültig, wenn für die Anzahl der möglichen Einträge gilt $\pi \geq 3$. Im folgenden Diagramm ist ein möglicher kreisfreier Digraph $G(\rightarrow 7)$ für $G$ mit Knoten $\{1, 2, ..., 7\}$ dargestellt, wobei jeder von 7 verschiedene Knoten 2 oder 3 Nachfolger besitzt:




Die fünf fett herausgehobenen Kanten bilden einene minimale kritische Menge $T_5$, denn der Routing-Plan lässt sich auch für die übrigen Zielknoten so vervollständigen, dass bei Ausfall einer echten Teilmenge von $T_5$ immer noch jede Kommunikation möglich ist. Auf der anderen Seite ist selbstverständlich, da es in $G(\rightarrow 7)$ Knoten mit nur 2 Nachfolgern gibt, wieder $c = 2$.

Wie oben bereits festgestellt, gilt in dem von uns vorwiegend betrachteten Fall $\pi = 2$ stets $c = 2$. Der Satz besagt, dass im Falle der Kreisfreiheit die Anzahl $C_2$ der zweielementigen kritischen Menge auch gleichzeitig die Anzahl aller minimalen kritischen Mengen ist.

Die Zahl $C_2$ kann mit polynomialem Aufwand bestimmt werden: besitzt $G$ $m$ viele Kanten, so muss jede der $\binom{m}{2}$ vielen zweielementigen Kantenmengen daraufhin überprüft werden, ob bei Ausfall dieser beiden Kanten immer noch in jedem Digraphen $G(\rightarrow x)$ ein gerichteter Weg von einem beliebigen $y \neq x$ nach $x$ intakt ist oder nicht.

**Beispiel 9.6-6:**
Wir schauen noch einmal auf die Netze $\mathcal{N}_1$ und $\mathcal{N}_2$ aus Beispiel 9.6-4

Für $\mathcal{N}_1$ ist $F_4 = 7$, folglich $C_2 = \binom{6}{2} - F_4 = 8$.

Da für $\mathcal{N}_2$ gilt $F_4 = 3$, hat man hier $C_2 = 12$.

Um die in Abschnitt 9.5 angeführten Abschätzungen für das Zuverlässigkeitspolynom auf beliebige Netze $\mathcal{N} = (G, C)$ (mit $\pi = 2$) anwenden zu können, bedürfte es eines guten Algorithmus (wie üblich gemessen an der Knotenanzahl von $G$), auch die Parameter $l$ und $F_l$ zu bestimmen. Ein solcher Algorithmus wurde jedoch unserer Wissens bisher nicht gefunden. Vielmehr vermuten wir, dass schon die Bestimmung von $l$, also der kleinstmöglichen Elementeanzahl einer funktionsfähigen Kantenmenge, ein NP-vollständiges Problem bildet.

Wir greifen zunächst auf die obere Abschätzung für das Polynom $z(\mathcal{N})$ zurück, die am Ende von Abschnitt 9.5 angesprochen wurde und für nahe bei 1 gelegene Kantenwahrscheinlichkeit $p$ eine recht gute Näherung für $z(\mathcal{N})(p)$ darstellt:

$$z(\mathcal{N})(p) \leq 1 - C_2 \cdot q^2 \cdot p^{m-2} =: \hat{z}(\mathcal{N})(p).$$

Dabei ist $m$ die Anzahl der Kanten des Graphen und wie üblich $q = 1 - p$.

Da wir jedoch in diesem Abschnitt nicht nur an guten Abschätzungen interessiert sind, sondern vor allem Erkenntnisse über den Einfluss des Designs von Routing-Plänen auf die Zuverlässigkeit gewinnen wollen, erweist sich die nähere Betrachtung einer der obigen Näherung sehr ähnlichen unteren Abschätzung als sinnvoll:

**Satz 9.6-2:** *Für beliebiges $p$ gilt*

$$\check{z}(\mathcal{N})(p) := 1 - C_2 \cdot q^2 \leq z(\mathcal{N})(p).$$

**Beweis:** Es seien $T_1, T_2, ..., T_{C_2}$ die (zweielementigen) minimalen kritischen Mengen. $E_i$ bezeichne das Ereignis "Beide Kanten von $T_i$ fallen aus". Es gilt $P(E_i) = q^2$. Ferner ist

$$1 - z(\mathcal{N})(p) = P(E_1 \vee E_2 \vee ... \vee E_{c_2}).$$

Das Prinzip der Inklusion-Exklusion (vgl. Anhang F) liefert sofort die obige Ungleichung.

**Beispiel 9.6-7:**
In Fortsetzung von Beispiel 9.6-4 und Beispiel 9.6-6 betrachten wir nochmals das Netz $\mathcal{N}_1$. Es gilt für jedes $p(0 \leq p \leq 1)$ und $q = 1 - p$:

$$\check{z}(\mathcal{N}_1) \leq z(\mathcal{N}_1) \leq \hat{z}(\mathcal{N}_1)$$

mit $\check{z}(\mathcal{N}_1) = 1 - 8q^2$,
$z(\mathcal{N}_1) = p^3 + 4p^4 - 5p^5 + p^6$,
$\hat{z}(\mathcal{N}_1) = 1 - 8q^2p^4$.

Zum Vergleich sind für $p = 0,9$, und $p = 0,99$ und $p = 0,9999$ jeweils die drei Werte zusammengestellt. Zur Heraushebung der Unterschiede wird hier mit 12stelliger Genauigkeit gearbeitet.

| $p$ | $\check{z}$ | $z$ | $\hat{z}$ |
|---|---|---|---|
| 0,9 | 0,92 | 0,932391 | 0,947512 |
| 0,99 | 0,9992 | 0,999231 | 0,999232 |
| 0,9999 | 0,99999992 | 0,999999920013 | 0,99999992 |

Wie man an dem obigen Beispiel sieht, ist die einfache untere Abschätzung $\check{z}$ für nahe an 1 gelegenes $p$ bereits eine gute Näherung für $z$.

Es ist leicht möglich, zur Güte der Näherung genauere Aussagen zu machen. Dazu seien für ein Netz $\mathcal{N}$ zunächst einige Bezeichnungen eingeführt. $\mathcal{T}$ sei die Menge der minimalen kritischen Kantenmengen. Da jedes $T \in \mathcal{T}$ aus zwei Kanten besteht, ist für $T, T' \in \mathcal{T}$ stets $T \cup T'$ 3- oder 4-elementig.

Es sei

$$\mathcal{D} := \{\{T, T'\} | \mid T \cup T' \mid = 3\} \text{ und}$$

$$\mathcal{V} := \{\{T, T'\} | \mid T \cup T' \mid = 4\}.$$

Aufgrund der in Anhang F gemachten Anmerkungen über das Prinzip der Inklusion-Exklusion folgt nun

$$\check{z} = 1 - C_2q^2 \leq z \leq 1 - C_2q^2 + \mid \mathcal{D} \mid q^3 + \mid \mathcal{V} \mid q^4,$$

wobei selbstverständlich $\mid \mathcal{D} \mid + \mid \mathcal{V} \mid = \binom{C_2}{2}$ ist.

Mit der noch etwas gröberen oberen Abschätzung

$$1 - C_2q^2 + \binom{C_2}{2}q^3$$

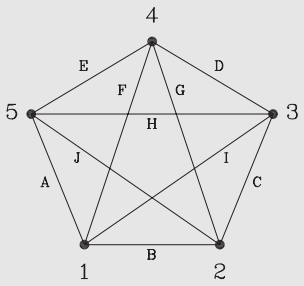
lässt sich nun für gegebene Knotenanzahl $n$ angeben, wie nahe $q$ an $0$ liegen muss, damit $\check{z}$ mit einer vorgegebenen Genauigkeit mit $z$ übereinstimmt. Auf die Darstellung der genauen Rechenschritte wird hier verzichtet.

Wir wollen unser Augenmerk vielmehr auf den Ausdruck

$$\tilde{z} = 1 - C_2 q^2 + \mid \mathcal{D} \mid q^3$$

richten, welcher zwar nicht mehr allgemein als obere Schranke für $z$ nachgewiesen werden kann, jedoch für kleines $q$ eine noch bessere Abschätzung für $z$ als $\tilde{z}$ darstellt.

**Beispiel 9.6-8:**
Wir betrachten den Graphen $K_5$ mit dem Routing-Plan, der bereits in Beispiel 9.5-6 von uns behandelt wurde. Das resultierende Netz bezeichnen wir mit $\mathcal{N}_3$.




| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | - | 2;5 | 3;2 | 4;5 | 5;2 |
| 2 | 1;3 | - | 3;1 | 4;3 | 5;1 |
| 3 | 1;2 | 2;4 | - | 4;2 | 5;4 |
| 4 | 1;5 | 2;3 | 3;5 | - | 5;3 |
| 5 | 1;4 | 2;1 | 3;4 | 4;1 | - |

Wie in Beispiel 9.5-6 dargestellt, lautet das exakte Polynom

$$z(\mathcal{N}_3) = p^{10} + 10p^9(1-p) + 30p^8(1-p)^2 + 30p^7(1-p)^3 + 10p^6(1-p)^4 + p^5(1-p)^5,$$

eine andere Darstellung ist

$$z(\mathcal{N}_3) = 5p^9 - 10p^8 + 5p^6 + p^5.$$

Mit $C_2 = 15$ hat man

$$\tilde{z} = 1 - 15q^2.$$

Die 15 minimalen kritischen Mengen sind
$\{B,C\}$,$\{C,I\}$,$\{B,I\}$,$\{A,E\}$,$\{E,F\}$,$\{A,F\}$,$\{A,B\}$,$\{A,J\}$,
$\{B,J\}$,$\{C,D\}$,$\{D,G\}$,$\{C,G\}$,$\{D,E\}$,$\{E,H\}$,$\{D,H\}$.

Man stellt fest, dass von den $\binom{15}{2} = 105$ möglichen Paaren minimaler Schnitte 35 eine gemeinsame Kante haben, also $| \mathcal{D} | = 35, | \mathcal{V} | = 80$ und

$$\tilde{z} = 1 - 15q^2 + 35q^3.$$

Für $p = 0,9999$ ergeben sich (mit 12stelliger Genauigkeit) die Werte

$$\check{z} = 0,999999850000,$$

$$z = 0,999999850030,$$

$$\tilde{z} = 0,999999850035.$$

Wir gehen in diesem Beispiel noch einen Schritt weiter. Das Prinzip der Inklusion-Exklusion legt nahe, von $\tilde{z}$ den Term

$$t \cdot q^3$$

abzuziehen, wobei $t$ die Anzahl der Dreiermengen minimaler Schnitte ist, deren Vereinigungsmenge nur drei Kanten enthält. Hier ist $t = 5$, denn diese Dreiermengen sind

$$\{\{B, C\}, \{C, I\}, \{B, I\}\},$$

$$\{\{A, E\}, \{E, F\}, \{A, F\}\},$$

$$\{\{A, B\}, \{B, J\}, \{A, J\}\},$$

$$\{\{C, D\}, \{D, G\}, \{C, G\}\},$$

$$\{\{D, E\}, \{E, H\}, \{D, H\}\}.$$

Man erhält so die Näherung

$$1 - 15q^2 + 30q^3,$$

die für $p = 0,9999$ (auf 12 Stellen) in der Tat mit dem exakten $z$ übereinstimmt.

**Beispiel 9.6-9:**
SP-Routing mit Präferenz bedeutet auf $K_5$, dass folgender Routing-Plan gegeben ist:

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | - | 2;3 | 3;2 | 4;2 | 5;2 |
| 2 | 1;3 | - | 3;1 | 4;1 | 5;1 |
| 3 | 1;2 | 2;1 | - | 4;1 | 5;1 |
| 4 | 1;2 | 2;1 | 3;1 | - | 5;1 |
| 5 | 1;2 | 2;1 | 3;1 | 4;1 | - |

Für das so gegebene Netz $\mathcal{N}_4$ gilt interessanterweise wie für $\mathcal{N}_3$ (s. voriges Beispiel) ebenfalls

$$C_2 = 15,$$

jedoch hat man hier $\mid \mathcal{D} \mid = 38$.

Unter der Prämisse, die Netze nach den Funktionen

$$\tilde{z} = 1 - C_2 q^2 + \mid \mathcal{D} \mid q^3$$

zu beurteilen, ist somit $\mathcal{N}_4$ zuverlässiger als $\mathcal{N}_3$.

Tatsächlich gilt für das exakte Polynom

$$z(\mathcal{N}_4) = p^4 + 4p^6 - p^7 - 6p^8 + 3p^9,$$

für $p = 0,999$ ergibt sich damit $z = 0,999999850035$.

In den beiden vorausgegangenen Beispielen zeigt sich eine Tendenz, die sich in folgender Aussage zusammenfassen lässt: sind auf einem Graphen zwei Netze gegeben mit gleicher Anzahl minimaler kritischer Mengen, so ist dasjenige Netz zuverlässiger, bei dem es häufiger der Fall ist, dass ein Paar minimaler kritischer Mengen eine Kante gemeinsam hat.

Die höhere Zuverlässigkeit von $\mathcal{N}_4$ gegenüber $\mathcal{N}_3$ geht einher mit einer ungleichmäßigen Auslastung der Knoten und Kanten. Es soll nun versucht werden, diesen Zusammenhang etwas genauer unter die Lupe zu nehmen.

Zunächst wird folgende Bezeichnung eingeführt. Für ein Netz $\mathcal{N} = (G, C)$ mit $G = (E, K)$ und $\mid K \mid = m$ sei für $k \in K$ $c_k$ die Anzahl der minimalen kritischen Mengen $T$ mit $k \in T$.

**Hilfssatz 9.6-1:** *Es gelten stets die folgenden Beziehungen:*

*a.* $C_2 = \sum\limits_{k \in K} \frac{c_k}{2}$

*b.* $\mid \mathcal{D} \mid = \sum\limits_{k \in K} \binom{c_k}{2}$

*c.* $C_2 \geq m$

**Beweis:**

a. Da jede minimale kritische Menge aus 2 Kanten besteht, muss für die Anzahl $C_2$ der minimalen kritischen Mengen gelten $2C_2 = \sum\limits_{k \in K} c_k$.

b. Offenbar ist jede Kante $k$ auf genau $\binom{c_k}{2}$ Weisen als gemeinsame Kante in zwei minimalen kritischen Mengen enthalten.

c. Da eine Kante stets in mindestens 2 minimalen kritischen Mengen enthalten ist, folgt $2C_2 \geq 2m$ bzw. $C_2 \geq m$.

Die Aussage c) des obigen Hilfssatzes hat eine überraschende Konsquenz hinsichtlich eines Vergleichs der Graphen $R_n$ ("Ring" mit $n$ Knoten und $m = n$ vielen Kanten) und $K_n$ (vollständiger Graph mit $n$ Knoten und $m = \binom{n}{2}$ vielen Kanten):

Auf $R_n$ gibt es offenbar nur einen möglichen Routing-Plan (bis auf die Wahl der Prioritäten), für diesen gilt

$$C_2 = \binom{n}{2}.$$

Andererseits muss nun aber für **jeden** Routing-Plan auf dem Graphen $K_n$ gelten

$$C_2 \geq m = \binom{n}{2}.$$

Unter der Prämisse, dass

$$1 - C_2 q^2$$

eine gute Näherung für die Zuverlässigkeit ist, bedeutet dies: ein auf $K_n$ basierendes Netz kann nicht zuverlässiger sein als das Netz auf $R_n$. (Bei dieser Aussage ist das Ausmaß der Kantenbelastungen allerdings ausgeklammert.)

Selbstverständlich kann es zuverlässigere Netze auf Graphen geben, die "zwischen" $R_n$ und $K_n$ liegen. Solche Graphen aufzuspüren, ist ein typisches Problem der Netzwerksynthese. Diese Thematik ist Inhalt des nächsten Abschnitts.

Wir kommen nun auf den Vergleich der oben behandelten Netze $\mathcal{N}_3$ und $\mathcal{N}_4$ zurück, indem wir die Gleichungen a) und b) des Hilfssatzes ausnutzen. Es wird gezeigt, dass für konstantes $C_2 = \sum\limits_{k \in K} \frac{c_k}{2}$ die Varianz der Zahlen $c_k (k \in K)$ zur Größe von

$$\mid \mathcal{D} \mid = \sum_{k \in K} \binom{c_k}{2}$$

positiv korreliert ist.

**Hilfssatz 9.6-2:** *Für ein Netz $\mathcal{N} = (G, C)$ gilt:*

*i. Der Mittelwert der $c_k (k \in K)$ ist $\bar{c} = \frac{2C_2}{m}$.*

*ii. Für die Standardabweichung $\sigma$ von diesem Mittelwert gilt*

$$\sigma^2 = \bar{c} - \bar{c}^2 + \frac{2 \mid \mathcal{D} \mid}{m}.$$

**Beweis:**

i. ergibt sich sofort aus der Definition des Mittelwerts.

ii. Zunächst ist

$$\sigma^2 = \frac{\sum\limits_{k \in K} (c_k - \bar{c})^2}{m}.$$

Daraus ergibt sich

$$\begin{aligned} m\sigma^2 &= \sum_{k \in K} (c_k^2 - 2c_k\bar{c} + \bar{c}^2) \\ &= \sum_{k \in K} \left( c_k^2 - \tfrac{4C_2 c_k}{m} + \tfrac{4C_2^2}{m^2} \right) \\ &= \tfrac{4C_2^2}{m} + \sum_{k \in K} (c_k^2 - c_k) + \sum_{k \in K} \left( c_k - \tfrac{4C_2 c_k}{m} \right) \\ &= m\bar{c}^2 + 2 \mid \mathcal{D} \mid + 2C_2 \left( 1 - \tfrac{4C_2}{m} \right) \\ &= m\bar{c}^2 + 2 \mid \mathcal{D} \mid + m\bar{c} - 2m\bar{c}^2 \end{aligned}$$

und damit die Behauptung.

Aus dem Hilfssatz folgt: hat man (wie im Falle von $\mathcal{N}_3$ und $\mathcal{N}_4$) zwei Routing-Pläne auf demselben Graphen mit identischen $C_2$ (und $\bar{c}$), so ist das Netz mit einer größeren Varianz der Zahlen $c_k (k \in K)$ zuverlässiger. Andererseits hat diese größere Varianz auch eine höhere Unterschiedlichkeit der Kantenbelastungen zur Folge. Diese zuletzt gemachte Aussage wollen wir jedoch nicht formalisieren und die Ausführungen an dieser Stelle abbrechen. Wir kommen hier in einen Bereich, in dem viele weitere interessante Fragestellungen bisher nicht weiter verfolgt wurden.

**Selbsttestaufgabe 9.6-1:**

*Es sei $\mathcal{N}$ ein Netz auf einem Graphen mit $n$ Knoten, $t$ eine natürliche Zahl. Zeigen Sie: Ist $q = 1 - p \leq (128 \cdot n^{-8} \cdot 10^{-t})^{\frac{1}{3}}$, so gilt*

$$z(\cdot\mathcal{N})(p) - \check{z}(\mathcal{N})(p) < 10^{-t}.$$

## 9.7 Synthese extremaler Netze

Bisher wurde das Problem betrachtet, zu vorgegebener Graphenstruktur und darauf definiertem Netz für ein Zuverlässigkeitsmaß das zugehörige Zuverlässkeitspolynom zu bestimmen. Erwies sich dieses Problem als algorithmisch zu schwierig, so wurde das Augenmerk auf möglichst gute und leicht berechenbare Abschätzungen gerichtet.

Nur gelegentlich kam am Rande der Aspekt der Netzsynthese ins Spiel. Darunter soll die Frage verstanden werden, unter gewissen Randbedingungen (z. B. gegebener Anzahl von Knoten) einen Graphen zu finden, der hinsichtlich eines gegebenen Kriteriums extremal ist.

Wir wollen hier nur das Problem betrachten, optimale Graphen in bezug auf die volle Zuverlässigkeit zu finden. Funktionsfähige Kantenmengen mögen also in diesem gesamten Abschnitt den zusammenhängenden Teilgraphen entsprechen.

Eine mögliche Fragestellung ist nun die folgende.

Es sei $\mathcal{G}(n,m)$ die Klasse aller zusammenhängenden Graphen mit $n$ Ecken und $m$ Kanten. $G_{opt} \in \mathcal{G}(n,m)$ heißt **optimal**, wenn $G_{opt}$ für jede Kantenwahrscheinlichkeit $p(0 \leq p \leq 1)$ zuverlässiger ist als jeder andere Graph in $\mathcal{G}(n,m)$, wenn also gilt: für alle $G \in \mathcal{G}(n,m)$ und $p \in [0,1]$ gilt

**optimal**

$$z(G_{opt})(p) \geq z(G)(p).$$

**Problem:** Man finde zu gegebenen $n$ und $m$ einen optimalen Graphen in $\mathcal{G}(n,m)$.

Dieses Problem ist bisher nicht allgemein befriedigend gelöst. Einen tieferen Einstieg in die Thematik kann man sich verschaffen durch das Studium z. B. der Arbeiten [BAU], [BOE1] oder [BOE2] , auch [MIN] bietet eine gute Übersicht. Gemeinsam ist allen diesen Untersuchungen der rein kombinatorisch graphentheoretische Ansatz, wie er auch von uns hier verfolgt wird.

Der Vollständigkeit halber soll erwähnt werden, dass es auch Untersuchungen zur Netzsynthese gibt, die sich der Problematik aus der Richtung des Operations Research nähern und Kantenkapazitäten, Verzögerungszeiten usw. mit einbeziehen. Die Lösungswege bedienen sich dann eher der "klassischen" mathematischen Optimierung.

Bei obiger Problemstellung ist zu beachten: Zu festem $p_0 \in [0,1]$ muss es selbstverständlich einen Graphen aus $\mathcal{G}(n,m)$ geben, der für dieses $p_0$ optimal ist, jedoch ist die Existenz eines $G_{opt}$ im obigen Sinne (also simultan für alle $p$) keineswegs klar. Man halte sich dazu vor Augen, dass sich die Polynome zweier Graphen "kreuzen" können (vgl. Abschnitt 9.4)!

Um uns dem Problem noch einen Schritt weiter anzunähern, erinnern wir uns zunächst an die Abschätzung

$$z(G)(p) \approx 1 - C_c \cdot (1-p)^c \cdot p^{m-c},$$

die für sehr nahe an 1 gelegenes $p$ gültig ist (vgl. Abschnitt 9.5). Dadurch wird es nahegelegt, nach solchen Graphen zu suchen, für die $c$ möglichst groß ist (dies macht $(1-p)^c \cdot p^{m-c}$ klein) und dann – nachdem $c$ festliegt – $C_c$ zu minimieren.

Statt nach einem Graphen $G_{opt}$ im Sinne der obigen Definition zu suchen, kann man nun das bescheidenere Ziel verfolgen, einen Graphen aus $\mathcal{G}(n,m)$ zu finden, der unter den Graphen aus dieser Klasse, die $c$ maximieren, $C_c$ minimiert. Einen solchen Graphen wollen wir $c-C_c$**-optimal** nennen.

$c-C_c$**-optimal**

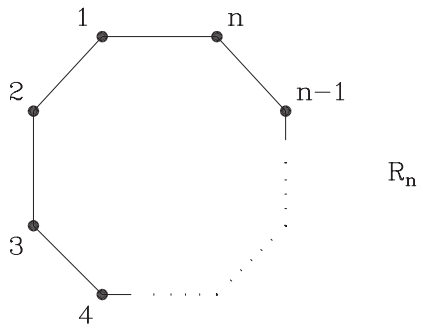
Einige der oben zitierten Arbeiten beschäftigen sich mit dem Aufspüren $c-C_c$-optimaler Graphen (für gegebene $n$ und $m$). Für die erzielten Teilergebnisse ist ein immenser kombinatorischer Aufwand notwendig, umfassend hat man das Problem nicht gelöst.

Wir wollen hier nur die beiden einfachsten Fälle für $m$ in Relation zu $n$ betrachten.

Ist $m = n-1$, so ist $\mathcal{G}(n,n-1)$ die Klasse aller Bäume mit $n$ Knoten. Alle diese Bäume haben das Zuverlässigkeitspolynom $z(G)(p) = p^{n-1}$ und sind somit optimal sowie $c-C_c-$optimal.

Für $m = n$ gilt folgender

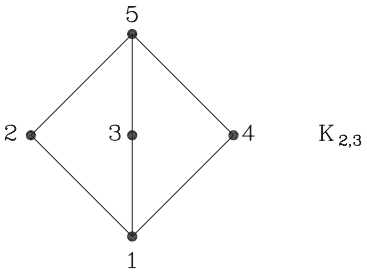
**Satz 9.7-1:** *Der (bis auf Isomorphie) einzige $c-C_c-$optimale Graph in $\mathcal{G}(n,n)$ ist der Ring $R_n$.*




**Beweis:** Zunächst überlegt man sich, dass ein Graph $G \in \mathcal{G}(n,n)$ niemals einen minimale kritische Menge aus drei oder mehr Kanten enthalten kann: wäre nämlich $S = \{k_1, k_2, k_3\}$ eine dreielementige kritische Menge, so wäre $\{k_1, k_2\}$ keine kritische Menge, d. h. bei Entfernen dieser beiden Kanten bliebe ein zusammenhängender Graph mit $n-2$ vielen Kanten übrig, was nicht möglich ist.

Man weiß nun insbesondere, dass jeder Knoten nur einen oder zwei Nachbarn hat. In einem $c-C_c-$optimalen Graphen muss folglich jede Ecke den Grad 2 haben. Es ist aber leicht zu sehen, dass $R_n$ der einzige zusammenhängende Graph mit $n$ Knoten ist, in dem jeder Knoten den Grad 2 hat.

Für den nächsten Fall $m = n + 1$ werden die Verhältnisse bereits erheblich komplizierter. Als Beispiel wollen wir den Graphen $K_{2,3}$ angeben, der $c - C_2-$optimal in der Klasse $\mathcal{G}(5, 6)$ ist (mit $c = 2$ und $C_c = 3$):




Auf eine genaue Untersuchung des Falles $m = n + 1$ wollen wir jedoch verzichten, wir verweisen auf die bereits zitierte Literatur.

Zum Abschluss gehen wir das Problem der Netzsynthese noch einmal anders an. Dabei benutzen wir den Begriff **Kantenzusammenhangszahl** für die minimale Größe $c$ einer kritischen Menge in einem Graphen $G$.

**Kantenzusammenhangszahl**

Es wird die folgende Frage gestellt: gegeben seien natürliche Zahlen $n$ und $c$ (mit $c \leq \binom{n}{2}$). Welcher Graph mit $n$ vielen Knoten und Kantenzusammenhangszahl $c$ ist am unzuverlässigsten?

Es liegt auf der Hand, dass die Zuverlässigkeit eines solchen gesuchten Graphen eine untere Schranke für die Zuverlässigkeit jedes anderen Graphen mit diesen Parametern darstellt.

Es muss allerdings auch hier der Bereich der Kantenwahrscheinlichkeiten $p$ eingeschränkt werden, um die anvisierten Ergebisse zu erhalten, und zwar nehmen wir nun $p$ als sehr klein an.

Bereits in Abschnitt 9.5 wurde erwähnt, dass dann der Ausdruck

$$F_{n-1} \cdot p^{n-1} \cdot (1 - p)^{m-n+1}$$

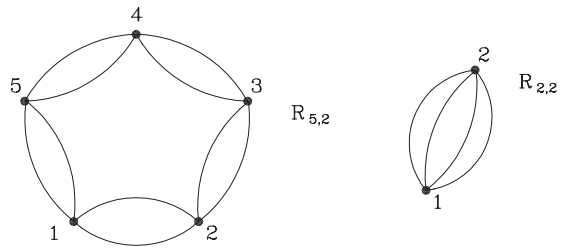
eine gute Näherung für $z(G)(p)$ ist. (Dabei bezeichnet $m$ die Anzahl der Kanten und $F_{n-1}$ die der Gerüste von $G$.)

Um die Zuverlässigkeit zu minimieren, hat es demnach Sinn, nach dem – bei gegebenem $n$ und $c$ – Graphen mit dem kleinstmöglichen Parameter $F_{n-1}$ zu fragen.

Bei der folgenden Betrachtung werden statt Graphen allgemeiner Multigraphen betrachtet, d. h. zwischen zwei Knoten sind mehrere "parallele" Kanten möglich.

Mit $R_{n,s}$ (für natürliche Zahlen $n$ und $s$ mit $n \geq 2$) bezeichnen wir den $(n, s)$-**Ring**, der aus dem Ring $R_n$ dadurch entsteht, dass jede Kante durch $s$ parallele Kanten ersetzt wird.

Im folgenden Bild sind z. B. $R_{2,2}$ und $R_{5,2}$ dargestellt:




In dem Sonderfall $n = 2$ verbindet man die beiden Knoten durch $2s$ viele Kanten (vgl. $R_{2,2}$ im Diagramm). $R_{n,s}$ hat so für alle $n \geq 2 \quad n \cdot s$ viele Kanten und Kantenzsammenhangszahl $2s$. Unter allen Multigraphen (Graphen eingeschlossen) ist $R_{n,s}$ hinsichtlich der Anzahl seiner Gerüste extremal.

**Satz 9.7-2:** *Sei $G = (E, K)$ ein Multigraph mit $n \geq 2$ vielen Knoten und Kantenzusammenhangszahl $c = 2s$. Dann hat $G$ mindestens $n \cdot s^{n-1}$ viele Gerüste; diese Abschätzung ist genau dann exakt, wenn $G$ isomorph zu $R_{n,s}$ ist.*

**Beweis:** Es wird Induktion über die Anzahl $n$ der Knoten geführt. Für $n = 2$ ist die Aussage des Satzes offensichtlich. Angenommen, der Satz sei richtig für alle Multigraphen mit weniger als $n$ vielen Knoten. Ein Multigraph mit $n$ Knoten und $c = 2s$ sei gegeben. Für jede Kante $k$ ist die Anzahl der Gerüste, die $k$ enthalten, gleich der Anzahl der Gerüste von $G \cdot k$. (Mit $G \cdot k$ ist der Multigraph bezeichnet, der entsteht, wenn die Endknoten von $k$ zu einem Knoten zusammengefaßt und $k$ sowie alle zu $k$ parallele Kanten gestrichen werden.) Da $G \cdot k \quad n - 1$ Knoten besitzt und Kantenzusammenhangszahl mindestens $2s$, folgt nach Induktionsvoraussetzung

$$b(k) \geq (n-1) \cdot s^{n-2},$$

wenn $b(k)$ die Anzahl der $k$ enthaltenden Gerüste von $G$ bezeichnet.

Mit der Summe

$$\sum_{k \in K} b(k)$$

zählt man nun jedes Gerüst von $G$ genau $(n-1)$ mal, mithin hat $G$ mindestens

$$\mid K \mid \cdot s^{n-2}$$

viele Gerüste. Da aber für $G$ gilt $c = 2s$, folgt $\mid K \mid \geq n \cdot s$ (denn jeder Knoten muss mindestens $2s$ viele Nachbarn haben), und die gewünschte Ungleichung folgt.

Damit $G$ genau $n \cdot s^{n-1}$ viele Gerüste besitzt, muss $G$ auch genau $n \cdot s$ viele Kanten besitzen, $\mid K \mid = n \cdot s$, und für jedes $k \in K$ muss $G \cdot k \quad (n-1) \cdot s$ viele Kanten haben. Folglich müssen die Kanten von $G$ aus $n$ "Parallelbündeln" von jeweils $s$ Kanten bestehen, was zusammen mit der Bedingung $c = 2s$ nur im Falle $G \cong R_{n,s}$ erfüllt ist.

**Folgerung 9.7-1:** *Sei $G = (E, K)$ ein Graph mit $\mid E \mid = n, \mid K \mid = m$ und gerader Kantenzusammenhangszahl $c = 2s$. Dann gilt für das Zuverlässigkeitspolynom $z(G)$ die Ungleichung*

$$z(G)(p) \geq n \cdot s^{n-1} \cdot p^{n-1} \cdot (1-p)^{m-n+1}.$$

*Im allgemeinen muss die rechte Seite der Ungleichung keine gute Näherung für $z(G)(p)$ sein. Dies ist jedoch der Fall, wenn die Anzahl der Gerüste von $G$ nicht wesentlich größer als $n \cdot s^{n-1}$ und $p$ nahe bei $0$ ist.*

**Selbsttestaufgabe 9.7-1:**

*Erläutern Sie, warum es sinnvoll ist, nach $c - C_c-$optimalen Graphen in der Klasse $\mathcal{G}(n, m)$ zu suchen.*

# 10 Kleine Welten

## 10.1 Soziale und andere Netze

Wem ist das noch nicht passiert: Man lernt eine bislang fremde Person kennen und stellt dann fest, dass beide einen gemeinsamen guten Bekannten haben. "Die Welt ist klein", wird dann erstaunt bemerkt. Ein wissenschaftlich denkender Mensch könnte allerdings an dieser Stelle sagen, dies sei ein Zufall, der einem besonders im Gedächtnis bleibt, während man die viel zahlreicheren Fälle, in denen man eine Person kennenlernt und es *keinen* gemeinsamen Bekannten gibt, nicht besonders beachtet. Jedoch stellt sich heraus: Es ist *kein* Zufall, sondern der Startpunkt einer Forschungsrichtung, die viele unterschiedliche Wissenschaftsbereiche berührt. Genau darum geht es in diesem Kapitel. Das *Small-World-Phänomen* ist seit den Untersuchungen von Stanley Milgram und anderen, die in den Jahren um 1960 in den USA begannen, zunächst zu einem Forschungsgegenstand in den Sozialwissenschaften geworden. Milgram schickte Briefe an zufällig ausgewählte Menschen in Nebraska und Kansas und bat jeden der Empfänger, sein Schreiben in Richtung eines Börsenmaklers weiterzuleiten, der in Boston lebte. Dessen Anschrift gab Milgram nicht an, sondern bat die Empfänger nur, die Briefe an jemanden zu schicken, den sie persönlich kannten und von dem sie annahmen, dass er "näher" an dem Börsenmakler ist. Die meisten dieser Briefe erreichten Milgrams Freund (den Börsenmakler in Boston), der Clou jedoch war: In der Regel wurden nur sechs Zwischenschritte benötigt! Man spricht von dem **Small-World-Phänomen** und meint damit, dass in dem betrachteten Netz je zwei Individuen (oder allgemeiner: Instanzen, Stationen) mit hoher Wahrscheinlichkeit durch eine kurze Kette dazwischen liegender "Bekannter" verbunden sind. Heute findet das Phänomen in vielen anderen Wissenschaften Beachtung, denn es zeigt sich in vielen Netzen, die in Natur oder Technik eine Rolle spielen, unter anderem auch im *World Wide Web*. Das Small-World-Phänomen hat seit langem seinen festen Platz in der Welt der Anekdoten. Es wird auch "zum Spaß" auf größere Menschengruppen angewendet (z. B. alle Filmschauspieler, alle Mathematiker usw.), wo es interessant erscheint (oder sogar zu verblüffenden Ergebnissen führt), zwischen Personen innerhalb dieser Gruppe die "Abstände" zu bestimmen. Es folgen zwei Beispiele.

**Small-World-Phänomen**

**Beispiel 10.1-1:**
Im Internet findet man unter der Adresse http://www.cs.virginia.edu/oracle eine Seite mit der Überschrift *The Oracle of Bacon at Virginia*. Es handelt sich um eine Website, auf der man sich zu zwei beliebigen Filmschauspielern deren "Abstand" anzeigen lassen kann - wobei zwei Schauspieler direkt miteinander verbunden sind, wenn sie gemeinsam in einem Film gespielt haben. Die Website wurde von dem Informatiker Brett Tjaden an der University of Virginia aufgebaut, der die Beobachtung machte, dass der recht wenig bekannte Schauspieler Kevin Bacon im Zentrum dieses "Filmuniversums" steht: Niemand, der je in

einem amerikanischen Film mitgespielt hat, besitzt eine Bacon-Zahl von mehr als 4.

Gibt man auf der Webseite beispielsweise die Namen Humphrey Bogart (im Jahre 1956 verstorbener Hollywood-Schauspieler) und Anke Engelke (heutige deutsche Komikerin und Schauspielerin) ein, so lautet die Antwort des Orakels: 3 - denn Humphrey Bogart hat zusammen mit Rod Steiger einen Film gemacht, dieser mit Steve Kramer, und letzterer taucht neben Anke Engelke in einem Film auf. (Die Namen der Filme werden auch genannt.)

**Beispiel 10.1-2:**
Der bekannte ungarische Mathematiker Paul Erdös, der im Jahre 1996 verstarb, hat eine überwältigende Anzahl wissenschaftlicher Veröffentlichungen verfasst (vorwiegend aus dem Bereich der Graphentheorie und der Kombinatorik), davon die meisten zusammen mit anderen Autoren. Seit vielen Jahren machen sich die Mathematiker auf der ganzen Welt den Spaß, ihre eigene *Erdös-Zahl* zu betrachten - das ist die Länge der kürzesten Kette zu Paul Erdös, wobei eine direkte Verbindung dann gegeben ist, wenn man eine gemeinsame Veröffentlichung verfasst hat.

Auch in dem hier vorliegenden Graphen beobachtet man überraschend kleine Abstände. Unter der Adresse http://www.oakland.edu/enp, hinter der sich das "*Erdös number project*" verbirgt, bekommt man Hilfen angeboten, die einen bei der Bestimmung der eigenen Erdös-Zahl unterstützen.

Übrigens besitzt der zweite Autor dieses Kurses (Poguntke) die Erdös-Zahl 2, denn er hat eine Veröffentlichung mit Peter Winkler von der Emory University in den USA verfasst, und dieser hat ein gemeinsames Papier mit Erdös vorzuweisen. Damit hat der erste Autor (Kaderali) eine Erdös-Zahl von 3, wofür dieser Kurs ein Beleg ist.

Die folgenden Abschnitte sind der Frage gewidmet: *Wie weit kann man das Small-World-Phänomen graphentheoretisch erfassen, d. h. aus den graphentheoretischen Eigenschaften des jeweils zugrunde liegenden Netzes ableiten?* Es zeigt sich, dass die Verfolgung dieser Frage zu interessanten mathematischen Problemen führt. Jedoch gibt es vor allem auch außermathematische Motive, sich mit der Frage zu beschäftigen. Buchanan schreibt in seinem populärwissenschaftlichen Buch [Buc02]: "Es könnte sich erweisen, dass einige der tiefsten Wahrheiten unserer Welt nicht auf ihre Bestandteile oder deren Eigenschaften zurückzuführen sind, sondern auf ihre Organisationsformen." Sollte es also gelingen, das Small-World-Phänomen mathematisch (in der Sprache der Graphentheorie) zu erklären, so könnte die Untersuchung der relevanten Graphenstrukturen zu interessanten Aussagen für zahlreiche reale Netze führen, betreffend z. B. die Verbreitung von Nachrichten durch persönliche Weitergabe, die Ausbreitung von Epidemien über Kontaktpersonen, Suchstrategien im World Wide Web, die Struktur von Neuronennetzen oder Ökosystemen usw.

## 10.2 Durchmesser, Cluster-Koeffizient, und die Rolle von Zufallsgraphen

In diesem Abschnitt werden meist nur ungerichtete Graphen $G = (E, K)$ - also ohne Schleifen oder Doppelkanten - betrachtet. Für unsere Untersuchungen ist offensichtlich der *Durchmesser* eines Graphen von Interesse.

**Durchmesser**

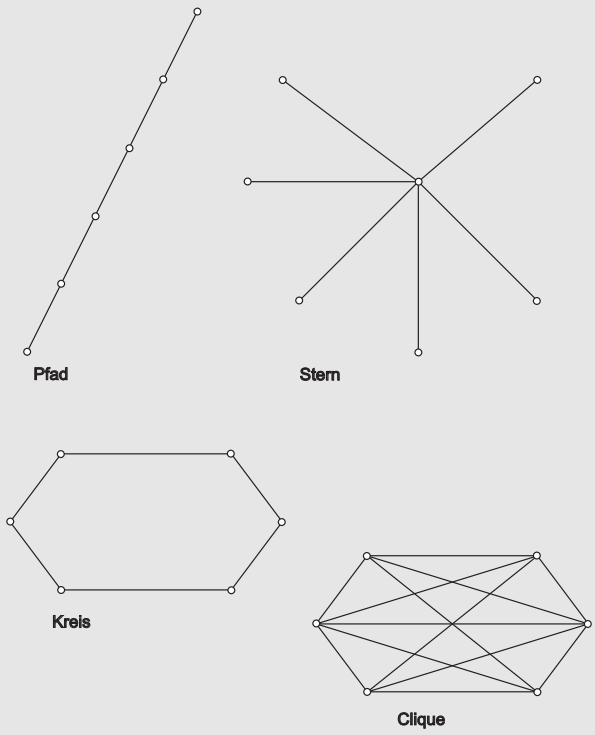
**Definition 10.2-1:** *$G = (E, K)$ sei ein Graph. Das Maximum der Abstände $\max_{e,f \in E} d(e, f)$ heißt **Durchmesser** von $G$.*

Ein verwandter Begriff ist der des mittleren Abstandes.

**mittlerer Abstand**

**Definition 10.2-2:** *$G = (E, K)$ sei ein Graph mit $n$ vielen Ecken. Der Durchschnittswert $\frac{2}{n(n-1)} \sum_{u \neq v} d(u, v)$ heißt **mittlerer Abstand** in $G$. Die zum mittleren Abstand analoge Begriffsbildung für Zufallsgraphen ist der erwartete Abstand, der noch mehrfach vorkommen wird.*

**Beispiel 10.2-1:**
Ein Pfad mit 6 Knoten hat offenbar den Durchmesser 5 und mittleren Abstand $\frac{7}{3}$. Ein Stern mit einem inneren und 6 äußeren Knoten hat den Durchmesser 2 und mittleren Abstand $\frac{12}{7}$. Ein Kreis aus 6 Knoten hat den Durchmesser 3 und mittleren Abstand $\frac{9}{5}$. Ein vollständiger Graph (Clique) mit 6 Knoten hat Durchmesser und mittleren Abstand 1.




Einige einfache Graphentypen

Der Durchmesser gibt also an, wie groß der Abstand zwischen zwei Knoten höchstens sein kann. Das Small-World-Phänomen scheint damit zu tun zu haben, dass ein Graph mit vielen Knoten einen recht kleinen Durchmesser besitzt. Nun ist andererseits klar, dass ein vollständiger Graph, in dem zwischen allen Knoten Kanten existieren, den Durchmesser 1 hat – ein kleiner Durchmesser ist mithin nicht *per se* etwas Besonderes. Bemerkenswert wird es erst dann, wenn in einem Graphen mit *vielen* Knoten und relativ *wenigen* Kanten (das wird bald präzisiert) der Durchmesser klein ist.

**Beispiel 10.2-2:**
Auf der Erde leben heute (8. September 2005) ca. 6,5 Milliarden Menschen (kurz: $6,5 \cdot 10^9$). Angenommen, jeder Mensch würde 1000 andere Menschen gut kennen. Der sich daraus ergebende Bekanntheitsgraph hat dann etwa $\frac{6{,}5 \cdot 10^9 \cdot 1000}{2} = 3,25 \cdot 10^{12}$ viele Kanten. Der vollständige Graph mit ebenso vielen Knoten hat demgegenüber ca. $2 \cdot 10^{19}$ viele Kanten, also etwa $10^7$ mal so viele. Anders gesagt: Der Bekanntheitsgraph hat nur ein Zehntausendstel Promille aller möglichen Kanten, trotzdem scheint der Durchmesser recht klein zu sein.

Schauen wir einmal auf die folgende Übersicht, in der für die charakteristischen Graphenarten aus Beispiel 9.8-1 die Kantenanzahlen sowie die Durchmesser in Abhängigkeit der Knotenanzahl $n$ angegeben sind:

| | #Kanten | Durchmesser |
|---|---|---|
| Pfad | $n-1$ | $n-1$ |
| Stern | $n-1$ | $2$ |
| Kreis | $n$ | $\lfloor \frac{n}{2} \rfloor$ |
| Clique | $\frac{n(n-1)}{2}$ | $1$ |

Wie man sieht, bekommt man einen kleinen Durchmesser bei sehr vielen Kanten (Clique) oder aber bei einer sehr speziellen Struktur wie beispielsweise einem Stern. An dem Beispiel des Erdös-Graphen sieht man, dass letzteres (also: *ein* Knoten mit sehr hohem Eckengrad) durchaus eine Erklärung für das Small-World-Phänomen sein kann. Was können wir jedoch allgemein sagen? Mit welchem Durchmesser ist "im Schnitt" zu rechnen? Anders gesagt: *Was ist der erwartete Durchmesser eines Zufallsgraphen?*

In Abschnitt 9.1 hatten wir als Folgerung im dortigen Zusammenhang die folgende Aussage formuliert.

**Folgerung 10.2-1:** *Für gegebenes $p \notin \{0,1\}$ hat fast jeder Graph in $\mathfrak{G}(n,p)$ den Durchmesser 2.*

Etwas locker formuliert, könnte man das auch so ausdrücken: *Fast jeder Graph hat Durchmesser 2.* Diese überraschend klingende Aussage könnte man bereits als eine Erklärung für das Small-World-Phänomen ansehen. Man darf diese Grenzwertaussage jedoch nicht überinterpretieren und sollte sich erinnern, was die in

Abschnitt 9.1 eingeführte Sprechweise "Fast jeder Graph..." bedeutet: dass mit immer weiter wachsender Eckenanzahl ein zufällig herausgegriffener Graph mit fast 100%iger Wahrscheinlichkeit die betreffende Eigenschaft hat. Und dies liegt dann daran, dass für große Graphen diejenigen mit der Eigenschaft zahlenmäßig immer mehr überwiegen. Dies bedeutet nun selbstverständlich nicht, dass *jeder* Graph (habe er auch noch so viele Knoten) die betrachtete Eigenschaft (etwa: Durchmesser 2) hat. Auch kann man daraus nicht schließen, dass ein konkret betrachteter Graph mit vielen Knoten (etwa der Bekanntheitsgraph aus Beispiel 10.1-2) wahrscheinlich den Durchmesser 2 hat – vielleicht ist dieser Graph ja immer noch viel zu klein!

Wir wollen uns der Fage noch einmal direkter nähern und fragen: *Wie groß ist der erwartete Abstand zwischen zwei beliebigen Ecken in einem Zufallsgraphen*$G \in \mathfrak{G}(n,p)$*?*

Dabei ist offensichtlich, dass für eine beliebige Ecke in $G \in \mathfrak{G}(n,p)$ der Erwartungswert des Eckengrades $z := p(n-1)$ beträgt, da jeder der möglichen $n-1$ vielen Nachbarn unabhängig voneinander mit Wahrscheinlichkeit $p$ ausgewählt wird.

Der folgende Satz und sein Beweis sind etwas unscharf formuliert, damit auf einige zu technischen Details verzichtet werden kann.

**Satz 10.2-1:** *Ist* $p << 1$ *(also* $p$ *sehr klein), so ist der erwartete Abstand zwischen zwei beliebigen Ecken in* $G \in \mathfrak{G}(n,p)$ *ungefähr gleich* $\frac{\log n}{\log z}$.

**Beweis:** Jeder der erwarteten $z$ vielen Nachbarn einer beliebigen Ecke $e$ hat selbst wieder $z$ viele Nachbarn, die vorwiegend (wegen $z << n$) "neue" Ecken sind, weshalb man so zu ungefähr $z^2$ vielen Ecken kommt, die von $e$ höchstens den Abstand $2$ haben. Wählt man nun $l$ so groß, dass gerade gilt $z^l = n$, so hat man alle Knoten erfasst, d. h. jeder Knoten hat einen erwarteten Abstand von höchstens $l$ zur Ecke $e$. Aus $z^l = n$ bekommt man $l = \frac{\log n}{\log z}$.

Bei der Untersuchung der Eigenschaften von Zufallsgraphen $G \in \mathfrak{G}(n,p)$ für wachsendes $n$ ist es üblich, nicht $p$ konstant zu halten, sondern den erwarteten Eckengrad $z = p(n-1)$, m. a. W. wird $p = p(n)$ als fallende Funktion angenommen. Für die erwarteten Abstände führt dies mit Satz 10.2-1 zu der Folgerung:

**Folgerung 10.2-2:** *Für wachsendes* $n$ *ist der erwartete Abstand zwischen zwei beliebigen Knoten in* $G \in \mathfrak{G}(n,p)$ *proportional zu* $\log n$.

In der Literatur findet man die Aussage dieser Folgerung oft vereinfacht so formuliert:

***Der Durchmesser eines Zufallsgraphen*** $G \in \mathfrak{G}(n,p)$ ***ist ungefähr gleich*** $\log n$**.**

Wir wollen das Beispiel 10.2-2 unter diesem Gesichtspunkt noch einmal analysieren.

**Beispiel 10.2-3:**
Hier ist $n = 6,5 \cdot 10^9$, und wir gehen von einem Grad $z = 1000$ für jede Ecke aus. Handelte es sich um einen Zufallsgraphen, so würde sich daraus wegen $z = p(n-1)$ ergeben $p \approx 10^{-7}$; mit $\log n \approx 9,8$ und $\log z = 3$ ergäbe sich aus Satz 9.8-1 für beliebige Knoten ein erwarteter Abstand von höchstens 3,3.

An diesem Beispiel sieht man: In Zufallsgraphen hat man im Schnitt sogar noch kleinere Abstände, als sie in unseren Beispielen für das Small-World-Phänomen vorliegen. Da es das Ziel ist, das Small-World-Phänomen möglichst passend graphentheoretisch zu modellieren, kann die Schlussfolgerung nur lauten, dass Zufallsgraphen nicht die geeigneten Modelle liefern.

In der Tat hat man beobachtet, dass "Soziale Netze" eine charakteristische Eigenschaft haben, die in Zufallsgraphen nicht gegeben ist – es ist die Tendenz zur Bildung von *Clustern*. Ausgangspunkt ist die folgende Beobachtung: Für zwei Freunde einer Person A ist die Wahrscheinlichkeit, dass diese *ebenfalls* Freunde sind, größer als im Durchschnitt. Bei einem Zufallsgraphen ist dies nicht der Fall: für zwei beliebige Ecken ist die Wahrscheinlichkeit, dass sie durch eine Kante verbunden sind, immer gleich $p$ – egal, ob sie bereits einen gemeinsamen Nachbarn haben oder nicht. Die präzise Terminologie ist die folgende. (Die Menge aller zweielementigen Teilmengen einer Menge $N$ ist hier kurz mit $\binom{N}{2}$ bezeichnet.)

**Definition 10.2-3:** *Für eine Ecke $e \in E$ in einem Graphen $G = (E, K)$ sei $N_e$ die Menge der zu $e$ benachbarten Ecken.*

*Wir setzen*

$$C_e := \frac{\left|\binom{N_e}{2} \cap K\right|}{\binom{N_e}{2}}.$$

*$C_e$ ist also der Bruchteil aller Paare von Nachbarn von $e$, die selbst durch eine Kante verbunden sind.*

*Der **Cluster-Koeffizient** von $G$ ist der durchschnittliche Wert der $C_e$, d. h.* **Cluster-Koeffizient**

$$C := \frac{\sum C_e}{E}.$$

Der Cluster-Koeffizient misst also, in welchem Anteil im Schnitt die Nachbarn einer Ecke selbst wieder benachbart sind. In der folgenden Tabelle ist der Cluster-Koeffizient für eine Reihe von Graphenklassen eingetragen. Dabei steht $n$ für die Anzahl der Ecken, $m$ für die Anzahl der Kanten, $l$ für den mittleren Abstand zweier Ecken und $C$ für den Cluster-Koeffizienten; bei den Zufallsgraphen sind für $m$, $l$ und $C$ die Erwartungswerte eingetragen. Mit "Filmstars" ist der Graph aus Bei-

spiel 10.1-1 gemeint, für den Amaral et al. (siehe [Ama00]) die entsprechenden Werte (Stand:2000) angegeben haben.

| | Ecken $n$ | Kanten $m$ | mittl. Abst. $l$ | Clu.-Koeff. $C$ |
|---|---|---|---|---|
| Kreis | $n \geq 4$ | $n$ | $\approx \frac{n}{4}$ | $0$ |
| Clique | $n \geq 3$ | $\binom{n}{2}$ | $1$ | $1$ |
| $G \in \mathfrak{G}(n,p)$ | $n$ | $p\binom{n}{2}$ | $\approx \frac{\log n}{\log p(n-1)}$ | $p$ |
| $G \in \mathfrak{G}(449913,\ 0,00025)$ | $449913$ | $25516482$ | $2,75$ | $0,00025$ |
| Filmstars | $449913$ | $25516482$ | $3,48$ | $0,78$ |

Besonders interessant ist nun der direkte Vergleich zwischen dem Graphen "Filmstars" und einem Zufallsgraphen mit ebenso vielen Ecken und Kanten: Bei dem Zufallsgraphen kann man zwar einen noch kleineren mittleren Abstand erwarten $(2,75)$, als er bei den Filmstars vorliegt $(3,48)$, aber der Cluster-Koeffizient ist bei diesem "Sozialen Netz" dramatisch höher. Man kann dies so sehen: In einem Sozialen Netz werden gegenüber einem Zufallsgraphen mehr Kanten für die Clusterbildung verwendet, wodurch weniger Kanten für die restlichen Wege zur Verfügung stehen und so die mittleren Abstände größer werden. In den nächsten Abschnitten werden wir uns mehrere Graphenklassen ansehen, die sich gegenüber den Zufallsgraphen als bessere Modelle für Soziale Netze herausgestellt haben.

**Selbsttestaufgabe 10.2-1:**

*Berechnen Sie den erwarteten mittleren Abstand sowie den erwarteten Cluster-Koeffizienten in einem Zufallsgraphen mit den Parametern aus Beispiel 10.2-2, d. h.* $n = 6,5 \cdot 10^9$ *und* $z = 1000$.

## 10.3 Das Modell von Watts und Strogatz

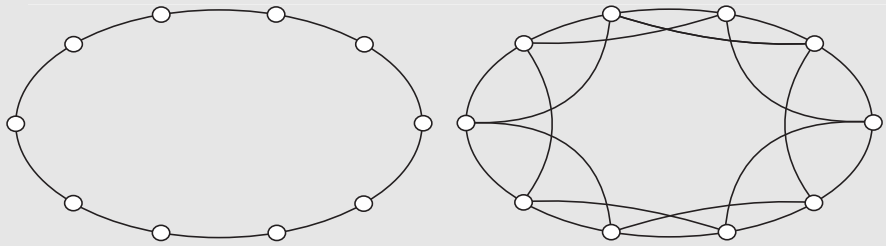
Aufgrund der Ergebnisse des vorigen Abschnitts ist klar, dass man auf der Suche nach Graphenklassen mit den folgenden Eigenschaften ist:

- große Eckenzahl $n$, relativ wenige Kanten (z. B. $O(n)$)
- mittlerer Abstand $O(\log n)$ oder wenig größer
- möglichst hoher Cluster-Koeffizient, ab $C \geq 0,1$

**Small-World-Graphen**

Einige Autoren nennen Graphen mit diesen Eigenschaften **Small-World-Graphen**. Von Watts und Strogatz (siehe [Wat98]) stammt die folgende Konstruktion, die nach den beiden Autoren benannt wurde. Zunächst wird ein Graph konstruiert, in dem jede Ecke $z$ viele Nachbarn hat. (Der Einfachheit halber setzen wir $z$ als gerade natürliche Zahl voraus.) Man geht dazu von einem Kreis $K_n$ aus, in dem jedoch jede Ecke nicht nur mit ihren beiden Nachbarn, sondern insgesamt mit den $z$ vielen ihr am nächsten gelegenen Ecken *direkt* (also durch eine Kante) verbunden wird. Den resultierenen Graphen nennen wir $K_{n,z}$.

**Beispiel 10.3-1:**
Das erste Diagramm zeigt den Graphen $K_{10}$, das zweite $K_{10,4}$.



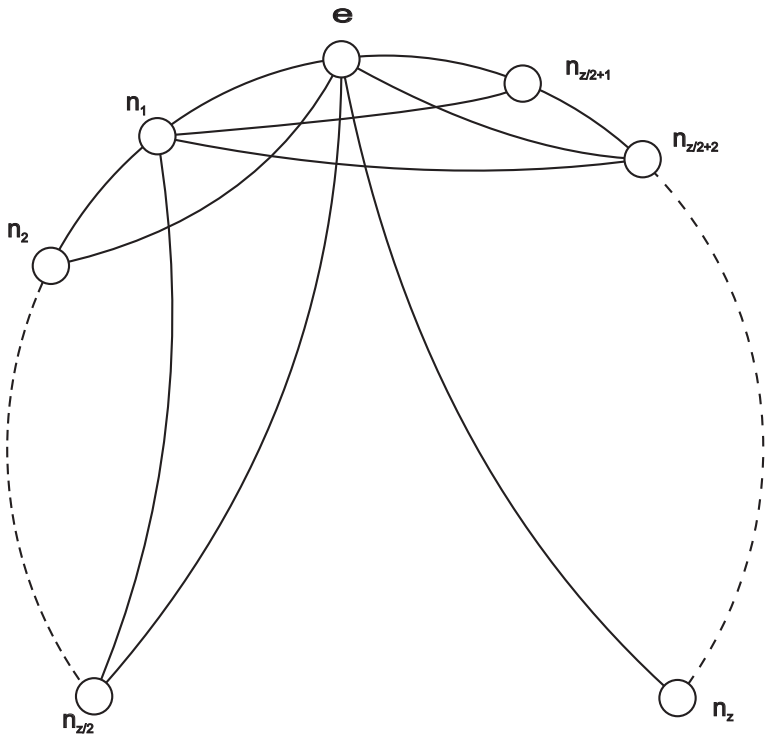
Die Graphen $K_{10}$ und $K_{10,4}$.

$K_{n,z}$ ist eine Art Zwischenstufe bei der Konstruktion der Small-World-Graphen. Im Extremfall $z = 2$ hat man den Kreis $K_n$ (mit $C = 0$ falls $n > 3$), für $z = n - 1$ einen vollständigen Graphen (mit $C = 1$). Generell ist der Cluster-Koeffizient recht hoch:

**Satz 10.3-1:** *Ist* $2 < z < \frac{2}{3}n$*, so hat* $K_{n,z}$ *den Cluster-Koeffizienten*

$$C = \frac{3(z-2)}{4(z-1)}.$$

**Beweis:** Aus Symmetriegründen reicht es, den Cluster-Koeffizienten $C_e$ einer beliebigen Ecke $e$ zu bestimmen. Dazu schauen wir auf die folgende Grafik:




Die Nachbarn einer Ecke e in $K_{n,z}$.

Auf jeder der beiden Seiten von $e$ (im Sinne des ursprünglichen Kreises $K_n$) sind die nächsten $\frac{z}{2}$ vielen Ecken eingezeichnet, die zur Bildung von $K_{n,z}$ mit $e$ direkt verbunden werden. Wir haben nun die Anzahl der Kanten zwischen den $n_i$ zu bestimmen; Division durch $\binom{z}{2}$ ergibt dann den gesuchten Koeffizienten $C_e$. Die Ecke $n_1$ ist mit $n_2, n_3, ..., n_{\frac{z}{2}}$ sowie mit $n_{\frac{z}{2}+1}, n_{\frac{z}{2}+2}, ..., n_{z-1}$ verbunden, insgesamt also mit $z-2$ der übrigen Ecken. Die entsprechende Betrachtung für $n_2$ führt zu $z-3$ Kanten usw. bis zur Ecke $n_{\frac{z}{2}}$, die nur mit den "auf ihrer Seite liegenden" $\frac{z}{2}-1$ vielen übrigen $n_i$ verbunden ist. Da die gleiche Betrachtung für die auf der anderen Seite liegenden Ecken $n_{\frac{z}{2}+1}$ etc. gilt, dabei aber insgesamt jede Kante doppelt gezählt wird, kommen wir zu dem Ergebnis

$$C_e = \frac{(z-2)+(z-3)+...+(\frac{z}{2}-1)}{\binom{z}{2}}$$

Dies kann durch elementare Umformungen überführt werden in.

$$C_e = \frac{3(z-2)}{4(z-1)}.$$

Zu diesem Satz und Beweis sind zwei Anmerkungen nötig.

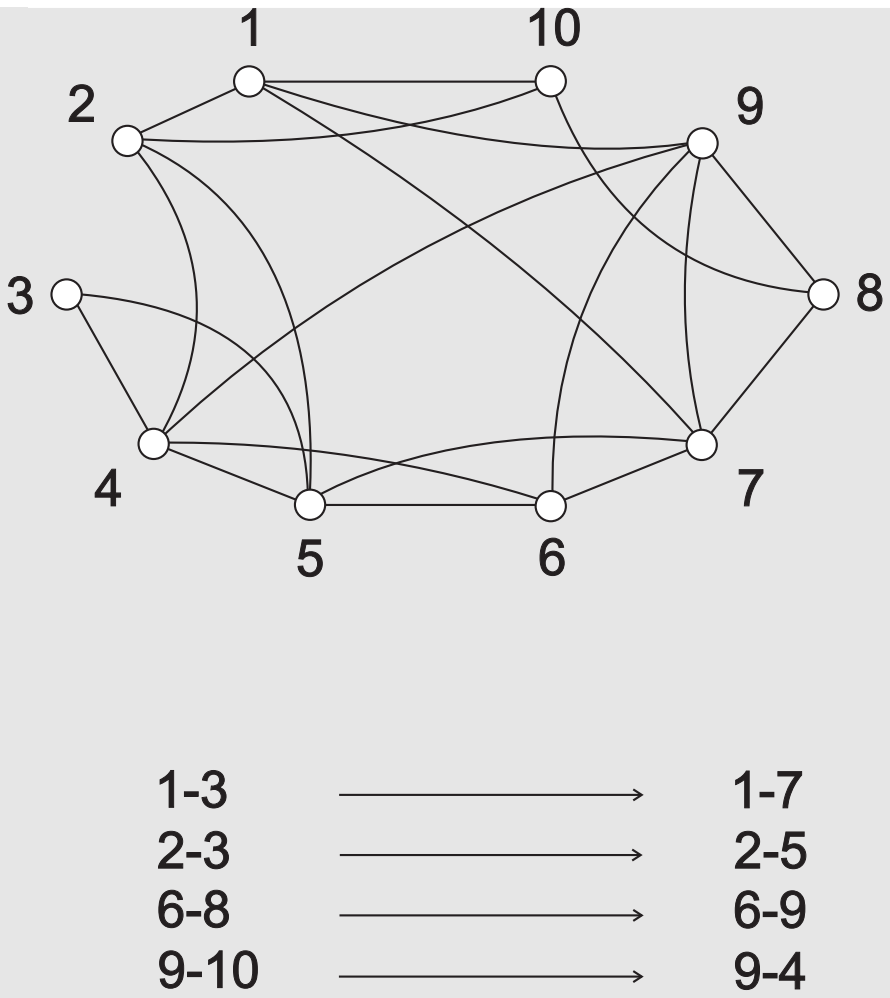
1. Die Voraussetzung $z < \frac{2}{3}n$ sorgt dafür, dass nicht *noch mehr* der $n_i$ miteinander verbunden sind (z. B. $n_1$ mit $n_z$), was der Fall wäre, wenn sich der Kreis (anschaulich gesprochen) zu bald schließen würde. Anders gesagt: Ist diese Voraussetzung nicht erfüllt, ist der Cluster-Koeffizient *noch größer*.

2. Der hier gegebene Cluster-Koeffizient ist offenbar unabhängig von $n$ und tendiert für großes $z$ zum Wert $\frac{3}{4}$.

Die Graphen $K_{n,z}$ besitzen also einen recht hohen Cluster-Koeffizienten, jedoch sind die mittleren Abstände zwischen den Ecken immer noch recht groß – wenn auch gegenüber $K_n$ immerhin leicht verkleinert: Ist $z$ gegenüber $n$ klein, so liegt der mittlere Abstand $l$ immer noch in der Nähe von $\frac{n}{4}$. Die Konstruktion von Watts und Strogatz, auf die wir hinaus wollen, sieht nun folgendermaßen aus:

*Es wird von einem Graphen $K_{n,z}$und einer festen Wahrscheinlichkeit $p \in (0,1)$ ausgegangen. Nun wird jede der in $K_{n,z}$ vorhandenen Kanten mit Wahrscheinlichkeit $p$ "umgepolt". Umpolen einer Kante bedeutet, eine der beiden Endecken der Kante (welche, wird zufällig bestimmt) durch eine zufällig ausgewählte unter den übrigen $n-2$ Ecken zu ersetzen und die Kante mit dieser Ecke zu verbinden.*

**Beispiel 10.3-2:**
Ausgegangen wird von dem Graphen $K_{10,4}$ (wie in Beispiel 10.3-1). Mit $p = 0,2$ kann sich z. B. der im folgendem Bild dargestellte Graph ergeben. Vier Kanten sind umgepolt. Beispielsweise ist die Kante $\{1,3\}$ in die Kante $\{1,7\}$ übergegangen.




Ein möglicher Zufallsgraph nach dem Modell von Watts-Strogatz

Wir nennen einen wie oben konstruierten Graphen einen **Zufallskreis**. Dazu merken wir an:

**Zufallskreis**

**Bemerkung 10.3-1:**

- *Für eine wahrscheinlichkeitstheoretisch exakte Definition müsste man analog der Situation bei Zufallsgraphen von einem Wahrscheinlichkeitsraum $\mathfrak{K}(n, z, p)$ sprechen, der aus allen möglichen Graphen als Resultate des beschriebenen Erzeugungsprozesses besteht.*

- *Beim Umpolen werden erwartungsgemäß $\frac{p}{2}Nz$ der $\frac{1}{2}Nz$ vielen in $K_{n,z}$ vorhandenen Kanten umgepolt. In Beispiel 10.3-2 sind 4 von 20 möglichen Kanten umgepolt, was dem Erwartungswert entspricht.*

- *Das Modell "Zufallskreis" lässt sich im Kontext sozialer Netze durchaus motivieren: Beispielsweise sind die meisten Menschen vorwiegend mit solchen aus ihrer Nachbarschaft, Familie usw. gut bekannt – jedoch gibt es immer auch "Ausreißer" in dem Sinne, dass jemand (meist als Resultat einer Kette von Zufällen) einen guten Freund in einer anderen Stadt, einem anderen Land oder aus einem sonstigen gänzlich anderen Lebensumfeld besitzt.*

Untersuchungen sowohl von Watts und Strogatz als auch anderer Autoren haben ergeben, dass bei der Konstruktion von Zufallskreisen in der Tat Small-World-Graphen herauskommen: Der Cluster-Koeffizient nimmt zwar gegenüber $K_{n,z}$ mit dem Faktor $(1-p)^3$ ab (vgl. [Bar00]), der mittlere Eckenabstand wird jedoch ebenfalls deutlich geringer und kommt in die Nähe der Werte für Zufallsgraphen. Leider ist es bisher nicht gelungen, den Wert $l$ (bzw. den Erwartungswert) rechnerisch genau zu bestimmen. Watts und Strogatz haben jedoch durch Simulationen festgestellt, dass schon für kleine Werte von $p$ (also tendenziell wenige "Umpolungen") der mittlere Abstand im Schnitt in die Nähe von $\log n$ kommt (wie bei Zufallsgraphen). In der folgenden Tabelle sind die durch Simulationen geschätzten Werte von $G \in \mathfrak{K}(1000,\ 10,\ 0,25)$ denen von Zufallsgraphen und $K_{1000,10}$ gegenüber gestellt.

| | Ecken $n$ | Kanten $m$ | mittl. Abst. $l$ | Clu.-Koeff. $C$ |
|---|---|---|---|---|
| $K_{1000,10}$ | 1000 | 5000 | 50 | $0,67$ |
| $G \in \mathfrak{K}(1000,\ 10,\ 0,25)$ | 1000 | 5000 | $3,6$ | $0,28$ |
| $G \in \mathfrak{G}(1000,\ 0,01)$ | 1000 | 5000 | 2 | $0,01$ |

Das hier diskutierte Modell von Watts und Strogatz kann problemlos auf den Fall verallgemeinert werden, dass nicht mit einem *Kreis* begonnen wird, in dem zunächst auf regelmäßige Weise Kanten hinzugefügt und anschließend einige von diesen zufallsgesteuert "umgepolt" werden, sondern dass ein ebenes regelmäßiges Gitter oder ein höherdimensionales entsprechendes Gebilde den Ausgangspunkt bildet. Eine weitere Variante des Watts-Strogatz-Modells haben Newman und Watts in [New99] betrachtet: Ausgehend von $K_{n,z}$ (oder einem höherdimensionalen Gebilde) werden keine Kanten umgepolt, sondern lediglich zufallsgesteuert einige weitere Kanten hinzugefügt. In dieser Variante bleibt der Graph stets zusammenhängend. Zur Bestimmung des Erwartungswertes des mittleren Abstandes $l$ existiert umfangreiche Literatur.

**Selbsttestaufgabe 10.3-1:**

*Erläutern Sie, wieso Zufallsgraphen nach dem Modell von Watts und Strogatz geeignetere Modelle für das Small-World-Phänomen darstellen als Zufallsgraphen der Klasse* $\mathfrak{G}(n,p)$.

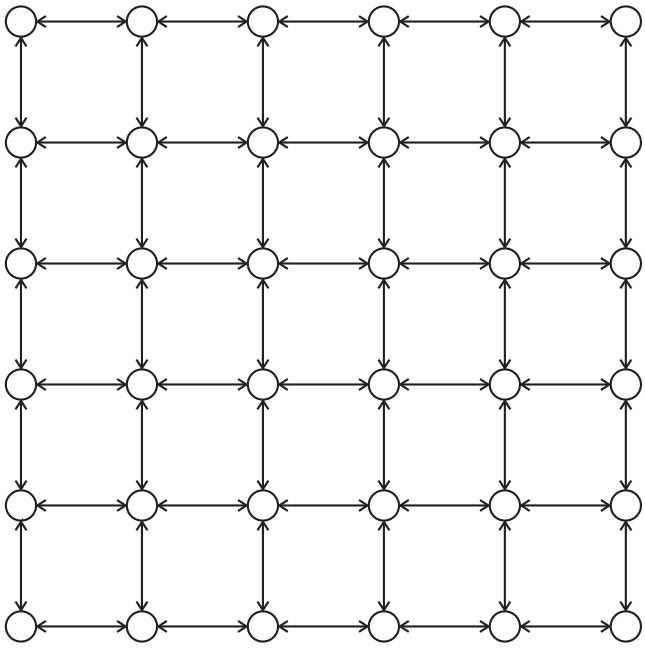
## 10.4 Modelle mit effizientem Routing

Die im vorliegenden Abschnitt behandelten Ansätze basieren auf der Beobachtung, dass in dem beschriebenen Brief-Experiment von Milgram (siehe oben im Abschnitt über "Soziale und andere Netzwerke") die kurzen Wege zu dem Bostoner Börsenmakler nicht nur existierten, sondern von den beteiligten Personen auch tatsächlich gefunden und für den Brieftransport genutzt wurden. Wir stellen also die folgende

Frage: *Angenommen, in einem Sozialen Netz würden kurze Wege stets existieren. Wie gelingt es den Individuen, diese tatsächlich zu finden?* Mathematisch formuliert: Es geht nicht nur um die *Existenz* kurzer Wege, sondern auch um *Algorithmen* zu ihrer *Konstruktion*. (Hier sind wir bei ähnlichen Problemstellungen angekommen wie beim Routing in Kommunikationsnetzen, das in den vorherigen Kapiteln behandelt wurde.)

Wir beginnen mit den Ergebnissen von Kleinberg (siehe [Kle00]). Kleinberg definiert eine Klasse graphentheoretischer Modelle, die von einigen Parametern gesteuert sind und (wie bei Watts-Strogatz) "Elemente des Zufalls" beinhalten, das Watts-Strogatz-Modell fällt darunter. Es wird dann gezeigt, dass nur für eine ganz spezielle Auswahl der Parameter ein effizienter Algorithmus zum Auffinden der kurzen Wege existiert – für das Watts-Strogatz-Modell ist dies nicht der Fall.

Wir beschreiben das Modell in mehreren Schritten. Ausgegangen wird von einem zweidimensionalen Gitter von Seitenlänge $n$, wobei die Gitterpunkte als Ecken eines Graphen aufgefasst werden und benachbarte Ecken durch gerichtete Kanten in beiden Richtungen verbunden sind. In der nächsten Abbildung ist ein solches Gitter mit Seitenlänge 6 dargestellt. (Der Einfachheit halber werden Kanten mit Doppelpfeilen gezeichnet, um zwei entgegengesetzt gerichtete Kanten zwischen benachbarten Ecken darzustellen.)



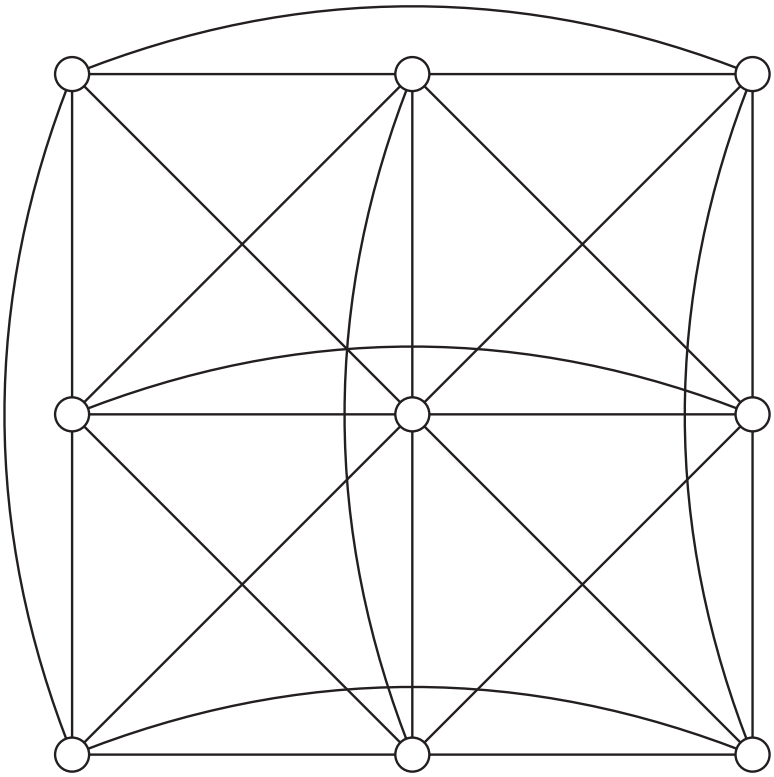
6x6 - Gitter

Formal kann man die Eckenmenge des Gitters definieren als

$$\{(i,j) \mid i,j \in \{1,2,...,n\}\}.$$

Der *Gitterabstand* zwischen zwei Ecken $(i,j)$ und $(k,l)$ ist offenbar $|k-i|+|l-j|$. Nun wird im nächsten Schritt eine natürliche Zahl $d \geq 1$ gewählt und von jeder Ecke $e$ des Gitters eine gerichtete Kante zu allen anderen Ecken gezogen, die von

$e$ höchstens den Gitterabstand $d$ haben. Das nächste Bild zeigt den Graphen mit Seitenlänge $n = 3$ und $d = 2$. (Der Übersichtlichkeit halber sind nur ungerichtete Kanten eingezeichnet, die jeweils für zwei gerichtete Kanten in beiden Richtungen stehen.)



Ein Gitter mit Kanten bis zu Gitterabstand 2

Für den letzten Konstruktionsschritt werden nun weitere Konstanten $q \geq 0$ und $r \geq 0$ gewählt und zufällig gerichtete Kanten gemäß folgender Regel hinzugefügt: *Jede Ecke $u$ erhält gerichtete Kanten zu $q$ vielen weiteren Ecken (diese nennen wir "Fernkontakte"), indem unabhängige zufällige Entscheidungen so getroffen werden, dass jede der Kanten von $u$ nach $v$ mit einer Wahrscheinlichkeit ausgewählt wird, die proportional ist zu $d(u, v)^{-r}$.* (Um eine Wahrscheinlichkeitverteilung zu erhalten, muss man den Wert $d(u, v)^{-r}$ noch durch die Normalisierungskonstante $\sum_x d(u, x)^{-r}$ teilen.)

In [Kle00] sind die nächsten drei Sätze gezeigt, von denen wir den zweiten beweisen wollen. Dazu wird die folgende Terminologie verwendet.

**dezentraler Algorithmus**

Ein **dezentraler Algorithmus** ist ein Verfahren, nach dem eine Nachricht (die mit einer Zieladresse versehen ist) schrittweise vom jeweiligen Knoten, in dem sich die Nachricht gerade befindet, basierend auf rein lokaler Information zu einem von dessen lokalen oder "fernen" Nachbarn weitergeleitet wird. Das bedeutet, dass der aktuelle "Nachrichten-Inhaber" $u$ bei der Ausführung eines solchen Schrittes Kenntnis hat

- von der Gitterstruktur aller Knoten

- von der Lage des Zieles $t$ innerhalb des Gitters
- von der Lage und den Fernkontakten aller Knoten, durch die die Nachricht bereits gelaufen ist.

$u$ hat also **keine** Kenntnis der Fernkontakte derjenigen Knoten, durch die die Nachricht **nicht** gelaufen ist.

**erwartete Zustellzeit**

Die **erwartete Zustellzeit** eines solchen dezentralen Algorithmus ist die erwartete Anzahl von Schritten, die der Algorithmus benötigt, um eine Nachricht zwischen zwei zufällig ausgewählten Knoten über ein Netzwerk zuzustellen, welches gemäß der obigen Konstruktion (mit Wahrscheinlichkeiten proportional zu $d(u,v)^{-r}$) erzeugt wurde.

**Satz 10.4-1:** *Es gibt eine Konstante $\alpha_0$, die von $d$ und $q$, jedoch nicht von $n$ abhängt, so dass im Falle $r = 0$ die erwartete Zustellzeit jedes dezentralen Algorithmus mindestens $\alpha_0 n^{\frac{2}{3}}$ beträgt.*

Man beachte: $r = 0$ bedeutet, dass bei der Festlegung der "Fernkontakte" jeder in Frage kommende Knoten mit der gleichen Wahrscheinlichkeit ausgewählt wird. Da die Abstände zwischen den Knoten mehrheitlich in der Größenordnung von $\log n$ liegen (wie bei Zufallsgraphen), ist die Aussage des Satzes so zu interpretieren, dass unter den gegebenen Bedingungen **kein** effizientes Routing existiert, denn $n^{\frac{2}{3}}$ ist exponentiell gemessen an $\log n$.

Wie der folgende Satz zeigt, ist die Situation für $r = 2$ anders.

**Satz 10.4-2:** *Es gibt einen dezentralen Algorithmus $\mathfrak{A}$ und eine Konstante $\alpha_2$, die unabhängig von $n$ ist, so dass im Falle $r = 2$ und $d = q = 1$ die erwartete Zustellzeit von $\mathfrak{A}$ höchstens $\alpha_2(\log n)^2$ beträgt.*

**Beweis:** Da $d = q = 1$ gilt, ist jede Ecke $u$ mit ihren vier unmittelbaren Nachbarn verbunden (Randecken nur mit zwei oder drei), und zusätzlich mit einer weiter entfernten Ecke $v$ (durch eine "Fernkante"). Die Wahrscheinlichkeit, dass ein gewisses $v$ gerade diese weiter entfernte Ecke für $u$ ist, beträgt $\frac{d(u,v)^{-2}}{\sum_{x \neq u} d(u,x)^{-2}}$. Wegen der Ungleichungskette

$$\sum_{x \neq u} d(u,x)^{-2} \leq \sum_{j=1}^{2n-2} (4j)(j^{-2}) = 4 \sum_{j=1}^{2n-2} j^{-1} \leq 4 + 4\ln(2n-2) \leq 4\ln(6n)$$

kann man schließen, dass $v$ mindestens mit Wahrscheinlichkeit $[4\ln(6n) d(u,v)^2]^{-1}$ ausgewählt wird.

Der dezentrale Algorithmus $\mathfrak{A}$ wird nun folgendermaßen definiert:

In jedem Schritt wählt die Ecke $u$, die eine für Ecke $t$ bestimmte Nachricht weiterleiten möchte, dazu eine zu ihr selbst benachbarte Ecke aus, die (im Sinne des Gitterabstandes) möglichst nahe an $t$ liegt.

Nun ist zu zeigen, dass der Algorithmus $\mathfrak{A}$ das Behauptete leistet.

Für $j > 0$ sagen wir, Algorithmus $\mathfrak{A}$ sei in Phase $j$, falls der Gitterabstand der aktuellen Ecke zu $t$ mehr als $2^j$, jedoch höchstens $2^{j+1}$ beträgt. $\mathfrak{A}$ ist in Phase $0$, falls die Gitterdistanz zu $t$ höchstens 2 ist.

Der Anfangswert von $j$ ist folglich höchstens $\log n$. Da nun der verbleibende Gitterabstand der Nachricht zur Zielecke $t$ mit jedem Schritt kleiner wird, bekommt jede der auf dem Weg liegenden Ecken die Nachricht nur einmal. Folglich kann man für die Argumentation annehmen, dass die "Fernkante" der aktuellen Ecke erst in diesem Moment (also wenn die Ecke die Nachricht weiterleitet) erzeugt wird.

Angenommen, wir sind in Phase $j$, $\log(\log n) \leq j < \log n$, und die Nachricht befindet sich gerade in der Ecke $u$. Wir betrachten die Wahrscheinlichkeit, dass in diesem Schritt die Phase $j$ gerade beendet wird. Dies bedeutet, dass die Nachricht in die Menge $B_j$ aller Ecken eintritt, die höchstens Gitterabstand $2^j$ von $t$ haben.

Es gibt mindestens $1 + \sum_{i=1}^{2^j} i = \frac{1}{2}2^{2j} + \frac{1}{2}2^j + 1 > 2^{2j-1}$ viele Ecken in $B_j$, jede innerhalb eines Gitterabstandes $2^{j+1} + 2^j < 2^{j+2}$ von $u$, und folglich besitzt jede eine Wahrscheinlichkeit von mindestens $[4\ln(6n)2^{2j+4}]^{-1}$, das Ende der "Fernkante" von $u$ zu sein. Falls eine dieser Ecken tatsächlich der Fernpartner von $u$ ist, so wird dies der am nächsten an $t$ gelegene Nachbar von $u$ sein; folglich tritt die Nachricht mit einer Wahrscheinlichkeit von mindestens $\frac{2^{2j-1}}{4\ln(6n)2^{2j+4}} = \frac{1}{128\ln(6n)}$ in die Menge $B_j$ ein.

Es bezeichne $X_j$ die Anzahl der in Phase $j$ verbrachten Schritte, mit $\log(\log n) \leq j < \log n$. Für den Erwartungswert erhält man

$E(X_j) = \sum_{i=1}^{\infty} P(X_j \geq i) \leq \sum_{i=1}^{\infty}(1 - \frac{1}{128\ln(6n)})^{i-1} = 128\ln(6n)$.

Durch eine ähnliche Abschätzung kann man auch $E(X_j) \leq 128\ln(6n)$ für $j = \log n$ zeigen.

Schließlich gilt die Ungleichung $E(X_j) \leq 128\ln(6n)$ auch für $0 \leq j \leq \log(\log n)$, weil der Algorithmus höchstens $\log n$ viele Schritte in Phase $j$ verbringen kann, selbst wenn alle Ecken die Nachricht an lokale Nachbarn weiterleiten.

Ist nun $X$ die Gesamtanzahl der Schritte des Algorithmus, so gilt $X = \sum_{j=0}^{\log n} X_j$, und aufgrund der Linearität des Erwartungswerts erhält man $E(X) \leq (1 + \log n)(128\ln(6n)) \leq \alpha_2(\log n)^2$ für eine geeignete Wahl von $\alpha_2$.

Damit ist der Satz bewiesen.

Wir merken an dieser Stelle an: Der innerhalb des obigen Beweises verwendete Algorithmus, bei dem jede Ecke die Nachricht an ihren am nächsten am Zielknoten gelegenen Nachbarn weiterleitet, wird auch "Greedy - Algorithmus" genannt. Allgemein nennt man in der Kombinatorischen Optimierung einer Algorithmus

"greedy" (englisch für: geitzig), wenn in jedem Schnitt die hinsichtlich der Zielsetzung am günstigen erscheinende Alternative ausgewählt wird. Dies ist nicht zwangsläufig die optimale Vorgehensweise.

Der nächste Satz den wir ohne Beweis zitieren, deckt die noch fehlenden Fälle in Bezug auf die Größe von $r$ ab. Auch hier existiert kein effizientes Routing.

**Satz 10.4-3:** *Sei $0 \leq r < 2$. Es gibt eine Konstante $\alpha_r$, die von $d$, $q$ und $r$, jedoch nicht von $n$ abhängt, so dass die erwartete Zustellzeit jedes dezentralen Algorithmus mindestens $\alpha_r n^{\frac{2-r}{3}}$ beträgt.*

*Sei $r > 2$. Es gibt eine Konstante $\alpha_r$, die von $d$, $q$ und $r$, jedoch nicht von $n$ abhängt, so dass die erwartete Zustellzeit jedes dezentralen Algorithmus mindestens $\alpha_r n^{\frac{r-2}{r-1}}$ beträgt.*

Wie erwähnt, ist uns das Thema "Routing in einem großen Netz, basierend auf lokalen Informationen" schon bei den Kommunikationsnetzen begegnet. Beim Modell von Kleinberg für Small-World-Graphen hat sich herausgestellt, dass nur ein sehr präzises Zusammenspiel von lokaler Struktur und Fernkontakten es ermöglicht, die Wege zum Ziel effizient (mit dem Greedy-Algorithmus) zu finden: Der Grund dürfte darin liegen, dass für das zweidimensionale Gitter nur im Falle $r = 2$ die Fernkontakte der Knoten gleichmäßig auf die Abstandsniveaus verteilt sind.

Als nächstes wenden wir uns den Ergebnissen aus [Barr01] zu, in denen effizientes Routing in sogenannten **Fernkontakt-Graphen** behandelt wird. Ausgehend von Kreisen mit zufälligen zusätzlichen Kanten werden anschließend Ergebnisse für eine große Klasse von Graphem allgemeiner Struktur erzielt. $G = (V, E)$ sei ein Graph mit $n$ vielen Knoten. Ferner sei

**Fernkontakt-Graphen**

$$p : V \times V \longrightarrow \mathbb{R}$$

eine Abbildung mit der Eigenschaft, dass für alle $u \in V$ gilt:

$$\sum_{v \in V} p(u, v) = 1$$

Man kann dies auch so formulieren: Zu jedem $u \in V$ gehört eine Wahrscheinlichkeitsverteilung

$$p(u, \quad) : V \longrightarrow \mathbb{R}.$$

**Definition 10.4-1:** *Ein Graph $G = (V, E)$ und $p$ (wie oben) seien gegeben. Der Fernkontakt-Graph$(G, p)$ ist ein gerichteter Graph auf der Knotenmenge $V$, in dem jeder Knoten eine gerichtete Kante zu allen seinen Nachbarn (im Sinne der Kanten in $E$) besitzt plus einer weiteren gerichteten Kante zu einem Nachbarn, der gemäß $p$ zufällig ausgewählt wird.*

Man beachte: In Wahrheit ist der Fernkontakt-Graph – wie üblich – kein Graph, sondern ein Wahrscheinlichkeitsraum bestehend aus vielen möglichen Graphen (den

Ereignissen), von denen jeder mit einer gewissen Wahrscheinlichkeit auftritt. Ist der gemäß $p$ zufällig ausgewählte weitere Nachbar bereits einer der ursprünglichen Nachbarn, so kommt selbstverständlich keine weitere Kante hinzu.

Es werden nun zunächst nur solche Fernkontakt-Graphen untersucht, bei denen der zugrunde liegende Graph ein Kreis $C_n$ ist und $p$ eine harmonische Verteilung:

**Definition 10.4-2:** *Für einen Graphen $G = (V, E)$ und $r \geq 0$ ist die $r$-harmonische Abbildung $p_r$ definiert als (für $u \neq v$)*

$$p_r(u, v) = \frac{d(u, v)^{-r}}{\sum_{w \neq u} d(u, w)^{-r}},$$

*wobei $d$ wie üblich die Abstandsfunktion im Graphen bezeichnet.*

In der folgenden Tabelle sind für Fernkontakt-Graphen, die auf Kreisen $C_n$ und $r$-harmonischen Verteilungen basieren, die Ergebnisse zur Effizienz des Greedy-Routing zusammen gefasst (aus [Barr01]):

| $r$-harmonische Verteilung | untere Schranke | obere Schranke |
|---|---|---|
| $0 \leq r < 1$ | $\Omega(n^{\frac{1-r}{2-r}})$ | $O(n^{1-r})$ |
| $r = 1$ | $\Omega(\log^2 n)$ | $O(\log^2 n)$ |
| $1 < r < 2$ | $\Omega(n^{\frac{r-1}{r}})$ | $O(n^{r-1})$ |
| $r = 2$ | $\Omega(\sqrt{n})$ | $O(\frac{n \log\log n}{\log n})$ |
| $2 < r$ | $\Omega(n^{\frac{r-1}{r}})$ | $O(n)$ |

Wir wollen hiervon nur das Hauptergebnis beweisen, dass im Falle $r = 1$ Greedy-Routing nicht mehr als $O(\log^2 n)$ viele Schritte benötigt. Man beachte: Die Tabelle sagt aus, dass in allen anderen Fällen ($r \neq 1$) Greedy-Routing exponentiell viele Schritte brauchen kann (gemessen an $\log n$).

Im Gegensatz zu den Gittermodellen von Kleinberg, bei denen nur im Fall $r = 2$ Greedy-Routing effizient ist (wie oben gezeigt), trifft dieses bei den auf Kreisen basierenden Fernkontakt-Graphen also nur im Falle $r = 1$ zu

Es wird die Bezeichnung *$r$-te harmonische Zahl der Ordnung $n$* für

$$H_n^{(r)} = \sum_{i=1}^{n} i^{-r}$$

verwendet.

**Lemma 10.4-1:** *Für die Zahlen $H_n^{(r)}$ gilt*

$$H_n^{(r)} = \left\{ \begin{array}{c} \frac{1}{1-r} n^{1-r} + O(1) \textit{ falls } r < 1 \\ \log n + O(1) \textit{ falls } r = 1 \\ O(1) \textit{ falls } r > 1 \end{array} \right\}.$$

**Beweis:** Offenbar ergeben sich untere und obere Schranken für $H_n^{(r)}$, wenn man die Funktion $f(x) = x^{-r}$ im Intervall $[1; n]$ integriert. Da $f$ eine fallende Funktion ist, hat man $\sum_{i=2}^{n} i^{-r} \leq \sum_{i=1}^{n} x^{-r} dx \leq \sum_{i=1}^{n-1} i^{-r}$. Nach Berechnung des Integrals führt dies zu

$$\left\{ \begin{array}{c} H_n^{(r)} - 1 \leq \frac{n^{1-r}-1}{1-r} \leq H_n^{(r)} - n^{-r} \text{ falls } r \neq 1 \\ H_n^{(r)} - 1 \leq \log n \leq H_n^{(r)} - n^{-r} \text{ sonst} \end{array} \right\}.$$

Dies ergibt

$$\left\{ \begin{array}{c} \frac{1}{1-r} n^{1-r} + n^{-r} - \frac{1}{1-r} \leq H_n^{(r)} \leq \frac{1}{1-r} n^{1-r} + 1 - \frac{1}{1-r} \text{ falls } r < 1 \\ \log n + n^{-r} \leq H_n^{(1)} \leq \log n + 1 \text{ falls } r = 1 \\ \frac{1}{r-1} + \frac{n^{1-r}}{1-r} + n^{-r} \leq H_n^{(r)} \leq \frac{1}{r-1} + \frac{n^{1-r}}{1-r} + 1 \text{ falls } r > 1 \end{array} \right\}.$$

Daraus folgt die Aussage des Lemmas.

Nun wird die *$r$-harmonische Zufallsvariable* $H_r$ mit Werten in $\{1, 2, ..., n\}$ betrachtet, deren Verteilung durch

$$P(\{H_r = k\}) = \frac{k^{-r}}{H_n^{(r)}}$$

definiert ist.

**Lemma 10.4-2:** *Der Erwartungswert von $H_r$ ist*

$$\mathbf{E}(H_r) = \left\{ \begin{array}{c} \Theta(n) \textit{ falls } 0 \leq r < 1 \\ \Theta(\frac{n}{\log n}) \textit{ falls } r = 1 \\ \Theta(n^{r-1}) \textit{ falls } 1 < r < 2 \\ \Theta(\frac{1}{\log n}) \textit{ falls } r = 2 \\ \Theta(1) \textit{ falls } 2 < r \end{array} \right\}.$$

**Beweis:** Man hat $\mathbf{E}(H_r) = \sum_{k=1}^{n} k \cdot P(\{x = k\}) = \frac{1}{H_n^{(r)}} \sum_{k=1}^{n} k^{1-r}$. Daraus folgt

$$\mathbf{E}(H_r) = \left\{ \begin{array}{c} \frac{1}{H_n^{(r)}} \sum_{k=1}^{n} k^{1-r} \text{ falls } r < 1 \\ \frac{n}{H_n^{(1)}} \text{ falls } r = 1 \\ \frac{H_n^{(r-1)}}{H_n^{(r)}} \text{ falls } 1 < r \end{array} \right\}.$$

Die Aussagen des Lemmas folgen nun aus Lemma 10.4-1 und (im Falle $r < 1$) aus Schranken für $\sum_{k=1}^{n} k^{1-r}$, die ähnlich wie in Lemma 10.4-1 gewonnen werden.

Die nächsten Begriffsbildungen und Hilfssätze sind zunächst allgemein für beliebige Graphen formuliert. Mit ihnen wird es dann gelingen, den angestrebten Satz für Kreise zu beweisen.

Für einen Knoten $u$ eines Graphen $G = (V, E)$ und eine reelle Zahl $r > 0$ wird mit $K_r^G(u)$ die *Kugel vom Radius* $r$ um $u$ bezeichnet, also die Menge aller Knoten aus

$V$, die von $u$ höchstens den Abstand $r$ haben. (Falls ohnehin nur von einem Graphen die Rede ist, schreibt man kurz $K_r(u)$.) Für eine beliebige Knotenmenge $S \subseteq V$ und $u \in V$ setzen wir

$$p\left[u \to S\right] = \sum_{v \in S} p(u,v).$$

Dies kann als die Wahrscheinlichkeit interpretiert werden, dass der Knoten $u$ einen Fernkontakt innerhalb der Menge $S$ besitzt.

**Definition 10.4-3:** *Sei $G$ ein Graph, $p$ eine Wahrscheinlichkeitsabbildung, $c > 1$ eine Konstante, und $f$ eine Funktion. Dann heißt das Paar $(G,p)$ ein $(f,c)$-Fernkontakt-Graph, falls für je zwei Knoten $u$ und $t$ von Abstand höchstens $d$ in $G$ gilt $p\left[u \to K_{\frac{d}{c}}(t)\right] \geq \frac{1}{f(d)}$.*

**Lemma 10.4-3:** *$G = (V,E)$ sei ein Graph von Durchmesser $D$. Ist $(G,p)$ ein $(f,c)$-Fernkontakt-Graph, dann benötigt Greedy-Routing $O(\sum_{i=1}^{\log_c D} f(\frac{D}{c^i}))$ viele erwartete Schritte.*

**Beweis:** Nach Definition ist $p\left[u \to K_{\frac{d}{c}}(t)\right]$ für einen gegebenen Knoten $u$, der höchstens Abstand $d$ vom Zielknoten $t$ hat, die Wahrscheinlichkeit, dass der gewählte Fernkontakt höchstens den Abstand $\frac{d}{c}$ vom Ziel hat. Die erwartete Anzahl von Versuchen bis zum Erfolg ist damit $\frac{1}{p\left[u \to K_{\frac{d}{c}}(t)\right]}$. Wenn ein Versuch fehlschlägt, kann man sich in Richtung des Ziels bewegen, indem zu einem Nachbarn entlang eines kürzesten Weges im Graphen $G$ übergegangen wird. Daher wird der nächste Versuch immer noch in einem Knoten ausgeführt, der höchstens Abstand $d$ von $t$ hat. Aufgrund der Definition 10.4-4 folgt damit für die erwartete Anzahl der Versuche, bis man in $K_{\frac{d}{c}}(t)$ landet:

$$\frac{1}{p\left[u \to K_{\frac{d}{c}}(t)\right]} \leq f(d)$$

Dies bedeutet, dass man ausgehend von $u$ nach höchstens $f(d)$ vielen erwarteten Routing-Schritten in der Kugel $K_{\frac{d}{c}}(t)$ ankommt. Mit der Iteration dieses Arguments kann man nun schließen, dass die erwartete Anzahl von Schritten für das Routing höchstens $O(\sum_{i=1}^{\log_c D} f(\frac{D}{c^i}))$ beträgt.

Für die weitere Argumentation führt die folgende Definition zu einer Vereinfachung.

**Definition 10.4-4:** *Eine Wahrscheinlichkeitsabbildung $p$ auf einem Graphen $G$ ist abstandsinvariant, falls der Wert von $p(u,v)$ nur von dem Abstand $d(u,v)$ abhängt. Eine abstandsinvariante Abbildung heißt fallend, wenn sie als Funktion des Abstands eine fallende Funktion ist.*

*Zur Vereinfachung der Schreibweise kann man bei einer abstandsinvarianten Abbildung nun auch $p(d(u,v))$ statt $p(u,v)$ schreiben.*

**Lemma 10.4-4:** *Ist $p$ eine fallende abstandsinvariante Wahrscheinlichkeitsabbildung auf dem Graphen $G$, dann gilt für alle Knoten $u,t$ mit $d(u,t) \leq d$ und $c > 0$ die Ungleichung $p\left[u \to K_{\frac{d}{c}}(t)\right] \geq p(\frac{(c+1)d}{c}) \cdot |K_{\frac{d}{c}}(t)|$.*

**Beweis:** Es sei $v$ ein Knoten in $K_{\frac{d}{c}}(t)$. Offenbar gilt $d(u,v) \leq d(u,t) + d(t,v) \leq d + \frac{d}{c} = \frac{(c+1)d}{d}$. Damit folgt:

$$p\left[u \to K_{\frac{d}{c}}(t)\right] = \sum_{v \in K_{\frac{d}{c}}(t)} p(u,v) = \sum_{v \in K_{\frac{d}{c}}(t)} p(d(u,v))$$

$$\geq \sum_{v \in K_{\frac{d}{c}}(t)} p(\frac{(c+1)d}{c}) \qquad = p(\frac{(c+1)d}{c}) \cdot |K_{\frac{d}{c}}(t)|.$$

Die $\geq$-Beziehung ergibt sich daraus, dass $p$ als fallend vorausgesetzt ist. Damit ist das Lemma bewiesen.

*Als direkte Konsequenz bekommt man hieraus:*

**Lemma 10.4-5:** *Gegeben seien ein Graph $G$ und eine fallende abstandsinvariante Abbildung $p$. Dann ist für jedes $c > 1$ das Paar $(G,p)$ ein $(f,c)$-Fernkontakt-Graph, wenn die Funktion $f$ definiert ist als*

$$f(d) = \frac{1}{p(\frac{(c+1)d}{c}) \cdot \min_{t \in V} |K_{\frac{d}{c}}(t)|}.$$

Wir können nun das angestrebte Resultat für die "angereicherten Kreise" beweisen:

**Satz 10.4-4:** *Die erwartete Anzahl von Schritten beim Greedy-Routing in Kreisen $C_n$ mit $r$-harmonischen Verteilungen beträgt* $\left\{ \begin{array}{c} O(\log^2 n) \text{ falls } r = 1 \\ O(n^{r-1}) \text{ falls } 1 < r < 2 \end{array} \right\}$.

**Beweis:** Die $r$-harmonischen Abbildungen sind fallend und abstandsinvariant. Aufgrund von Lemma 10.4-5 ist $(G,p_1)$ ein Fernkontakt-Graph mit $f(d) \simeq \log n$. Die $O(\log^2 n)$-Schranke ergibt sich dann aus Lemma 10.4-3. In ähnlicher Weise ist für $r > 1 (G,p_r)$ ein Fernkontakt-Graph mit $f(d) \simeq d^{r-1}$, woraus Lemma 10.4-3 die Behauptung folgt.

Abschließend geht es nun darum, die obigen Ergebnisse für Kreise auf allgemeine Graphen zu verallgemeinern. Es stellt sich heraus, dass eine solche Verallgemeinerung für epimorphe Bilder von Tori (dies wird jetzt genauer erläutert) in der Tat möglich ist.

Zunächst werden die Ergebnisse von Kleinberg für Gitter (siehe oben, Satz 10.4-2) verallgemeinert. Mit $T_q^k$ wird der *$k$-dimensionale Torus* der Kantenlänge $q$ bezeichnet, d. h. er besteht aus $n = q^k$ vielen Ecken.

Im Torus $T_q^k$ hat eine Kugel mit Radius $d$ offenbar die Größe $\Theta(d^k)$, eine Kugeloberfläche $\Theta(d^{k-1})$. Für den Durchmesser gilt $D = \Theta(n^{\frac{1}{k}})$. Wir wollen die $r$-harmonische Verteilung auf $T_q^k$ betrachten. Hier gilt

$$p(d) = \frac{d^{-r}}{\sum_{i=1}^{d} i^{-r}|KO_i(t)|} = \Theta(\frac{d^{-r}}{\sum_{i=1}^{q} i^{-1} \cdot i^{k-r}}).$$

(Mit $KO_i(t)$ ist die Kugeloberfläche im Abstand $i$ von $t$ bezeichnet.) Im Falle $r = k$ ergibt das $p(d) = \Theta(\frac{d^{-k}}{\log q})$. Mit Lemma 10.4-5 wird $(T_q^k, p)$ ein $(f, c)$-Fernkontakt-Graph, wobei

$$f(d) = \frac{1}{p(\frac{3d}{2}) \cdot |K_{\frac{d}{2}}(t)|} = \frac{1}{\Theta(\frac{(\frac{3d}{2})^{-k}}{\log q} \cdot (\frac{d}{2})^{-k})} = \Theta(\frac{3^k}{k} \log n)$$

Mit Lemma 10.4-3 ergibt sich aus diesen Überlegungen:

**Lemma 10.4-6:** *Ist $p_k$ die $k$-harmonische Abbildung auf dem Torus $T_q^k$, so ist $(T_q^k, p_k)$ ein $(f, 2)$-Fernkontakt-Graph, mit $f(d) = \Theta(\frac{3^k}{k} \log n)$. Greedy-Routing benötigt in $(T_q^k, p)$ eine erwartete Anzahl von $O(\frac{3^k}{k^2} \log^2 n)$ vielen Schritten.*

Abschließend gehen wir nun zu allgemeineren Graphen über. Zunächst muss an eine Begriffsbildung aus den Grundlagen der Graphentheorie erinnert werden.

**Definition 10.4-5:** *Zwei Graphen $G = (V, E)$ und $G' = (V', E')$ seien gegeben. Ein Epimorphismus von $G$ auf $G'$ ist eine surjektive Abbildung $\Phi : V \longrightarrow V'$ mit der Eigenschaft, dass Kanten in Kanten übergehen, d. h. aus $\{u, v\} \in E$ folgt stets $\{\Phi(u), \Phi(v)\} \in V'$. $\Phi$ heißt überdies $\alpha$-abstandserhaltend (für eine Konstante $\alpha > 0$), falls stets gilt $d_G(u, v) \leq \alpha \cdot d_{G'}(\Phi(u), \Phi(v))$.*

Man kann sich leicht folgendes überlegen: Ist $p$ eine Wahrscheinlichkeitsabbildung auf $G$ und $\Phi$ ein Epimorphismus auf $G'$, dann ist $p'$ eine Wahrscheinlichkeitsabbildung auf $G'$, wobei

$$p'(u', v') = \frac{1}{|\Phi^{-1}(u')|} \sum_{\substack{u \in \Phi^{-1}(u') \\ v \in \Phi^{-1}(v')}} p(u, v)$$

Das folgende technische Lemma kann nun bewiesen werden.

**Lemma 10.4-7:** *Es existiere ein $\alpha$-abstandserhaltender Epimorphismus von $G$ auf $G'$. $(G, p)$ sei ein $(f, \alpha c)$-Fernkontakt-Graph. Dann ist $(G', p')$ ($p'$ wie oben definiert) ein $(f', c)$-Fernkontakt-Graph, mit $f'(d) = f(\alpha d) \cdot \max_{u' \in V'} |\Phi^{-1}(u')|$.*

Damit können wir den folgenden Satz zeigen.

**Satz 10.4-5:** *Es sei $G = (V, E)$ ein Graph derart, dass ein abstandserhaltender Epimorphismus $\Phi$ eines $k$-dimensionalen Torus der Größe $O(n)$ auf $G$ existiert. Ferner sei angenommen, dass $\max_{v \in V} |\Phi^{-1}(v)| = O(1)$ gilt. Dann gibt es eine Wahrscheinlichkeitsabbildung $p$ auf $G$, so dass Greedy-Routing in $(G, p)$ eine erwartete Anzahl von $O(\frac{3^k}{k} \log^2 n)$ vielen Schritten benötigt.*

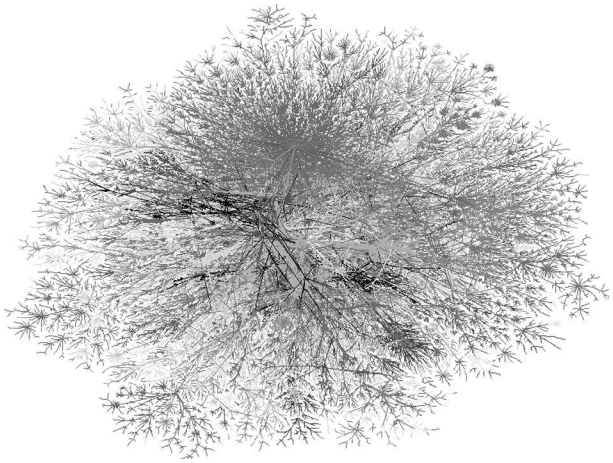
**Beweis:** Aufgrund von Lemma 10.4-6 ist $(T_q^k, p_k)$ ein $(f, 2)$-Fernkontakt-Graph mit $f(d) = \Theta(\frac{3^k}{k} \log n)$. Man kann nun ebenfalls zeigen, dass die oben definierte Abbildung $p'$ einen $(f', 2)$-Fernkontakt-Graphen $(G, p')$ ergibt, wobei $f'(d) \leq \beta \cdot f(\alpha d)$ für geeignete Konstanten $\alpha$ und $\beta$ gilt, d. h. $f'(d) = O(\frac{3^k}{k} \log n)$. Nach Lemma 10.4-3 benötigt damit Greedy-Routing in $(G, p')$ eine erwartete Anzahl von $O(\frac{3^k}{k} \log n)$ vielen Schritten.

**Selbsttestaufgabe 10.4-1:**

*Erläutern Sie den Begriff eines Greedy-Algorithmus. Geben Sie ein Beispiel an, bei dem der offensichtliche Greedy-Algorithmus nicht zum Ziel führt.*

## 10.5 Das Web als Graph

In diesem Abschnitt wollen wir uns mit der Frage beschäftigen, ob auch im Internet und im World Wide Web Small-World-Effekte auftreten. Wie sich zeigt, stoßen wir hier unter anderem auf ein Potenzgesetz für die Eckengrade, welches auch in vielen weiteren Netzen beobachtet wurde. Für Graphen mit dieser Eigenschaft hat sich die Bezeichnung **skalenfrei** durchgesetzt. Solche skalenfreien Graphen sind in den letzten Jahren von zahlreichen Autoren aus unterschiedlichen Fachrichtungen untersucht worden. Einigen Ergebnissen ist der übernächste Abschnitt 10.7 gewidmet.



Das Internet im Jahre 1998

Beginnen wir mit dem Internet, also der Hardware-Struktur aus Leitungen, Routern usw. Die Website Internet Mapping Project [4], die von den Informatikern Bill Cheswick und Hal Burch für die Firma Lumeta (ein Spin-Off von Lucent Technologies) betrieben wird, widmet sich der ständigen Beobachtung der Struktur des Internet und stellt hierzu zahlreiche Diagramme zur Verfügung. Das obige Bild zeigt ein Abbild der Internetstruktur, das im Dezember 1998 in der Zeitschrift Wired abgedruckt wurde. Eine genaue Untersuchung ergibt, dass das Internet in der Tat das Small-World-Phänomen zeigt: Die mittleren Abstände sind recht kurz, und der Cluster-Koeffizient ist um einige Zehnerpotenzen höher, als man es von einem Zufallsnetz erwarten würde. Jedoch ähnelt die Struktur des Internet weder dem Watts-Strogatz-Modell noch dem Modell von Kleinberg, die von uns bisher betrachtet wurden. Wir kommen auf die graphentheoretische Modellierung bald zurück, wenden uns aber zunächst dem WWW zu.

Die Linkstruktur des World Wide Web bildet einen gigantischen Graphen, der den "Internet-Graphen" noch übertrifft. Es wird geschätzt, dass heute im WWW eine Milliarde Dokumente abgelegt sind, täglich werden es mehr. Die Links zwischen Dokumenten bilden die Kanten eines Graphen, wobei man diese Verweise als ungerichtet oder auch gerichtet ansehen kann, da ja stets von einem Dokument zu einem anderen hin (also mit einer Richtung) verwiesen wird. Aufgrund der Größe und der ständigen Änderungen im WWW können die Ecken und Kanten dieses Graphen natürlich niemals vollständig katalogisiert werden. Umso interessanter sind allgemeinere Fragen nach der Topologie dieses Netzes, denn diese Topologie bestimmt letztlich, wie das Web "zusammenhängt" und wie man dort Informationen am besten finden kann. Es liegt auf der Hand, dass Aussagen zur Struktur des Web insbesondere für Suchmaschinen höchst relevant sind.

Bildet auch das Web einen Small-World-Graphen? Die Antwort lautet: Nicht nur das! Es zeigt sich, dass die typischen Eigenschaften eines Small-World-Graphen vorliegen (also gemessen an der Eckenzahl relativ wenige Kanten, geringe Abstände, respektabler Cluster-Koeffizient), jedoch gibt es eine andere Eigenschaft, die ursprünglich von den Brüdern Faloutsos für das Internet entdeckt wurde (vgl. [Fal99]) und sich als äußerst bedeutungsvoll erweist: Es gilt ein *Potenzgesetz für die Eckengrade*.

Zur Erinnerung: In einem ungerichteten Graphen bezeichnen wir für eine Ecke $x$ mit $\gamma(x)$ den *Eckengrad*, also die Anzahl der von $x$ ausgehenden Kanten bzw. der zu $x$ benachbarten Ecken. In einem gerichteten Graphen wird zwischen dem *Innengrad* $\gamma^+(x)$ (in $x$ hineinlaufende Kanten) und dem *Außengrad* $\gamma^-(x)$ (von $x$ ausgehende Kanten) unterschieden.

**Definition 10.5-1:** *Für einen ungerichteten Graphen $G = (E, K)$ mit $n$ vielen Ecken sei $P_i$ ($1 \leq i \leq n-1$) die Anzahl der Ecken mit $i$ vielen Nachbarn. Man setzt ferner $p_i := \frac{P_i}{n}$.*

4 http://research.lumeta.com/ches/map/

Die Zahl $p_i$ kann man offenbar interpretieren als die Wahrscheinlichkeit, dass eine beliebig herausgegriffene Ecke genau $i$ viele Nachbarn hat.

Für Zufallsgraphen kann man die Wahrscheinlichkeit $p_i$ direkt angeben (vgl. z. B. [New03]):

**Lemma 10.5-1:** *In einem Zufallsgraphen $G \in \mathfrak{G}(n,p)$ gilt für beliebiges $i$ ($1 \leq i \leq n-1$)*

$p_i = \binom{n-1}{i} p^i (1-p)^{n-1-i}$.

Setzt man - wie üblich - für wachsendes $n$ den mittleren Grad einer Ecke $z = p(n-1)$ als konstant an, so strebt $p_i$ für $n \to \infty$ gegen $\frac{z^i e^{-z}}{i!}$.

Diese Aussage bedeutet insbesondere, dass für wachsendes $i$ die Wahrscheinlichkeit $p_i$ recht schnell klein wird - anschaulich gesprochen: In einem großen Zufallsgraphen gibt es sehr wenige Ecken mit sehr vielen Nachbarn. Wie sich zeigt, erfasst die folgende Definition Graphen, für die letzteres nur in milderer Form gilt.

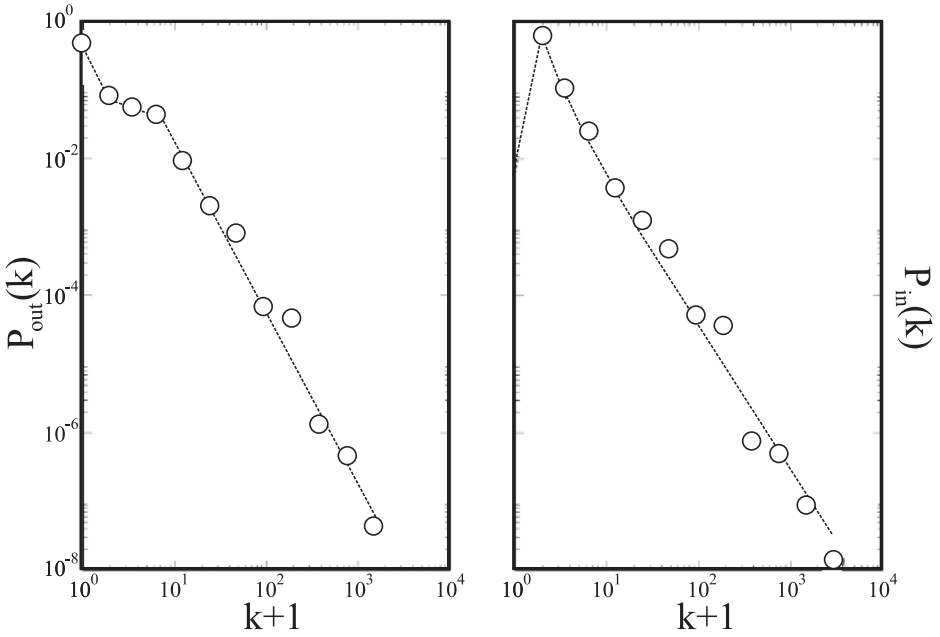
**Definition 10.5-2:** *In dem ungerichteten Graphen $G = (E, K)$ mit $n$ vielen Ecken gilt ein **Potenzgesetz für die Eckengerade**, wenn es eine Konstante $\gamma > 0$ gibt, so dass die Folge der Wahrscheinlichkeiten $p_i$ proportional ist zu $i^{-\gamma}$.*

Man beachte: Gilt in einem Graphen ein Potenzgesetz für die Eckengrade, so nimmt die Anzahl der Ecken mit Grad $i$ für wachsendes $i$ selbstverständlich ab, jedoch lange nicht so rapide wie in einem Zufallsgraphen. Dabei lässt man meistens zu (insbesondere bei sehr großen Graphen), dass das Potenzgesetz erst ab einem gewissen $i_0$ gilt, d. h. die Folge der Wahrscheinlichkeiten $p_i$ mit $i \geq i_0$ ist proportional zu $i^{-\gamma}$. Um endliche Graphen nicht von vornherein auszuschliessen, kann man auch zulassen, dass ab einem gewissen $j$ die Werte $p_i\,(i \geq j)$ Null sind.

Die Aussage, auf die wir hinaus wollen, ist nun die folgende: *Im World Wide Web gilt ein Potenzgesetz für die Eckengrade*[5].

Dies ist selbstverständlich kein mathematischer Satz, sondern eine empirische Aussage, die auf mehreren Untersuchungen basiert. In der Arbeit von Barabasi et al. ( [Bara00]) wird beschrieben, wie mit Hilfe eines Computerprogramms (hier als "Roboter" bezeichnet) die Domäne *nd.edu* der Notre Dame University mit allen ihren Webseiten samt der aus- und eingehenden Links untersucht wurde. Es handelte sich um 325.729 Webseiten und 1.469.680 Links. Die Richtung der Links wurde mit berücksichtigt (d. h. der gerichtete Graph betrachtet). In den beiden folgenden Grafiken, die dem zitierten Papier entnommen sind, ist zu sehen, dass in der Auswertung dieser Daten sowohl $P_{out}(k)$ als auch $P_{in}(k)$ einem Potenzgesetz folgen; die eingezeichneten Linien führen zu den Werten $\gamma_{out} = 2,45$ und $\gamma_{in} = 2,1$.

5 Dies gilt auch für die Struktur des Internet selbst, jedoch konzentrieren wir uns jetzt auf das World Wide Web




Ein Beleg für Potenzgesetze bei ein- und ausgehenden Links (nach [Bara00])

Zu ähnlichen Ergebnissen kommen Broder et al. in [Bro00], die aufgrund von "web crawls" ebenfalls auf $\gamma_{in} = 2,1$ und für $\gamma_{out}$ auf den etwas größeren Wert 2,72 stoßen. Bemerkenswert ist allerdings, dass bei den Außengraden das Potenzgesetz erst ab einer gewissen Größe zu beobachten ist, zu Beginn ist die Abnahme weniger steil.

In den zitierten Arbeiten zur empirischen Untersuchung der Web-Struktur werden neben den Eckengraden zahlreiche weitere Eigenschaften unter die Lupe genommen, so z. B. auch der mittlere Abstand. Die Autoren von [Bar00] gehen dabei so vor: Um ein Modell des WWW zu erzeugen, werden gerichtete Zufallsgraphen mit $n$ vielen Ecken untersucht, die jedoch der Bedingung genügen, dass sich die Außen- und Innengrade der beteiligten Ecken gemäß den Ergebnissen aus den zitierten Untersuchungen verhalten, d. h. $\gamma_{out} = 2,45$ und $\gamma_{in} = 2,1$. Das erstaunliche Ergebnis ist, dass sich als mittlerer Abstand $l$ zwischen zwei beliebigen Ecken herausstellt:

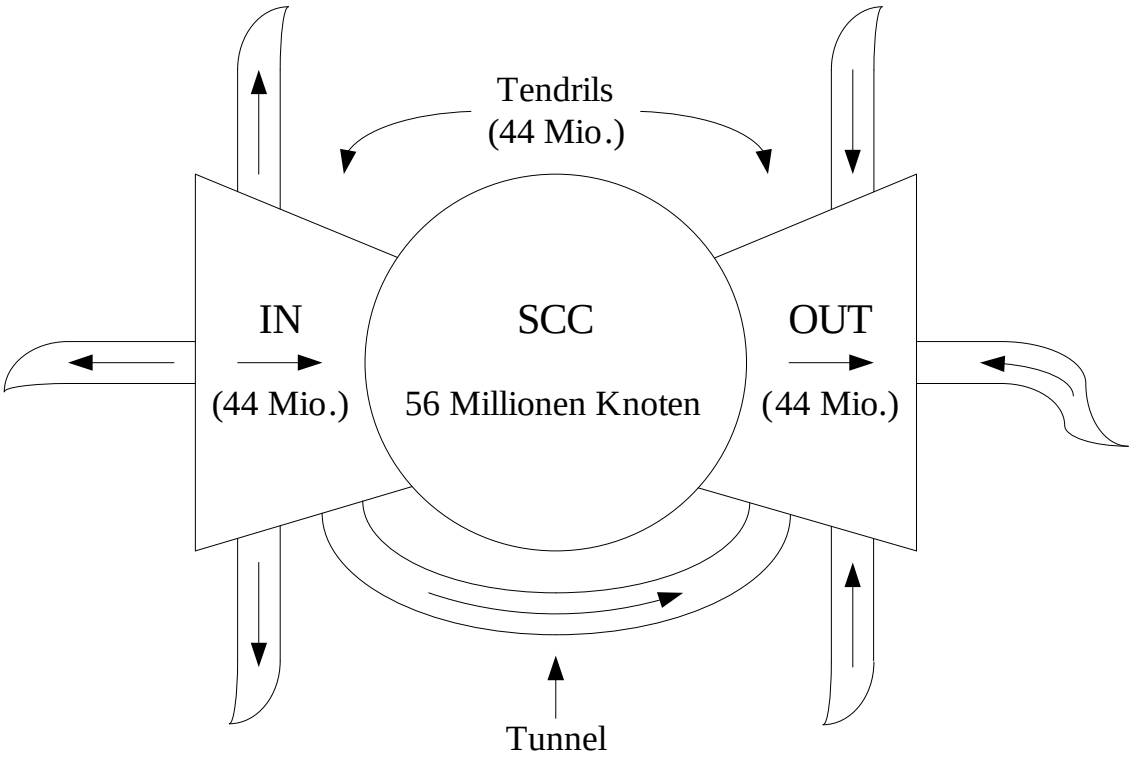
$$l = 0,35 + 2,06 \cdot \log(n)$$

Setzt man hier $n = 8 \cdot 10^8$ ein (für das Jahr 2000 geschätzte Anzahl von Dokumenten im WWW), so ergibt sich $l = 18,59$, was von den Autoren so interpretiert wird: *Zwei Dokumente im World Wide Web sind im Schnitt 19 "Klicks" voneinander entfernt.* Interessant ist hier vor allem das Wachstum mit $\log(n)$, was z. B. bedeutet, dass ein Faktor 10 bei $n$ sich als nur um den Summanden 2 vergrößerte mittlere Weglänge niederschlägt.

Zu leicht anderen Ergebnissen kommen die Autoren in [Bro00], die ihre per "web crawl" erhaltenen Daten auch für die Bestimmung der mittleren Weglänge ausnutzen. Sie erhalten für Webseiten, zwischen denen eine Kette von Links *existiert*, eine mittlere (gerichtete) Weglänge von ca. 16; werden die Links als ungerichtete Kanten angesehen, ist der mittlere Abstand sogar nur ca. 7.

Ein weiterer Untersuchungsgegenstand in [Bro00] ist die **Konnektivität** des Web. Die Untersuchung ermittelt eine aus ca. 186 Millionen Knoten bestehende Zusammenhangskomponente im *ungerichteten* Web, die 91 Prozent der betrachteten Knoten umfasst - der Rest sind kleinere isolierte Komponenten. Betrachtet man diese "Riesenkomponente" (im Englischen *giant component*) genauer, so ergibt sich die im folgenden Bild dargestellte Situation.

Mit **SCC** (für: strongly connected component) ist die größte stark zusammenhängende Komponente (im Sinne der Theorie gerichteter Graphen) bezeichnet. Bezieht man die anderen Knoten auf diese Komponente, so hat man ferner eine Knotenmenge IN mit der Eigenschaft, dass von jedem Knoten in IN aus ein gerichteter Weg in SCC hinein (und damit zu jedem Knoten von SCC) existiert; analog ergibt sich eine Knotenmenge OUT, bei der jeder Knoten aus SCC heraus erreichbar ist. An IN und OUT "hängen" sogenannte **Tendrils** (zu Deutsch: Ranken), die aus IN heraus erreichbar sind bzw. von denen aus man in OUT hineinkommt, ohne durch SCC zu passieren. Ferner kommt es vor, dass eine Ranke aus IN heraus in eine in OUT hineinlaufende übergeht, ohne SCC zu berühren - hier spricht man von einem **Tunnel**. (Kleinere Komponenten, die zu dieser Gesamtkonfiguration keinen Zusammenhang haben, sind in dem Bild nicht dargestellt.)




Komponenten des Web

Als interessante Frage bleibt, ob es möglich ist, ein graphentheoretisches Modell anzugeben, mit dem man auf abstrakte Weise die Entstehung und Struktur des WWW nachbilden und so transparent machen kann. Darauf gehen wir im übernächsten Abschnitt näher ein, in dem skalenfreie Graphen untersucht werden. Unter

anderem werden Untersuchungen zu "Epidemien" in skalenfreien Graphen angesprochen, die sich auf die Problematik der Ausbreitung von Viren im Internet anwenden lassen.

Zum Abschluss erwähnen wir - stellvertretend für eine Reihe von Untersuchungen - die Ergebnisse von Adamic (vgl. [Ada99] ). Hier wird die Small-World-Eigenschaft des WWW verwendet, um Schlüsse für verbesserte Strategien für Suchmaschinen zu ziehen. Die Grundidee für die Nutzung von Ergebnissen zur Konnektivität des Web durch Suchmaschinen besteht darin, aus dem gerichteten Graphen aller Dokumente, die für den Suchbegriff relevant erscheinen, die größte *stark zusammenhängende Komponente* (im Sinne der Theorie gerichteter Graphen) herauszusuchen und dort ein *zentrales Element* auszuwählen, welches an die erste Stelle der Suchergebnisse zu setzen ist. Dies leuchtet sicher auch ohne Graphentheorie als sinnvolle Strategie ein - die Small-World-Ergebnisse besagen hier jedoch, dass sich der entsprechende Aufwand in Grenzen hält. Auf weitere Details gehen wir nicht ein.

**Selbsttestaufgabe 10.5-1:**

*Zeigen Sie: Mit den Bezeichnungen des Bildes "Komponenten des Web" in Abschnitt 10.5 ist es nicht möglich, dass eine gerichtete Kante von einem Knoten der Menge $OUT$ zu einem Knoten der Menge $IN$ existiert.*

## 10.6 Der Graphen-Erzeugungsprozess nach Barabasi, Albert und Jeong

Wie im vorigen Abschnitt angekündigt, ist es nun das Ziel, ein mathematisches Modell für Small-World-Graphen aufzustellen, in denen ein Potenzgesetz für die Eckengrade gilt. Mit diesem Modell kann dann – so die Absicht – auch die Struktur des World Wide Web untersucht werden. Das Modell (vgl. [Bara00]) weicht in zwei entscheidenden Punkten von den bisher beschriebenen Modellen (Watts-Strogatz, Kleinberg) ab:

- Es wird nicht von einer festen Eckenzahl ausgegangen, vielmehr wird der Graph schrittweise erweitert.
- Neue Kanten werden nicht nach einer festen Wahrscheinlichkeit zwischen beliebigen Ecken gezogen, sondern es werden Kanten zwischen solchen Ecken bevorzugt, die schon relativ viele Kanten haben. ("Die Reichen werden reicher.")

Man beachte, dass diese beiden Eigenschaften auf die Link-Struktur des World Wide Web zutreffen: Ständig kommen neue Webseiten hinzu, und die von dort aus aufgebauten neuen Links verweisen häufiger auf "wichtige" Webseiten, auf die bereits oft verwiesen wird.

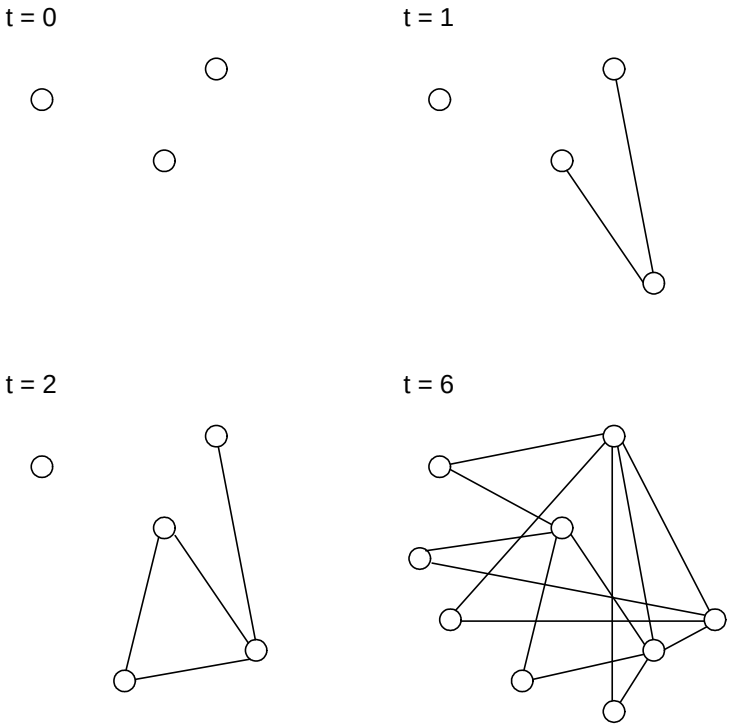
Das Modell wird nun zunächst so beschrieben, wie es in der Originalarbeit von A. Barabasi, R. Albert und H. Jeong vorgestellt wird - die Autoren sind der statistischen Physik zuzuordnen. Dem aufmerksamen Leser wird dabei auffallen, dass die gewohnte mathematische Strenge hier nicht immer eingehalten wird - dies gilt sowohl bei der Beschreibung des Modells wie beim Beweis von Satz 10.6-1. Im nächsten Abschnitt über skalenfreie Graphen wird das Modell noch einmal aus streng mathematischer Perspektive aufgegriffen.

**Beschreibung des Modells** Ausgegangen wird von einer Anzahl $m_0$ von isolierten Ecken. Nun wird in jedem Schritt eine weitere Ecke hinzugefügt, die mit $m$ vielen der bisherigen Ecken durch jeweils eine neue Kante verbunden wird. Diese $m$ vielen Ecken werden zufällig ausgewählt, wobei die Wahrscheinlichkeit $\Pi_e$, dass eine bestimmte Ecke $e$ ausgewählt wird, proportional zum bisherigen Eckengrad $\gamma(e)$ ist, also:

$$\Pi_e = \frac{\gamma(e)}{\sum_{x \in E} \gamma(x)}$$

Nach $t$ Schritten führt dieses Verfahren zu einem zufälligen Graphen mit $m_0 + t$ vielen Ecken und $mt$ vielen Kanten.

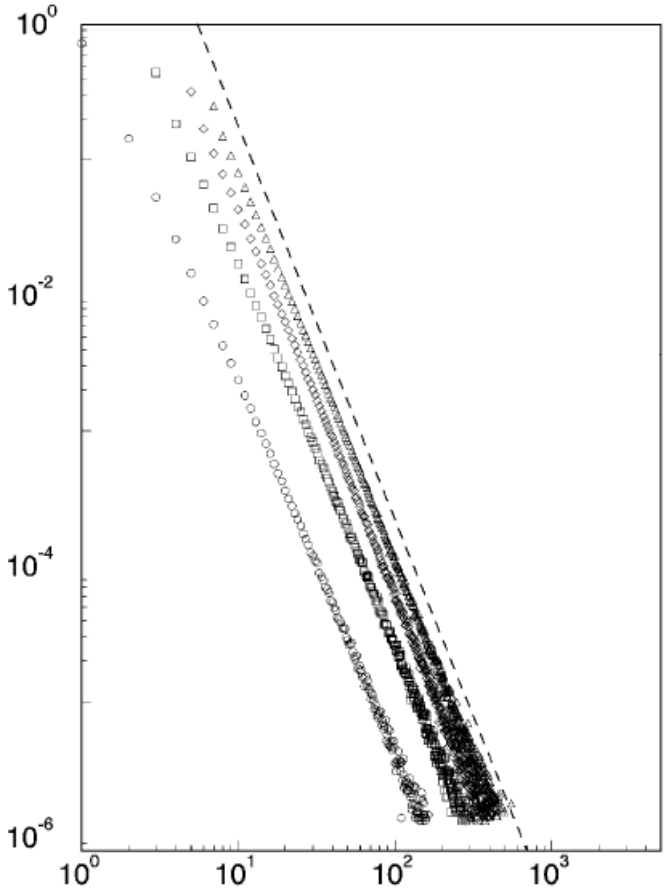
Im nächsten Bild ist eine beispielhafte Abfolge der entstehenden Graphen mit $m_0 = 3$ und $m = 2$ dargestellt.




Eine mögliche Folge von Graphen nach dem Modell von Barabasi, Albert und Jeong

In [Bara00] sind die Ergebnisse einer Reihe von Computersimulationen vorgestellt, in denen der hier beschriebene Erzeugungsprozess einer zufälligen Folge von Graphen nachgebildet wurde. Ergebnis ist, dass für wachsendes $t$ stets Graphen entstehen, für die ein Potenzgesetz für die Eckengrade mit $\gamma \approx 2,9$ gilt - und zwar unabhängig von der Wahl von $m_0$ (Anzahl der Anfangsecken) und $m$ (Anzahl der Kanten pro neuer Ecke). Die Graphen werden für großes $t$ hinsichtlich der Wahrscheinlichkeiten der Eckengrade stationär, unabhängig von der Größe von $m_0$ und $m$.

In der Grafik sind die erwähnten Simulationsergebnisse grafisch dargestellt.




Potenzgesetz der Eckengrade beim Modell von Barabasi, Albert und Jeong

In [Bara99-2] werden die Simulationsergebnisse theoretisch untermauert. Tatsächlich gelingt es, für den Graphenerzeugungsprozess - allerdings nur mit einer Reihe zusätzlicher Annahmen - $\gamma = 3$ nachzuweisen. Aus der Argumentation in [Bara99-2] fomulieren wir den folgenden Satz:

**Satz 10.6-1:** *Wird bei einem Graphenerzeugungsprozess nach Barabasi, Albert und Jeong (mit $m_0$ und $m$) die Anzahl $t$ der Schritte sehr groß, so ergibt sich für die Wahrscheinlichkeiten $p(k)$ asymptotisch $p(k) = \frac{2m^2}{k^3}$.*

**Beweis:** Für den Eckengrad $\gamma_e(t)$ der Ecke $e$ in Abhängigkeit des Schrittes $t$ gilt offenbar $\frac{\partial \gamma_e}{\partial t} = \frac{\gamma_e}{2t}$. Lösung dieser Differentialgleichung ist $\gamma_e(t) = C\sqrt{t}$ mit einer noch zu bestimmenden Konstante $C$.

Mit den Anfangsbedingungen ergibt sich daraus $\gamma_e(t) = m\sqrt{\frac{t}{t_i}}$, wobei $t_i$ den Schritt bezeichnet, in dem die Ecke $e$ hinzugefügt wurde. Damit gilt für die Wahrscheinlichkeit, dass Ecke $e$ weniger als $k$ viele Nachbarn hat, $P(\gamma_e(t) < k) = P(t_i > \frac{m^2 t}{k^2})$. Unter der Voraussetzung, dass neue Ecken in gleichen Zeitintervallen hinzugefügt werden, erhält man hieraus $P(t_i > \frac{m^2 t}{k^2}) = 1 - P(t_i \leq \frac{m^2 t}{k^2}) = 1 - \frac{m^2 t}{k^2(t+m_0)}$. Da sich die Wahrscheinlichkeitsdichte $p(k)$ ergibt als $p(k) = \frac{\partial P(\gamma_e(t)<k)}{\partial k}$, erhält man hier $p(k) = 2\frac{tm^2}{t+m_0}k^{-3}$, was für $t \to \infty$ gegen $\frac{2m^2}{k^3}$ strebt.

Es stellt sich heraus, dass die mittlere Weglänge in den hier betrachteten Graphen mit $\frac{logn}{loglogn}$ wächst - dazu mehr im folgenden Abschnitt.

Anmerkung: In der mathematischen Literatur werden der hier beschriebene Graphen-Erzeugungsprozess sowie der letzte Beweis (Satz 10.6-1) als "heuristisch" kritisiert. Der Grund liegt darin, dass die Formulierung des Erzeugungsprozesses nicht ganz präzise ist (z. B.: Werden die $m$ neuen Kanten nacheinander oder gemeinsam zusammen hinzugefügt?) und dass im Beweis die Funktion $\gamma_e(t)$ als überall definiert und differenzierbar vorausgesetzt wird. Im nächsten Abschnitt werden einige der vorgeschlagenen mathematischen Modelle angesprochen, die ebenfalls auf dem Prinzip "Die Reichen werden reicher" basieren. Die sich dabei ergebenden Resultate beinhalten stets ähnliche Aussagen wie Satz 10.6-1.

Das Modell von Barabasi, Albert und Jeong scheint geeignet, das in vielen realen Netzen annähernd geltende Potenzgesetz für die Eckengrade sowie sehr kurze Weglängen zu erklären. Allerdings finden sich die Voraussagen des Modells über die Größe von $\gamma$ in diesen Netzen in der Regel so nicht wieder, es gibt noch große Diskrepanzen. Korrekturen am Modell scheinen nötig, die das Modell komplexer machen - auch dazu mehr im nächsten Abschnitt.

**Selbsttestaufgabe 10.6-1:**

*Begründen Sie: In einem sehr großen Graphen, der nach dem Modell von Barabasi, Albert und Jeong erzeugt wurde, beträgt der durchschnittliche Eckengrad etwa $2m$.*

## 10.7 Skalenfreie Graphen

Skalenfreie Graphen sind in den letzten Jahren Gegenstand zahlreicher Untersuchungen gewesen, wobei die Eigenschaft der "Skalenfreiheit" nicht immer präzise definiert wird. Wir gehen von der folgenden einfachen Begriffsbildung aus.

**Definition 10.7-1:** *Ein ungerichteter Graph heißt **skalenfrei**, wenn in ihm ein Potenzgesetz für die Eckengrade gilt.*

Man könnte nun beliebige endliche Graphen auf Skalenfreiheit untersuchen bzw. die Frage stellen, für welche Eckenanzahl $n$ und Kantenanzahl $m$ ein skalenfreier Graph überhaupt existiert. Eine präzise Proportionalität zwischen der Verteilung der Eckengrade und einer Folge der Form $k^{-\gamma}$ dürfte dabei selten auftreten.

Im Zusammenhang mit dem Thema Kleine Welten, um welches es in dem vorliegenden Kapitel geht, sind solche Einzelfragen freilich recht uninteressant. Vielmehr werden entweder *unendliche* Graphen betrachtet, für die ein Potenzgesetz durchaus gelten kann oder aber *sehr große endliche* Graphen, für die ein Potenzgesetz näherungsweise gilt (Graphen mit sehr vielen – mehreren Millionen – Ecken und Kanten werden mitunter als **massive Graphen** bezeichnet).

**massive Graphen**

Die Fragen, mit denen wir uns hier weiter beschäftigen wollen, lauten:

- Gibt es strukturelle Eigenschaften, die massiven skalenfreien Graphen gemeinsam sind?
- Welche Modelle des Graphenwachstums führen zu skalenfreien Graphen?

Vor dem Hintergrund, dass auch das Internet und das World Wide Web auf einer skalenfreien Graphenstruktur zu basieren scheinen, ist ferner die folgende Frage von Interesse:

- Haben skalenfreie Graphen hinsichtlich der Ausbreitung von Epidemien besondere Eigenschaften?

Bevor auf diese Fragen weiter eingegangen wird, wollen wir kurz erläutern, woher die Bezeichnung *skalenfrei* stammt. Der Begriff kommt aus der Physik [6], wo man Funktionen der Form

$$f(x) = c \cdot x^{-(1+\alpha)}$$

skalenfrei nennt, da hier stets die Beziehung

$$f(ax) = g(a) \cdot f(x) (\text{mit geeigneter Funktion } g)$$

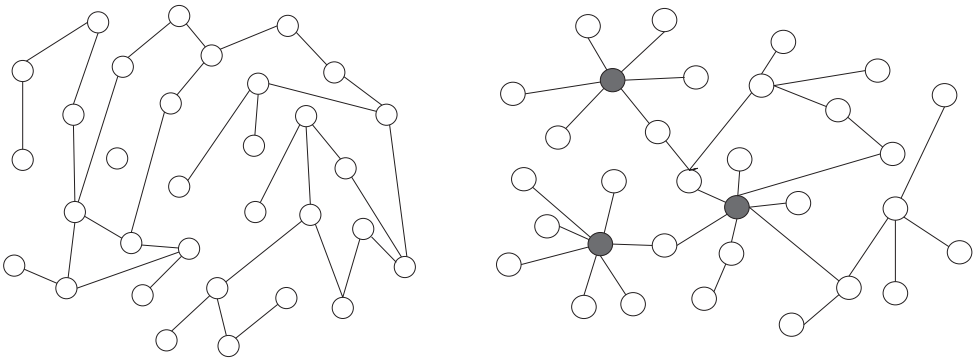
gilt: Streckt oder staucht man die $x$-Achse mit einem konstanten Faktor, so wird der Funktionsgraph ebenfalls getreckt oder gestaucht, ohne seinen "qualitativen Verlauf" zu ändern.

Skalenfreie Graphen sind in den letzten Jahren in den Mittelpunkt des Interesses gerückt, weil sich herausgestellt hat, dass sie als Modelle für viele reale komplexe Netze geeignet sind - dort sind Potenzgesetze für die Eckengrade festgestellt worden. Als *eine* mögliche Erklärung dafür gelten Graphen-Wachstumsmodelle, bei denen (ähnlich dem im vorigen Abschnitt beschriebenen Modell von Barabasi und

6 Es fällt sicher auf, dass die meisten Literaturzitate auf Physik-Zeitschriften verweisen.

anderen) neue Knoten sich bevorzugt mit solchen mit bereits hohem Eckengrad verbinden ("preferential attachment") - auf ein präzises Modell eines solchen Prozesses wird noch eingegangen.

Im folgenden werden einige der interessantesten Ergebnisse über skalenfreie Graphen vorgestellt. Dabei wird auf einige Beweise verzichtet, da diese mitunter sehr komplexe wahrscheinlichkeitstheoretische Argumente verwenden und in den jeweiligen Originalarbeiten viele Seiten beanspruchen - die kompletten Beweise würden den Rahmen des vorliegenden Kurses sprengen.



Zufallsgraph und skalenfreier Graph - das Auftreten von Hubs

**Eigenschaften skalenfreier Graphen**

Skalenfreie Graphen besitzen gegenüber den Zufallsgraphen eine Reihe von Eigenschaften, die in verschiedener Hinsicht interessant sind, jedoch nicht schon *auf den ersten Blick* aus dem Potenzgesetz für die Eckengrade folgen.

Beispielsweise führt die gegenüber einem Zufallsgraphen langsamere Abnahme der Wahrscheinlichkeit, dass eine herausgegriffene Ecke einen hohen Eckengrad hat, dazu, dass eine höhere Anzahl von Ecken mit hohem Grad zu erwarten ist - solche Ecken werden **Hubs** genannt.

Um dies zu illustrieren, sind in der obigen Grafik typische Vertreter eines Zufallsgraphen und eines skalenfreien Graphen dargestellt. Aus dieser Eigenschaft ergibt sich sofort eine Folgerung, die für das Internet und das Web offenbar von Bedeutung ist:

- Ein skalenfreies Netz ist robust gegenüber zufälligen Ausfällen, da im Schnitt kein Hub betroffen ist.
- Ein skalenfreies Netz ist höchst anfällig gegenüber zielgerichteten Angriffen, da der Ausfall von Hubs leicht den Zusammenhang des Netzes gefährden kann.

Was kann über den Durchmesser skalenfreier Graphen gesagt werden?

Wir beginnen mit Ergebnissen, die in dem Papier [Coh03] von R. Cohen und S. Havlin mit dem bezeichnenden Titel "Scale-free networks are ultrasmall" abgeleitet wurden. Da die Beweisführung nicht auf einem präzisen mathematischen Wahrscheinlichkeitsmodell beruht (ähnlich wie bei Satz 10.6-1), wollen wir die Ergebnisse nicht als "Satz" formulieren. (Die Ergebnisse werden allerdings durch die mathematischen Modelle, die bald angesprochen werden, bestätigt.) Sie lauten:

In skalenfreien Netzen mit $p(k) \propto k^{-\lambda}$ gilt für den Durchmesser $d$:

- $d \propto loglogn$ falls $2 < \lambda < 3$
- $d \propto \frac{logn}{loglogn}$ falls $\lambda = 3$
- $d \propto logn$ falls $\lambda > 3$.

Eine weitere allgemeine Eigenschaft, die man in großen skalenfreien Netzen beobachten kann, betrifft die Größe der Zusammenhangskomponenten. Hier gibt es das erstaunliche Ergebnis, dass mit hoher Wahrscheinlichkeit *eine* sehr große ("gigantische") Komponente existiert, daneben einige kleinere Komponenten. Auch auf diesen Punkt gehen wir noch ein.

**Erzeugungsprozesse für skalenfreie Graphen**

Als Vorbereitung auf die Thematik der skalenfreien Graphen haben wir in Abschnitt 10.6 den Graphen-Erzeugungsprozess nach Barabasi, Albert und Jeong vorgestellt, der sozusagen die "Initialzündung" für zahlreiche in den Folgejahren vorgeschlagene mathematische Modelle gewesen ist, deren Intention darin besteht, das Phänomen der Skalenfreiheit in zahlreichen realen Netzen zu "erklären". Wir wollen hier auf zwei dieser Modelle kurz eingehen sowie einige weitere zumindest erwähnen.

Wir beginnen mit dem **Modell von Bollobas und Riordan** (siehe [Bol01], [Bol04]).

Bei diesem Graphen-Erzeugungsprozess werden (wie im Barabasi-Modell) in jedem Schritt eine neue Ecke und $m$ viele neue Kanten hinzugefügt - und zwar werden die neuen Kanten *gleichzeitig* hinzugefügt, was dazu führt, dass man Mehrfachkanten zwischen den Ecken sowie Schleifen erlauben muss.

Begonnen wird mit dem Fall $m = 1$. Der Einfachheit halber geht man von einer festen Folge $v_1, v_2, ...$ von Ecken aus, auf denen der wachsende Graph definiert wird. Nun wird induktiv ein Zufallsgraph-Prozess $(G_1^t)_{t\geq 0}$ definiert, wobei $G_1^t$ jeweils ein Graph auf der Eckenmenge $\{v_i : 1 \leq i \leq t\}$ ist. Gestartet wird mit dem Graphen $G_1^1$ mit einer Ecke und einer Schleife. Ausgehend von $G_1^{t-1}$ wird nun $G_1^t$ gebildet, indem die Ecke $v_t$ sowie eine Kante zwischen $v_t$ und $v_i$ hinzugefügt wird, wobei es für $i$ eine zufällige Auswahl gibt gemäß der Wahrscheinlichkeit

$$\mathbf{P}(i = s) = \left\{ \begin{array}{c} \frac{\gamma_{G_1^{t-1}}(v_s)}{2t-1} \text{ falls } 1 \leq s \leq t-1 \\ \frac{1}{2t-1} \text{ falls } s = t \end{array} \right\}$$

Mit anderen Worten wird $v_t$ mit einer zufällig ausgewählten Ecke $v_i$ durch eine neue Kante $k$ verbunden, wobei die Wahrscheinlichkeit einer Ecke, als dieses $v_i$ ausgewählt zu werden, proportional zu ihrem derzeitigen Grad ist (wie beim Barabasi-Modell) und $k$ bereits beim Eckengrad von $v_t$ mitgezählt wird.

Im Falle $m > 1$ wird $v_t$ mit $m$ vielen Ecken *nacheinander* jeweils durch eine neue Kante k verbunden, wobei in jedem Einzelschritt die bereits zugefügten Kanten sowie die bereits an $v_t$ "hängenden" und noch nicht an einer anderen Ecke befestigten Ecken bei den Eckengraden schon mitgezählt werden. Auf diese Weise hat man nun einen Zufallsgraphen-Prozess $(G_m^t)_{t\geq 0}$ definiert.

Bei der Formulierung der folgenden mathematischen Sätze ist man allerdings weniger an dem beschriebenen Prozess als vielmehr an den Eigenschaften der so entstehenden Graphen interessiert. Dazu bezeichnet man mit $\mathfrak{G}_m^n$ den Wahrscheinlichkeitsraum aller Graphen auf der Eckenmenge $\{v_1, v_2, ..., v_n\}$, wobei zufällig herausgegriffene Graphen $G_m^n \in \mathfrak{G}_m^n$ die sich aus dem obigen Prozess abgeleitete Verteilung besitzen.

Der erste Satz, der dem Papier [Bol01] entnommen ist und den wir ohne Beweis zitieren, bestätigt für das hier diskutierte mathematische Modell die Ergebnisse aus Abschnitt 10.6 zum Barabasi-Modell.

Mit $\#_m^n(d)$ wird ide Anzahl der Ecken von $G_m^n$ mit Innengrad $d$ bezeichnet.

**Satz 10.7-1:** *Sei $m \geq 1$ fest und $(G_m^n)_{n\geq 0}$ der oben definierte Zufallsgraphprozess. Ferner sei $\alpha_{m,d} = \frac{2m(m+1)}{(d+m)(d+m+1)(d+m+2)}$ und $\epsilon > 0$ ebenfalls fest gegeben. Dann gilt für $n \to \infty$ mit $d \leq n^{\frac{1}{15}}$ die Ungleichungskette*

$$(1-\epsilon)\alpha_{m,d} \leq \frac{\#_m^n(d)}{n} \leq (1+\epsilon)\alpha_{m,d}$$

*mit einer gegen 1 konvergierenden Wahrscheinlichkeit.*

Die Aussage des Satzes bedeutet insbesondere, dass – zumindest tendenziell – eine Proportionalität von $p(d)$ zu $d^{-3}$ vorliegt und man es mit skalenfreien Graphen zu tun hat. Ferner liefert der Satz – wegen des ähnlichen Konstruktionsprinzips – eine Bestätigung der Ergebnisse von Barabasi, Albert und Jeong.

In [Bol04] haben B. Bollobas und O. Riordan für dieselbe Graphenklasse den erwarteten Durchmesser untersucht. Auch hier findet sich eine Bestätigung von Ergebnissen, die zu anderen ähnlichen Methoden erzielt wurden. Ohne Beweis formulieren wir:

**Satz 10.7-2:** *Eine natürliche Zahl $m \geq 2$ und eine positive reelle Zahl $\epsilon$ seien fest gegeben. Dann ist fast jeder Graph $G_m^n \in \mathfrak{G}_m^n$ zusammenhängend und hat einen Durchmesser $diam(G_m^n)$ mit*

$$(1-\epsilon)\frac{log\ n}{log\ log\ n} \leq diam(G_m^n) \leq (1+\epsilon)\frac{log\ n}{log\ log\ n}.$$

Alternative Modelle für Graphen-Erzeugungsprozesse, die zu ähnlichen Ergebnissen führen, sind von W. Aiello, F. Chung und L. Lu untersucht worden – die

**Modelle von Aiello, Chung und Lu** wollen wir hier jedoch nicht näher untersuchen (siehe [Aie02]).

Ein weiterer interessanter Ansatz in [Aie02] betrachtet eine Klasse von Zufallsgraphen, deren Verteilung der Eckengrade von zwei festen Werten $\alpha$ und $\beta$ gesteuert wird:

Man nimmt an, es gebe $y$ viele Ecken vom Grad $x > 0$, wobei $x$ und $y$ gemäß

$$logy = \alpha - \beta \cdot logx$$

verknüpft sind. (Ecken vom Grad 0 werden der Einfachheit halber ausgeschlossen.)

Anders gesagt gilt:

$$|\{v : \gamma(v) = x\}| = y = \frac{e^{\alpha}}{x^{\beta}}$$

Man kann also $\alpha$ als den Logarithmus der Anzahl der Ecken vom Grad 1 und $\beta$ als die log-log-Abnahmerate der Eckenzahl eines festen Grads ansehen.

Betrachten wir nun die Klasse aller so definierten ($\alpha$,$\beta$)-Potenzgesetz-Graphen, wobei jeder der möglichen konkreten Graphen als gleich wahrscheinlich angesehen wird. Es können einige interessante strukturelle Eigenschaften nachgewiesen werden, die mit hoher Wahrscheinlichkeit gelten.

Wir skizzieren ein Ergebnis, das die Zusammenhangskomponenten betrifft. $\beta_0 = 3,47875...$ sei eine Lösung der Gleichung[7]

$$\zeta(\beta - 2) - 2\zeta(\beta - 1) = 0.$$

Folgendes wird unter anderem gezeigt:

- Ist $\beta > \beta_0$, so hat der Zufallsgraph fast sicher *keine* gigantische Komponente.
- Ist $\beta < \beta_0$, so hat der Zufallsgraph fast sicher *genau eine* gigantische Komponente.

Weitere Aussagen beziehen sich auf die wahrscheinlichen Größenordnungen der zweitgrößten Komponente und vieles mehr.

Ein weiterer Graphen-Erzeugungsprozess, der eng an das Barabasi-Modell angelehnt ist, wird in [Ber05] betrachtet und dort mit dem Namen **Ployа-Urnen-Darstellung des Barabasi-Albert-Graphen** bezeichnet. Wir gehen nicht weiter darauf ein, kommen jedoch auf das Papier [Ber05] im Kontext von Viren und Epidemien in skalenfreien Graphen in Kürze noch einmal zurück.

Alle von uns bisher geschilderten bzw. zumindest erwähnten Graphen-Erzeugungsprozesse verwenden auf die eine oder andere Weise einen Mechanismus des "preferential attachment", wie es in der englischsprachigen Literatur heißt - neue Ecken

7 Mit $\zeta(t)$ wird die sogenannte Riemannsche Zetafunktion $\sum_{n=1}^{\infty} \frac{1}{n^t}$ bezeichnet.

werden mit einer höheren Wahrscheinlichkeit mit solchen bisher vorhandenen Ecken verbunden, die schon einen hohen Eckengrad haben.

Eine interessante Idee wird in [Kle99] verfolgt: Hier werden die Nachbarn einer neuen Ecke ausgewählt, indem mit hoher Wahrscheinlichkeit Nachbarn einer bereits vorhandenen Ecke "kopiert" werden - eine Idee, die für Weblinks durchaus plausibel ist. Auch hier kann gezeigt werden, dass sich skalenfreie Graphen ergeben!

**Epidemien in skalenfreien Graphen**

Die Ausbreitung von Krankheiten und die Entstehung von Epidemien sind naheliegenderweise von Seiten unterschiedlicher Wissenschaften untersucht worden - Literatur dazu findet man in der Medizin, der Biologie, der Soziologie sowie der Mathematik und der Informatik. Die Fragestellungen sind typischerweise der folgenden Art:

- Unter welchen Bedingungen breitet sich eine übertragbare Infektion im ganzen Netz aus?
- Wie ist die Situation, wenn jedes infizierte Individuum nach einer gewissen Zeit wieder gesund wird aber gleichwohl andere Individuen angesteckt haben kann?
- Unter welchen Bedingungen und nach welcher Zeit ist die Infektion aus dem Netz wieder verschwunden?

Wir beschränken uns hier auf die Sichtweise der Graphentheorie: Die Ecken eines Graphen stellen die Individuen oder auch die Computer dar, die durch Übertragung längs der existierenden Graphenkanten von bereits infizierten Nachbarn ihrerseits ebenfalls (z. B. mit einem Virus) infiziert werden können.

Wir folgen in unseren weiteren Betrachtungen dem sogenannten **SIS-Modell** (steht für: susceptible-infected-susceptible), bei dem jede Ecke für die Infektion empfänglich ist (susceptible) und nach einer Infektion (infected) mit Sicherheit nach einer festen Zeit geheilt wird und dann wieder empfänglich ist (susceptible). Dieses Modell ist passend für die Vireninfektionen von Computern.

(Beim sogenannten **SIR-Modell** ist jedes Individuum nach einer Infektion und erfolgter Heilung resistent, daher der Buchstabe "R" am Schluss.)

Von folgenden Voraussetzungen wird im Weiteren beim SIS-Modell ausgegangen:

- Der Zustand des Systems (hier: Graph) ändert sich in Zeittakten.

- Zu jedem Zeitpunkt ist eine Ecke entweder gesund oder infiziert.

- In einem Zeittakt wird eine gesunde Ecke mit einer Rate $\nu$ (interpretierbar als Wahrscheinlichkeit) infiziert, falls mindestens einer ihrer Nachbarn infiziert ist (genauer: im vorigen Zeittakt infiziert war).

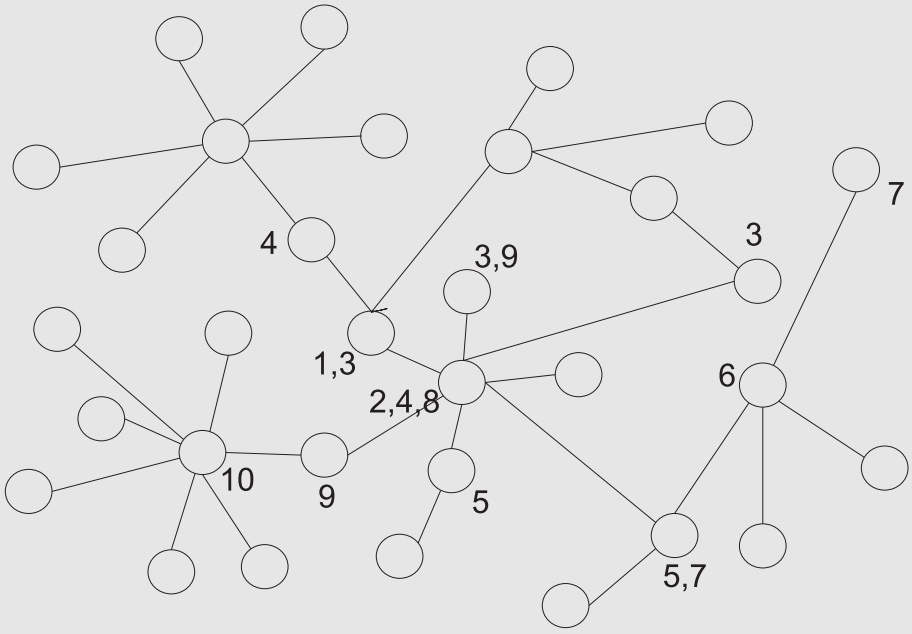
- Eine infizierte Ecke wird mit einer Rate (Wahrscheinlichkeit) $\delta$ im nächsten Zeittakt wieder gesund.

Mit

$$\lambda = \frac{\nu}{\delta}$$

wird die **Ausbreitungsrate** bezeichnet. Dabei ist es üblich, $\delta = 1$ zu setzen, mit anderen Worten wird ein infiziertes Individuum stets im nächsten Zeittakt wieder gesund. Man überlege sich, dass man dies als eine andere Skalierung der Zeitachse auffassen kann – da man ohnehin an qualitativen Aussagen interessiert ist, die sich ergeben, wenn die Zeit "ins Unendliche läuft", macht dies nichts aus.

**Beispiel 10.7-1:**
Die untenstehende Grafik soll einen Ausschnitt eines skalenfreien Graphen darstellen, in dem die Ecken mit einer Ausbreitungsrate von $\lambda = 0,5$ infiziert werden. Es sind nur die Zeittakte von 1 bis 10 dargestellt. Die neben einer Ecke notierten Zahlen sollen besagen, in welchem Zeittakt die betreffende Ecke infiziert ist. Begonnen wird mit einer Infektion der im Zentrum dargestellten Ecke, die eine "1" trägt und in Zeittakt 3 ebenfalls wieder infiziert ist usw. (Da die Infektionen stets mit Wahrscheinlichkeit 0,5 auf Nachbarknoten übergehen, ist der hier dargestellte Ablauf selbstverständlich willkürlich.) In Zeittakt 11 würden nun im Schnitt drei oder vier der sieben Nachbarn des die "10" tragenden Knotens infiziert etc.




Verbreitung eines Virus in 10 Zeittakten

Das zentrale Ergebnis über Epidemien gemäß dem SIS-Modell besagt nun folgendes (siehe [Bai75][Mar99][Mur93]):

*Sind die Eckengrade des zugrunde liegenden Graphen beschränkt (wie z. B. in einem regelmäßigen Gitter, man spricht von lokaler Konnektivität), oder handelt es sich um einen Zufallsgraphen (in dem die Wahrscheinlichkeit hoher Eckengrade exponentiell abnimmt), so gibt es eine epidemische Schranke $\lambda_c > 0$: Ist dann für die*

*betrachtete Infektion das zugehörige $\lambda \geq \lambda_c$, so breitet sich die Infektion epidemisch aus und stabilisiert sich, d. h. es ist stets ein gewisser Prozentsatz der Ecken infiziert. Ist $\lambda < \lambda_c$, so stirbt die Infektion zügig (genauer: exponentiell schnell) aus.*

Nun hat man auf der anderen Seite beobachtet, dass Viren in Computernetzen sehr lange zu "überleben" scheinen, so klein auch ihre Ausbreitungsrate ist. In [Pas01] wird über entsprechende Statistiken berichtet. In [Pas01] wird auch die Erklärung für dieses Phänomen geliefert:

*In skalenfreien Graphen ist $\lambda_c = 0$!*

Aus dem Papier [Pas01], welches ähnlich wie [Bara99-2] im Stil von Physikern recht forsch argumentiert und u. a. die Differenzierbarkeit aller vorkommenden Funktionen voraussetzt, ziehen wir den folgenden Satz, der dort nicht explizit formuliert ist:

**Satz 10.7-3:** *In einem sehr großen (bzw. unendlich großen) Graphen mit $p(k) \propto k^{-3}$ und durchschnittlichem Eckengrad $2m$ (wie bei den Barabasi-Graphen in Abschnitt 10.6) ist $\lambda_c = 0$. Der Prozentsatz der infizierten Knoten konvergiert nach $2e^{-\frac{1}{m\lambda}}$.*

Die Aussage des Satzes bedeutet: So klein die Ausbreitungsrate $\lambda$ einer Infektion in einem skalenfreien Netz auch ist - die Infektion stirbt wahrscheinlich niemals aus, es stellt sich ein stabiler Prozentsatz infizierter Knoten ein. Unter der Prämisse, dass Internet und WWW skalenfreie Netze bilden, "lebt" dort ein Virus tendenziell recht lange.

Beweisidee: Mit $\rho_k(t)$ wird der Prozentsatz infizierter Ecken vom Grad $k$ bezeichnet. In Analogie zu aus der Physik bekannten sogenannten "mean field equations" kommt man zu der Gleichung

$$\partial_t \rho_k(t) = -\rho_k(t) + \lambda k \left[1 - \rho_k(t)\right] \Theta(\lambda).$$

Dabei bezeichnet $\Theta(\lambda)$ die Wahrscheinlichkeit, dass ein betrachteter Endpunkt einer beliebig herausgegriffenen Kante infiziert ist. Geht man nun davon aus, dass ein stationärer Zustand erreicht ist, so folgt mit $\partial_t \rho_k(t) = 0$ für die "stationäre Dichte"

$$\rho_k = \frac{k\lambda\Theta(\lambda)}{1 + k\lambda\Theta(\lambda)}.$$

Zusammen mit der offensichtlich erfüllten Gleichung

$$\Theta(\lambda) = \sum_k \frac{k p_k \rho_k}{\sum_s s p_s}.$$

ergibt sich daraus durch Integration schließlich

$$\Theta(\lambda) = \frac{e^{-\frac{1}{m\lambda}}}{m\lambda}.$$

Für $\rho(t)$ führt dies im Grenzprozess $t \to \infty$ zu

$$\rho = 2e^{-\frac{1}{m\lambda}}.$$

Auch bei dem Thema der Epidemien in skalenfreien Graphen haben sich eine Reihe mathematischer Autoren gefunden, die diese Ergebnisse im Rahmen präziserer mathematischer Modelle bestätigt haben. Stellvertretend zitieren wir im nächsten Satz ein Ergebnis aus [Ber05]. Das Papier enthält auch, wie bereits erwähnt, ein eigenes mathematisches Modell zur Erzeugung großer skalenfreier Graphen, auf dessen Grundlage dann der folgende Satz bewiesen wird:

**Satz 10.7-4:** *Zu jedem $\lambda > 0$ gibt es eine natürliche Zahl $N$, so dass für einen skalenfreien Graphen der Größe $n > N$ und eine darin ausgewählte beliebige Ecke $v$ gilt: mit Wahrscheinlichkeit $1 - O(\lambda^2)$ gilt für $v$, dass eine dort startende Infektion mit einer von unten durch $\lambda^{C_1 \frac{\log(\frac{1}{\lambda})}{\log\log(\frac{1}{\lambda})}}$ und von oben durch $\lambda^{C_2 \frac{\log(\frac{1}{\lambda})}{\log\log(\frac{1}{\lambda})}}$ beschränkten Wahrscheinlichkeit überlebt; dabei sind $C_1$ und $C_2$ Konstanten, die nicht von $\lambda$ oder $n$ abhängen.*

**Das Gradprodukt und skalenfreie Graphen**

Wie zu Anfang dieses Abschnitts erwähnt, gibt es zahlreiche Ansätze, den Begriff der skalenfreien Graphen mathematisch präzise zu erfassen, wobei die meisten Autoren zu einer Begriffsbildung kommen, bei der diese Graphen als Resultate von Zufallsprozessen mit wachsender Eckenzahl entstehen und bei der Entstehung neuer Kanten solche Knoten als Endpunkte bevorzugt werden, die schon viele Nachbarn haben ("preferential attachment"). Da es unterschiedliche solcher Ansätze gibt, sind die erzielten Resultate nicht immer miteinander vereinbar.

Der von uns gewählte Weg, einen beliebigen Graphen bereits dann "skalenfrei" zu nennen, wenn in ihm ein Potenzgesetz für die Eckengrade gilt, wird selten gewählt - obwohl er den Vorteil hat, nicht auf irgendwelchen Zufallsprozessen zu beruhen, so dass man sich sofort auf die strukturellen Eigenschaften solcher Graphen konzentrieren kann. Der Nachteil ist freilich, dass unter diesen schwachen Voraussetzungen viele interessante Eigenschaften, die Graphen wie dem Internet oder dem WWW eigen sind, nicht ohne weitere Voraussetzungen allgemein bewiesen werden können, so dass man Gefahr läuft, sich von den "interessanten" Graphen in seiner Theorie zu entfernen.

Einen interessanten neuen Weg schlagen die Autoren L. Li et al. in [Li05] ein, die zusätzlich zu einem Potenzgesetz für die Eckengrade noch ein großes Gradprodukt fordern.

**Definition 10.7-2:** *Ein endlicher ungerichteter Graph $G = (E, K)$ sei gegeben. Das Gradprodukt $s(G)$ ist definiert als*

$$s(G) := \sum_{(e,f) \in K} \gamma(e) \cdot \gamma(f).$$

$s(G)$ misst, inwieweit in $G$ Ecken von hohem Grad wieder mit solchen Ecken verbunden sind - in der Interpretation: inwieweit $G$ einen "Kern von Hubs" besitzt.

In [Li05] wird nun ferner der Parameter $s_{max}$ betrachtet als der maximal mögliche $s(G)$-Wert, den ein Graph mit der gegebenen Verteilung der Eckengrade haben kann. Mit

$$S(G) := \frac{s(G)}{s_{max}}$$

wird schließlich gemessen, inwieweit $G$ den bei gleicher Verteilung der Eckengrade maximal möglichen $s(G)$-Wert besitzt.

In [Li05] wird vorgeschlagen, neben dem Potenzgesetz für die Eckengrade einen nahe an 1 gelegenen $S(G)$-Wert als Kriterium für Skalenfreiheit auszuwählen. Ferner werden einige erste Ergebnisse vorgestellt, inwieweit für gewisse Graphenklassen skalenfreie Graphen (in diesem Sinne) existieren und wie solche konstruiert werden können.

Auf weitere Details gehen wir nicht ein.

**Selbsttestaufgabe 10.7-1:**

*Begründen Sie, dass in einem endlichen Graphen niemals ein Potenzgesetz für die Eckengrade im Sinne der strengen Definition 10.5-2 gelten kann.*

# 11 Random Walks und Elektrische Netzwerke

## 11.1 Zwei einführende Beispiele

Anhand zweier Beispiele, denen derselbe Graph zugrunde liegt, wird in die Thematik dieses Kapitels eingeführt.

Um das erste Beispiel vollständig nachvollziehen zu können, muss man sich noch einmal eine Konsequenz der Kirchhoffschen Gesetze ins Gedächtnis zurückrufen. Dabei werden die Ecken eines Graphen als Knoten eines elektrischen Netzwerkes aufgefasst, die Kanten als Widerstände einheitlicher Größe $R$. Für eine Ecke $x$ des Graphen bezeichne $N(x)$ die Menge der Nachbarecken.

**Lemma 11.1-1:** *Wird zwischen zwei beliebigen Ecken eines Graphen eine Spannung angelegt und der Stromkreis geschlossen, so gilt für die Spannung $v_x$ in einer beliebigen Ecke $x$ des Graphen mit $|N(x)| = n$ stets*

$$v_x = (1/n) \sum_{y(x)} v_y.$$

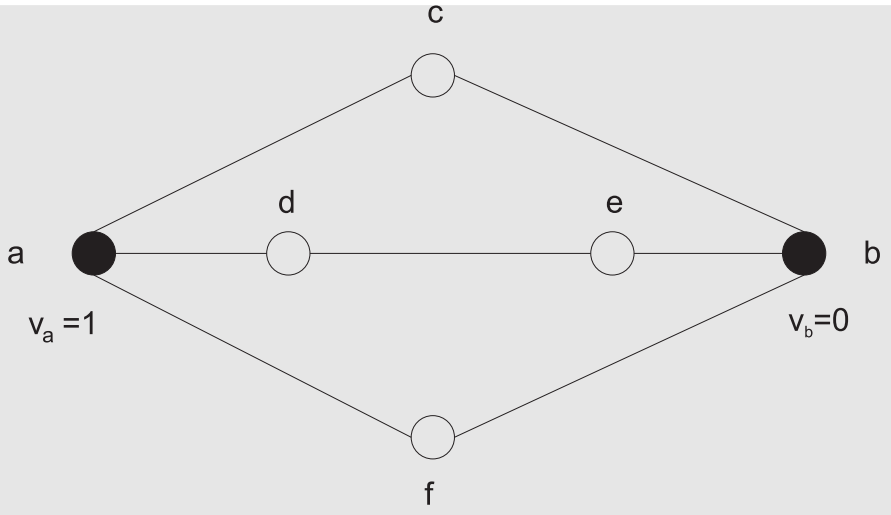
*Die Spannung ist also immer das arithmetische Mittel aller Nachbarspannungen.*

**Beweis:** Ist $y$ Nachbar von $x$, so wird mit $i_{yx}$ der von $y$ nach $x$ fließende Strom (eventuell mit negativem Vorzeichen) bezeichnet. Nach den Kirchhoffschen Gesetzen gilt $\sum_{y(x)} i_{yx} = 0$ und somit $\sum_{y(x)}((v_y - v_x)/R) = 0$.

Multiplikation mit $R$ und Herausnehmen von $v_x$ aus der Summe ergibt dann

$\sum_{y(x)} v_y - nv_x = 0$ und daraus $v_x = (1/n) \sum_{y(x)} v_y$.

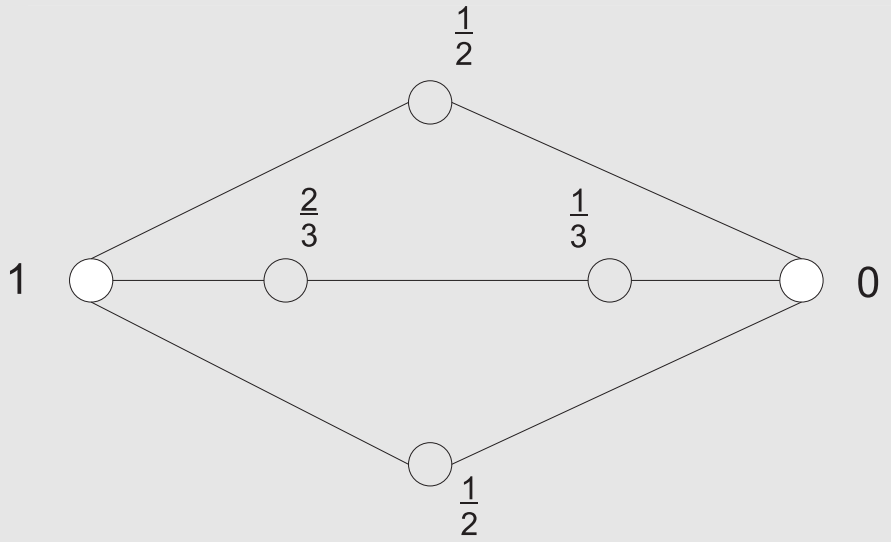
**Beispiel 11.1-1:**
Der in der folgenden Abbildung gezeigte Graph $G$ stelle ein elektrisches Netzwerk dar, in dem die Kanten zwischen den durch sie verbundenen Punkten Widerstände der gleichen Größe sind (also o. B. d. A. Widerstäde von 1 Ohm). Zwischen $a$ und $b$ wird eine Spannung von 1 Volt angelegt, $v_a = 1$ und $v_b = 0$. Welche Spannung ergibt sich daraus in den Punkten $c$, $d$, $e$ und $f$?




Ein Graph $G$

Das in der nächsten Abbildung gezeigte Ergebnis $v_c = 1/2$, $v_d = 2/3$, $v_e = 1/3$ und $v_f = 1/2$ lässt sich auf mehrere Arten ermitteln. **Eine** mögliche Argumentation (und für uns die interessanteste) geht so:

Die Spannung in einem Punkt ist stets das arithmetische Mittel der Spannungen in den Nachbarpunkten. Daraus folgt sofort $v_c = v_f = 1/2$ sowie $v_d = (1 + v_e)/2$ und $v_e = v_d/2$, was leicht die obigen Werte ergibt.

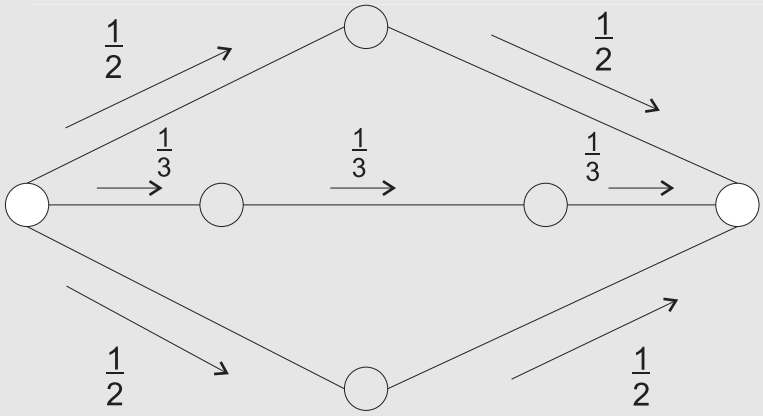



Anliegende elektrische Spannungen im Graphen $G$

Welche Stromstärken werden jeweils auf den Kanten erzeugt, wenn $a$ und $b$ kurzgeschlossen werden und so der Stromkreis geschlossen wird? Da der effektive Widerstand zwischen $a$ und $b$ $R_{eff} = 3/4$ beträgt, folgt

$$i_{ab} = 4/3,$$

m. a. W. strömt ein Fluss der Größe $4/3$ in $a$ hinein und bei $b$ wieder hinaus. Die folgende Abbildung zeigt, welche Ströme sich dadurch auf den einzelnen Kanten ergeben.




Stromstärken auf den Kanten von $G$

**Beispiel 11.1-2:**
Derselbe Graph $G$ aus dem obigen Beispiel stelle nun das Straßennetz eines Dorfes mit Knotenpunkten $a$ bis $f$ dar. Ein Betrunkener starte in einem Punkt $x$ und wähle auf seinem Weg an jeder Kreuzung stets zufällig eine der sich dort kreuzenden Straßen. $w_x$ bezeichne nun die Wahrscheinlichkeit, dass der Betrunkene auf seinem Weg durch den Punkt $a$ kommt, **bevor** er zum ersten Mal durch $b$ kommt. Wie groß sind die Wahrscheinlichkeiten $w_x$ für die unterschiedlichen Knotenpunkte $x$?

Aus dem Satz über die totale Wahrscheinlichkeit folgt, dass jede Wahrscheinlichkeit $w_x$ das arithmetische Mittel der entsprechenden Wahrscheinlichkeiten der Nachbarknoten von $x$ sein muss. Somit kommt man, da offensichtlich $w_a$=1 und $w_b = 0$ ist, zu den Gleichungen

$$w_c = w_f = 1/2,$$

$$w_d = (1 + w_e)/2$$

und

$$w_e = w_d/2,$$

was zu $w_c = w_f = 1/2$, $w_d = 2/3$ und $w_e = 1/3$ führt.

Diese Wahrscheinlichkeiten entsprechen also den im vorigen Beispiel berechneten elektrischen Spannungen.

An den beiden Beispielen ist deutlich geworden, dass offenbar ein Zusammenhang besteht zwischen Random Walks in Graphen und Fragestellungen, die sich erge-

ben, wenn dieselben Graphen als Grundstrukturen elektrischer Netzwerke aufgefasst werden. Dabei geht die Parallelität sogar noch weiter als im zweiten Beispiel angedeutet: Auch die Stromstärken aus dem ersten Beispiel haben bei den Random Walks eine wahrscheinlichkeitstheoretische Interpretation - dazu mehr im nächsten Abschnitt.

Auf eines muss an dieser Stelle hingewiesen werden: Die Parallelität zwischen Random Walks und elektrischen Netzwerken, die zu ähnlichen mathematischen Ergebnissen führt, bedeutet selbstverständlich **nicht**, dass ein realer physikalischer Zusammenhang besteht, der Zusammenhang besteht nur auf der logisch-mathematischen Ebene. Dies ist nicht ungewöhnlich: Daraus, dass zwei Äpfel plus zwei Äpfel vier Äpfel und zwei Autos plus zwei Autos vier Autos ergeben, folgt schließlich ebenfalls **nicht**, dass Äpfel etwas mit Autos zu tun haben.

Trotzdem kann der Zusammenhang ausgenutzt werden: Durch die Untersuchung elektrischer Netzwerke können Aussagen über Random Walks abgeleitet werden - und umgekehrt.

## 11.2 Random Walks in einer und zwei Dimensionen

Die Analogie zwischen Random Walks (im folgenden gelegentlich durch "RW" abgekürzt) und elektrischen Netzwerken kann besonders einfach anhand von Graphenstrukturen erklärt werden, die nur aus einem Pfad bestehen oder ein zweidimensionales Gitter darstellen. Schlüssel sind dabei die **harmonischen Funktionen**.

**harmonische Funktionen**

**Definition 11.2-1:** *Es sei $S$ die Menge der natürlichen Zahlen von $0$ bis $n$, $S = \{0, 1, 2, ..., n\}$. Wir nennen $I = \{1, 2, ..., n-1\}$ die **inneren** und $A = \{0, n\}$ die **äußeren** Punkte von $S$. Die Funktion*

$$f : S \to \mathbb{R}$$

*heißt **harmonisch**, wenn für jeden inneren Punkt $x \in D$ gilt*

$$f(x) = \frac{f(x-1) + f(x+1)}{2}.$$

**Beispiel 11.2-1:**
Der in der untenstehenden Grafik dargestellte Pseudograph mit Schleifen bei den Ecken 0 und 5 stelle Straßenecken dar, die Kanten stehen für Straßen zwischen den Ecken.

Das Szenario: Ein Mann geht in Ecke $x$ los ($x \in \{0, 1, ..., 5\}$) und wählt jedes Mal, wenn er sich an einer Ecke $x \in \{1, ..., 4\}$ befindet (also auch gleich am Anfang) zufällig mit gleicher Wahrscheinlichkeit eine der beiden Richtungen

zur nächsten Nachbarecke. Kommt er in der Bar (Ecke 0) oder zu Hause an (Ecke 5), so bleibt er dort.

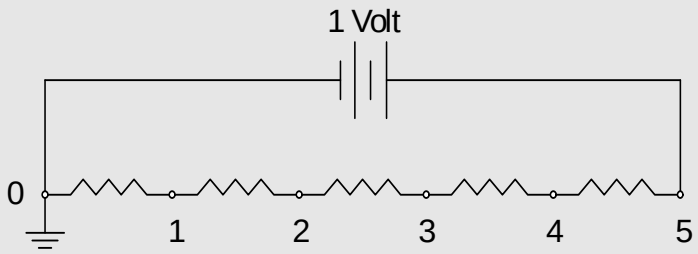
Die Frage: Wie groß ist die Wahrscheinlichkeit $p(x)$, dass der Mann, falls er in der Ecke $x$ losgeht, zu Hause ankommt?

Die Antwort: Offenbar gilt $p(0) = 0$, $p(5) = 1$ und

$$p(x) = 1/2 \cdot p(x-1) + 1/2 \cdot p(x+1)$$

für die restlichen $x$. Daraus folgt **zwingend** $p(x) = x/5$ für alle $x$!

**Beispiel 11.2-2:**
In der folgenden Grafik ist dieselbe Graphenstruktur als elektrisches Netzwerk dargestellt.




Es sind fünf Widerstände gleicher Größe in Reihe geschaltet und an die beiden Enden eine Spannung von 1 Volt angelegt. Gefragt ist nach den Spannungen $v_x$, die sich an den Punkten $x = 0, 1, ..., 5$ ergeben. Dabei wird der Punkt 0 geerdet, so dass $v_0 = 0$ ist. Aufgrund von Lemma 11.1-1 ergibt sich auch hier

$$v_x = 1/2 \cdot v_{x-1} + 1/2 \cdot v_{x+1}$$

mit dem Ergebnis wie im vorigen Beispiel, d. h. $v_x = x/5$.

Die Gleichheit der Ergebnisse in den beiden letzten beiden Beispielen hat eine einfache mathematische Begründung.

**Lemma 11.2-1: (Maximum-Prinzip)**

*Jede harmonische Funktion $f(x)$ nimmt ihren maximalen Wert $M$ und ihren minimalen Wert $m$ in einem äußeren Punkt an.*

**Beweis:** $M$ sei der größte von $f$ angenommene Wert. Falls $f(x) = M$ für einen inneren Punkt $x \in D$ ist, muss auch $f(x-1) = M$ und $f(x+1) = M$ gelten, da $f(x)$ der Mittelwert von $f(x-1)$ und $f(x+1)$ ist. Ist auch $x-1$ innerer Punkt, so führt dasselbe Argument zu $f(x-2) = M$ usw., d h. man kommt so zu $f(0) = M$ und $f(N) = M$. Analog kann man für den minimalen Wert schließen.

**Lemma 11.2-2: (Eindeutigkeits-Prinzip)**

*Eine harmonische Funktion auf $S$ ist durch die Werte für $x \in A$ eindeutig festgelegt.*

**Beweis:** Seien $f(x)$ und $g(x)$ harmonische Funktionen mit $f(x) = g(x)$ für $x \in A$. Eine leichte Rechnung zeigt, dass auch die Funktion

$$h(x) = f(x) - g(x)$$

harmonisch ist. Offenbar gilt $h(x) = 0$ für jedes $x \in A$. Nach dem Maximum-Prinzip folgt daraus $h(x) = 0$ für alle $x$, womit $f(x) = g(x)$ für beliebiges $x$ gezeigt ist.

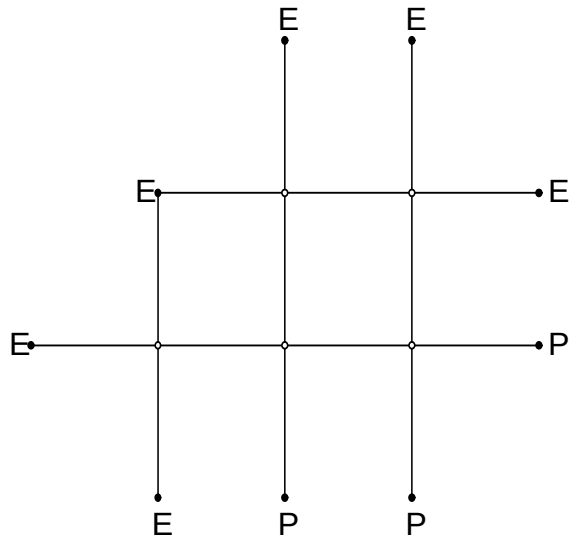
Durch die letzten beiden Lemmas ist klar geworden, dass die Eigenschaften harmonischer Funktionen die tiefere Begründung liefern, dass Random Walks etwas mit elektrischen Netzwerken zu tun haben.

Im nächsten Schritt kann man nun von den eindimensionalen Pfaden zu zweidimensionalen Gittern übergehen. Dies wird an dem folgenden Beispiel aus [Doy84] verdeutlicht.

Wir legen den in der nächsten Abbildung dargestellten Graphen zugrunde. Die Ecken mit vier Nachbarn sind hier die inneren Punkte, die restlichen sind äußere oder Randpunkte. Die äußeren Punkte sind noch einmal in zwei Klassen unterteilt, nämlich "P" für Polizei und "E" für Entkommen.

Die zugehörige Geschichte lautet: Ein Taschendieb flieht vor der Polizei, indem er sich in dem Graphen (Staßennetz) von Ecke zu Ecke bewegt und jeweils als nächste eine zufällige Nachbarecke besucht (also auf einem Random Walk).

Die Frage ist: Wie groß ist die Wahrscheinlichkeit $p(x)$, dass der Dieb, der bei dem inneren Punkt $x$ startet, zuerst eine Randecke des Typs E erreicht (also entkommt) und vorher nicht bei einer P-Ecke landet?




Es liegt auf der Hand, wie der Begriff der harmonischen Funktion verallgemeinert werden muss, um die hier vorliegende Situation einzufangen:

**Definition 11.2-2:** *Sei $S = I \cup A$ eine endliche Menge von Gitterpunkten der Ebene*

$$\{(a,b)|1 \leq a \leq N, 1 \leq b \leq N\}$$

*mit folgenden Eigenschaften:*

- $I \cap A = \varnothing$
- *Jeder Punkt $(a,b)$ von $I$ hat vier Nachbarpunkte $(a-1,b)$, $(a+1,b)$, $(a,b-1)$ und $(a,b+1)$ in $S$.*
- *Jeder Punkt in $A$ hat höchstens drei Nachbarpunkte, von denen mindestens einer zu $I$ gehört.*
- *Zwischen den Randpunkten in $A$ existieren keine Kanten.*

*Der sich so ergebende Graph sei ferner zusammenhängend. Dann heißen die Punkte von $I$ innere und die von $A$ äußere Punkte von $S$. Eine Funktion*

$$f : S \to \mathbb{R}$$

*heißt harmonisch, wenn für jeden Punkt $(a,b) \in I$ gilt*

$$f(a,b) = \frac{f(a-1,b) + f(a+1,b) + f(a,b-1) + f(a,b+1)}{4}.$$

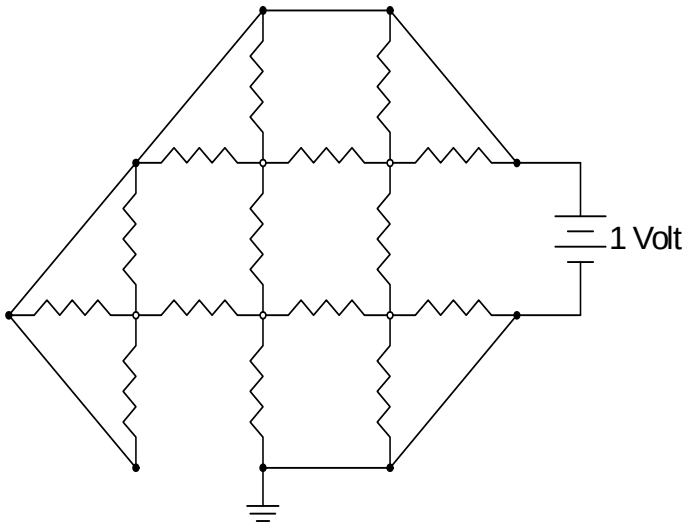
Die entscheidende Beobachtung ist nun auch hier:

**Die Funktion** $p(x)$ (Wahrscheinlichkeit zu entkommen) **ist harmonisch.**

Dies folgt wieder mit dem Satz über die totale Wahrscheinlichkeit. Ebenso lassen sich wieder sowohl das Maximum- als auch das Eindeutigkeits-Prinzip beweisen. Da die $p$-Werte für die Randpunkte feststehen - offenbar ist $p(x) = 1$ für die E-Knoten und $p(x) = 0$ für die P-Knoten -, sind die gesuchten Wahrscheinlichkeiten die restlichen Werte $p(x)$ für $x \in D$.

Nach wie vor bleibt die Frage offen, wie man die Wahrscheinlichkeiten $p(x)$ für die inneren Knoten $x$ denn nun **konkret ausrechnet**. Darauf kommen wir in Kürze zurück.

Zuvor wollen wir auch für dieses Beispiel das Problem der elektrischen Spannungen untersuchen. In derfolgenden Abbildung ist das relevante elektrische Netzwerk dargestellt - dabei sind die P-Knoten geerdet und die E-Knoten mit einer 1-Volt-Batterie verbunden.




Von Interesse sind die Spannungen $v(x)$ für die inneren Punkte $x \in I$. Auch hier hat man wieder, dass die Spannung

$$v : S \to \mathbb{R}$$

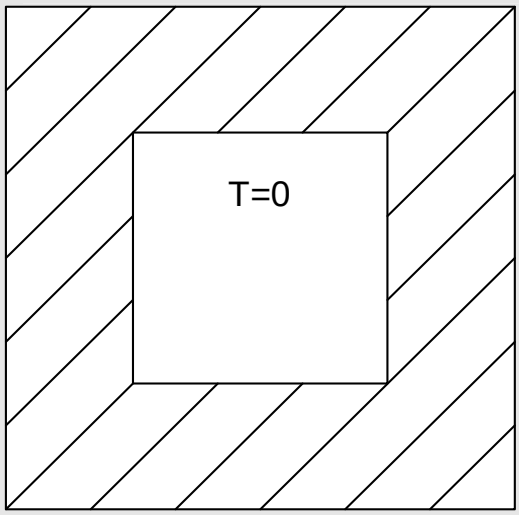
harmonische Funktion ist, die zudem für die Randpunkte mit der Wahrscheinlichkeits-Abbildung $p$ übereinstimmt – nach dem Eindeutigkeits-Prinzip (Lemma 11.2-2) folgt

$$v(x) = p(x)$$

für alle $x$!

Bevor wir nun auf einige Rechenmethoden eingehen, mit deren Hilfe die Spannungen $v(x)$ bzw. die Wahrscheinlichkeiten $p(x)$ aus den Werten für die Randpunkte berechnet werden können, wollen wir an einem physikalischen Beispiel aufzeigen, wie harmonische Funktionen außerhalb der Graphentheorie auftreten.

**Beispiel 11.2-3:**
Angenommen, man habe ein quadratförmiges dünnes Metallstück, aus dem ein kleineres Quadrat ausgeschnitten wurde. Durch eine äußere Vorrichtung wird der innere Rand auf der Temperatur 0 und der äußere Rand auf Temperatur 1 gehalten (siehe folgende Grafik).




Wie ist die Temperatur an einer beliebigen Stelle dieses Metallplättchens? Führt man ein Koordinatensystem ein und bezeichnet die Temperatur im Punkte $(x, y)$ mit $T(x, y)$, so besagt für diese Situation die sogenannte Laplacesche Differenzialgleichung:

$$T_{xx} + T_{yy} = 0$$

Die Lösungen dieser Differenzialgleichung heißen harmonisch. Sie haben die Eigenschaft, dass der Wert $T(x, y)$ stets der Mittelwert aller Werte in einem Kreis um $(x, y)$ ist, der ganz zu dem betrachteten Gebiet gehört. Das Problem, eine solche harmonische Funktion zu finden, wird auch als das Dirichletsche Problem bezeichnet.

Zum Ende dieses Abschnitts wollen wir einige Möglichkeiten skizzieren, für die eben betrachteten Gitter-Graphen die Werte $v(x)$ bzw. $p(x)$ aus den Randwerten zu berechnen. Es sind dies:

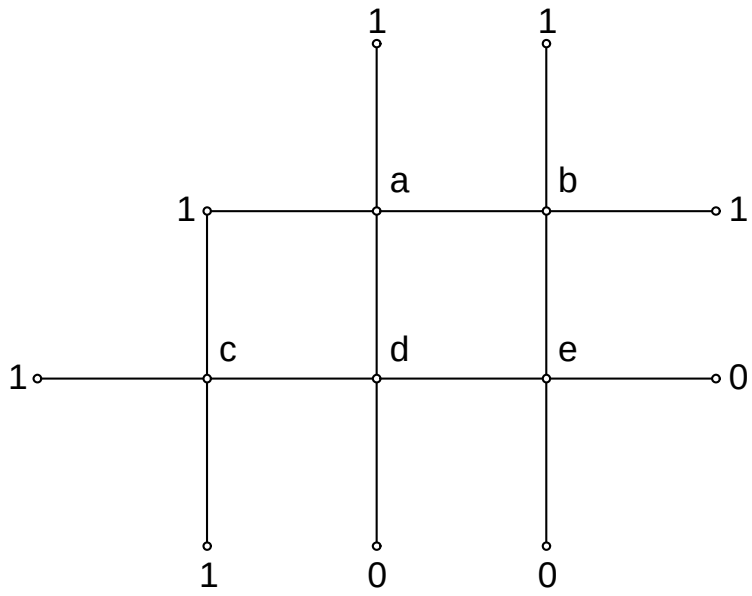
- Aufstellung eines linearen Gleichungssystems
- Auffassung als Markov-Kette
- Methode der Relaxationen
- Monte-Carlo-Methode

Wir skizzieren diese Methoden anhand des obigen Beispiels ("fliehender Taschendieb").

**Aufstellung eines linearen Gleichungssystems**

Wenn die inneren Punkte wie in der folgenden Abbildung bezeichnet sind, bekommt man für die Spannungen das folgende lineare Gleichungssystem aus fünf Gleichungen mit fünf Unbekannten:

$$v_a = \frac{v_b + v_d + 2}{4}$$
$$v_b = \frac{v_a + v_e + 2}{4}$$
$$v_c = \frac{v_d + 3}{4}$$
$$v_d = \frac{v_a + v_c + v_e}{4}$$
$$v_e = \frac{v_b + v_d}{4}$$




Die Lösung dieses Gleichungssystems lautet $v_a = 0,823$, $v_b = 0,787$, $v_c = 0,876$, $v_d = 0,506$ und $v_e = 0,323$.

**Auffassung als Markov-Kette**

Die Methode der Markov-Ketten kann als eine anspruchsvollere Version der vorigen Methode der linearen Gleichungssysteme ausgefasst werden.

Als endliche Markov-Kette bezeichnet man eine spezielle Art von Zufallsprozess, die folgendermaßen beschrieben werden kann: Man hat eine endliche Menge $S = \{s_1, s_2, ..., s_r\}$ von Zuständen und einen Zufallsprozess, aufgrund dessen stets einer dieser Zustände angenommen wird. Wenn der Prozess sich gerade im Zustand $s_i$ befindet, wird mit Wahrscheinlichkeit $P_{ij}$ danach der Zustand $s_j$ angenommen. Die Übergangswahrscheinlichkeiten $P_{ij}$ können in einer $r \times r$-Matrix $P$ angeordnet werden. (Um den Prozess eindeutig zu definieren, muss noch ein Startzustand festgelegt werden.)

Ein Random Walk auf einem endlichen Graphen kann offenbar als Markov-Kette aufgefasst werden. (Man kann auch umgekehrt zeigen, dass man sich jede Markov-Kette als Random Walk auf einem geeigneten Graphen vorstellen kann.) Der Vorteil der Auffassung als Markov-Kette liegt darin, dass in einem großen Graphen durch die spezielle Form der "Markov-Matrix" $P$ die interessierenden Wahrscheinlichkeiten (wie z. B. die "Spannungen" $v_x$) effizienter berechnet werden können als durch das lineare Gleichungssystem.

Auf Details gehen wir nicht ein.

**Methode der Relaxationen**

Bei dieser Methode fasst man die obigen linearen Gleichungen, die die Werte $v_x$ miteinander verknüpfen, als Iterationsanweisungen auf, um jeweils eine Folge $v_x^1, v_x^2, v_x^3, ...$ zu berechnen, die gegen den gesuchten Wert $v_x$ konvergiert:

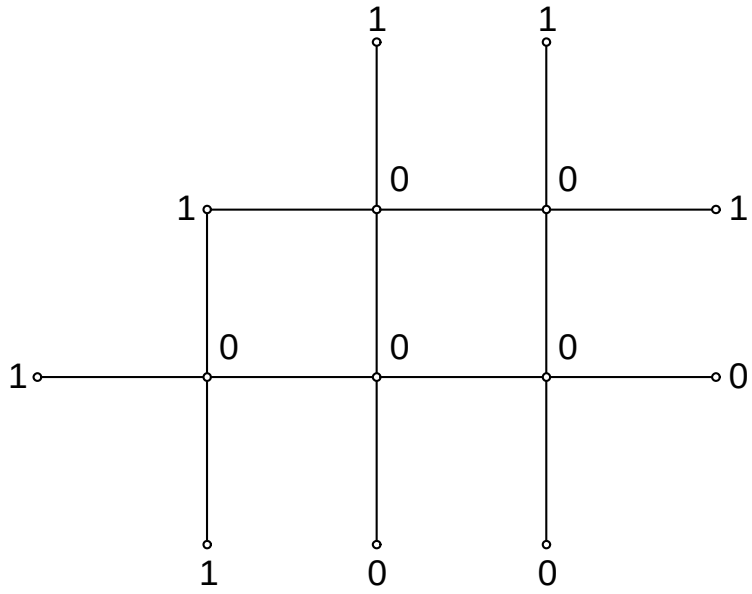
$$v_a^{n+1} = \frac{v_b^n + v_d^n + 2}{4}$$

$$v_b^{n+1} = \frac{v_a^n + v_e^n + 2}{4}$$

$$v_c^{n+1} = \frac{v_d^n + 3}{4}$$

$$v_d^{n+1} = \frac{v_a^n + v_c^n + v_e^n}{4}$$

$$v_e^{n+1} = \frac{v_b^n + v_d^n}{4}$$




Startet man mit den Werten wie in der obigen Abbildung dargestellt, so erhält man im zweiten Schritt:

$$v_a^2 = \frac{1}{2}$$

$$v_b^2 = \frac{1}{2}$$

$$v_c^2 = \frac{3}{4}$$

$$v_d^2 = 0$$

$$v_e^2 = 0$$

Diese Werte werden nun genommen, um $v_x^3$ zu berechnen etc. Man kann das Verfahren beschleunigen, indem man bei jeder Neuberechnung eines Wertes immer die **aktuellen** Werte der anderen $v_x$ einsetzt. (Bei einer Programmierung des Verfahrens kann man sich also die obere Indizierung sparen.) In der Praxis erweist sich dieses Verfahren zur Ermittlung der $v_x$-Werte als recht effizient.

**Monte-Carlo-Methode**

In der Mathematik spricht man von der Monte-Carlo-Methode, wenn ein Zufallsprozess involviert ist und man die interessierende Größe durch die wiederholte praktische Durchführung eines pseudo-zufälligen Experiments zu bestimmen versucht. In den Ingenieurwissenschaften wird in einem solchen Fall meist der Begriff der **Simulation** verwendet.

Angewendet auf die hier vorliegende Situation besteht die Monte-Carlo-Methode darin, mit Hilfe eines pseudo-zufälligen Prozesses (dies kann auch die Anwendung eines Pseudo-Zufallszahlen-Generators sein) auf einem Graphen sehr viele Random Walks nachzuspielen und auszuwerten, wie häufig man eine gewisse Ecke vor einer

anderen besucht hat usw. Dann können die interessierenden Wahrscheinlichkeiten abgeschätzt werden.

Die Monte-Carlo-Methode ist weniger effizient als die Methode der Relaxationen.

**Selbsttestaufgabe 11.2-1:**

*Skizzieren Sie die Grundidee der Methode der Relaxationen, um in einem Gitter-Graphen die Wahrscheinlichkeiten $p(x)$ für die inneren Punkte $x$ aus den Randwerten zu berechnen.*

## 11.3 Random Walks in endlichen Graphen

In diesem Abschnitt werden Random Walks und elektrische Netzwerke auf beliebigen endlichen Graphen betrachtet. Einige Ergebnisse sind bereits "klassisch" und haben den Weg in Lehrbücher gefunden, jedoch gibt es auch aktuellere Unersuchungen wie z. B. [Cha89], in denen weitere Verbindungen zwischen den Gebieten aufgedeckt werden.

Wir beginnen mit der folgenden Begriffsklärung.

**Widerstand**
**elektrisches Netzwerk**
**Leitwert**

**Definition 11.3-1:** *Es sei $G = (E, K)$ ein Graph und $r : K \to \mathbb{R}^+$ eine Abbildung, wobei $r(k)$ der **Widerstand** der Kante $k$ ist. Das Paar $(G, r)$ wird **elektrisches Netzwerk** genannt. Alternativ wird ein elektrisches Netzwerk als $(G, c)$ bezeichnet, wenn $c$ die **Leitwert**-Funktion ist (im Englischen "conductance"), also $c(k) = 1/r(k)$ für jede Kante $k$.*

Die folgenden Grundtatsachen sind aus der Theorie elektrischer Netzwerke bekannt.

- Ist jeder Ecke des Graphen ein Potenzial zugeordnet, so führt eine Potenzialdifferenz $p_k$ zwischen den Endecken $a$ und $b$ einer Kante $k$ zu einem elektrischen Strom $i_k$ von $a$ nach $b$, für den nach dem Ohmschen Gesetz gilt $i_k = \frac{p_k}{r(k)}$.
- Nach dem ersten Kirchhoffschen Gesetz ist die Potenzialdifferenz summiert über die Kanten eines Kreises stets Null. Nach dem zweiten Kirchhoffschen Gesetz hat der totale (also summierte) Strom in einer Ecke $x$ ebenfalls den Wert Null; dabei wird jeder in $x$ hinein fließende Strom (auch von außerhalb des Netzwerkes) positiv, jeder hinaus fließende negativ gerechnet. (Zur Erinnerung: In Kapitel 5 hatten wir eine solche Kantenbewertung allgemein eine Zirkulation genannt.)
- Allgemeiner lässt sich die Situation folgendermaßen formulieren: Sind $s_1, s_2, ..., s_k$ Knoten des zusammenhängenden elektrischen Netzwerks $(G, r)$, und sind $w_i (1 \leq i \leq k)$ reelle Zahlen mit $\sum w_i = 0$, so gibt es eine eindeutige Verteilung von Strömen und Potenzialdifferenzen auf den Kanten derart, dass

für jedes $i$ ein Strom der Größe $w_i$ das Netzwerk bei $s_i$ betritt (falls $w_i > 0$) bzw. verlässt (falls $w_i < 0$), und in allen Knoten das zweite Kirchhoffsche Gesetz gilt (also Strombilanzen insgesamt gleich Null).

Auch der Begriff der harmonischen Funktionen lässt sich allgemein übertragen:

**Definition 11.3-2:** *Gegeben sei ein endlicher zusammenhängender Graph $G = (E, K)$ und $S \subset E$ eine Menge von Ecken. Eine Funktion*

$$f : E \to \mathbb{R}$$

*ist **harmonisch auf** $G$ **mit Rand** $S$, falls für alle $x \in E \setminus S$ gilt*

$$f(x) = \frac{1}{d(x)} \sum_{y \in N(x)} f(y).$$

Aufgrund von Lemma 11.1-1 können wir nun (ohne Beweis) formulieren:

**Satz 11.3-1:** *Gegeben sei ein endliches zusammenhängendes Netzwerk $(G, r)$, für jede Kante $k \in K$ sei der Widerstand $r(k) = 1$. Ferner seien $s_1, s_2, ..., s_k$ Ecken in $G$. Falls in $(G, r)$ außer bei $s_1, s_2, ..., s_k$ kein Strom in das Netzwerk kommt oder austritt, ist die Funktion der absoluten Potenziale $v_x$ harmonisch auf $G$ mit Rand $S = \{s_1, s_2, ..., s_k\}$, d. h.*

$$v_x = \frac{1}{d(x)} \sum_{y \in N(x)} v_y.$$

Der hierzu korrespondierende Satz über Random Walks geht bei gegebenem $G = (E, K)$ und $S \subset E$ davon aus, dass für jedes $s \in S$ ein Gewinn $g(s)$ definiert ist. Nun wird bei $x \in E$ ein Random Walk gestartet, der endet, wenn ein Knoten $s \in S$ erreicht wird, der Gewinn ist dann $g(s)$. Mit $E_x$ werde der erwartete Gewinn bezeichnet, wenn bei $x$ gestartet wird.

**Satz 11.3-2:** *Für alle Knoten $x$ gilt*

$$E_x = \frac{1}{d(x)} \sum_{y \in N(x)} E_y.$$

Wie sich bereits in Abschnitt 11.1 abgezeichnet hat, geht die Analogie zwischen elektrischen Netzwerken und Random Walks wesentlich weiter. Für den nächsten Schritt benötigen wir die folgende Definition.

**Definition 11.3-3:** *$(G, a)$ sei ein Graph mit Kantengewichtung $a = (a_{xy})_{xy \in K}$. Man setzt*

$$A_x = \sum_{y \in N(x)} a_{xy}$$

*und*

$$P_{xy} = \left\{ \begin{array}{c} \frac{a_{xy}}{A_x} \text{ falls } xy \in K \\ 0 \text{ sonst} \end{array} \right\}$$

*Sind die Kantenbewertungen Leitwerte wie in Definition 11.3-1, so werden diese für Random Walks so interpretiert, dass $P_{xy}$ die Wahrscheinlichkeit darstellt, dass ein in $x$ angekommener RW als nächsten Knoten $y$ besucht.*

Für die jetzt folgenden Ergebnisse gehen wir davon aus, dass in den betrachteten Graphen zwei ausgezeichnete Ecken $s$ und $t$ gegeben sind.

Wie in der Literatur zu Random Walks und elektrischen Netzwerken üblich, werden wir die Terminologien aus beiden Gebieten simultan verwenden.

**Fluchtwahrscheinlichkeit**

**Definition 11.3-4:** *Gegeben sei ein elektrisches Netzwerk $(G, r)$, $s$ und $t$ seien Ecken in $G$. Die **Fluchtwahrscheinlichkeit** $P_{esc}(s \to t)$ ist die Wahrscheinlichkeit, dass ein Random Walk, der bei $s$ startet, durch die Ecke $t$ kommt, bevor er wieder zu $s$ zurückkehrt.*

Die nächsten interessanten Begriffe sind der des effektiven Widerstands und des effektiven Leitwerts.

**Definition 11.3-5:** *Gegeben sei ein einfacher zusammenhängender Graph $G$, $s$ und $t$ seien Ecken. Es wird von außen eine Spannung mit $v_s = 1$ und $v_t = 0$ angelegt. Dann fließt ein Strom $i_s$ in $s$ hinein.*

$$R_{eff}(s,t) = \frac{1}{i_s}$$

**effektive Widerstand**

*heißt der **effektive Widerstand** zwischen $s$ und $t$. Legt man statt der Potentialdifferenz zwischen $s$ und $t$ den in $s$ hineinfließenden Strom $i_s$ auf den Wert 1 fest, so wird die Potentialdifferenz betragsmäßig gleich $R_{eff}(s,t)$ sein. Der Kehrwert*

$$C_{eff}(s,t) = \frac{1}{R_{eff}(s,t)}$$

**effektiver Leitwert**

*heißt **effektiver Leitwert**.*

**Satz 11.3-3:** *Sei $(G, c)$ ein zusammenhängendes elektrisches Netzwerk und $s \neq t$ Ecken in $G$. Für jede Ecke $x$ sei*

$$v_x = \mathbb{P}\,(\text{nach Start bei } x \text{ kommt ein RW zuerst zu } s \text{ und erst dann zu } t),$$

*insbesondere also $v_s = 1$ und $v_t = 0$.*

*Dann ist $(v_x)_{x\in E}$ die Verteilung der absoluten Potentiale, wenn $s$ auf 1 und $t$ auf 0 gesetzt wird.*

*Der effektive Leitwert ist*

$$C_{eff}(s,t) = C_s P_{esc}(s \to t).$$

*Ferner gilt*

$$\frac{P_{esc}(s \to t)}{P_{esc}(t \to s)} = \frac{C_t}{C_s}.$$

**Beweis:** Startet der RW bei $x \neq s,t$, so hat man

$$v_x = \sum_y P_{xy} v_y = \sum_y \frac{c_{xy}}{C_x} v_y$$

bzw.

$$C_x v_x = \sum_{y\in N(x)} c_{xy} v_y.$$

Dies besagt jedoch gerade, dass $(v_x)_{x\in E}$ die Verteilung der Potenziale mit $v_s = 1$ und $v_t = 0$ ist.

Weiter stellen wir nun fest, dass

$$P_{esc}(s \to t) = 1 - \sum_{y\in N(s)} P_{sy} v_y$$

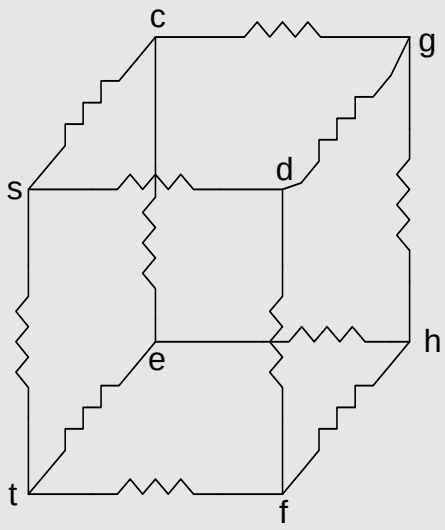
ist, denn im ersten Schritt gelangt man von $s$ mit Wahrscheinlichkeit $P_{sy}$ zum Nachbarn $y$, und von dort mit Wahrscheinlichkeit $v_y$ erst zu $s$ und dann zu $t$. Daher gilt für den gesamten Strom

$$\begin{aligned} C_{eff}(s,t) &= \sum_{y\in N(s)} (v_s - v_y) c_{sy} \\ &= \sum_{y\in N(s)} (v_s - v_y) \frac{c_{sy} C_s}{C_s} \\ &= C_s \sum_{y\in N(s)} (\frac{c_{sy}}{C_s} - v_y \frac{c_{sy}}{C_s}) \\ &= C_s (1 - \sum_{y\in N(s)} P_{sy} v_y) \\ &= C_s P_{esc}(s \to t). \end{aligned}$$

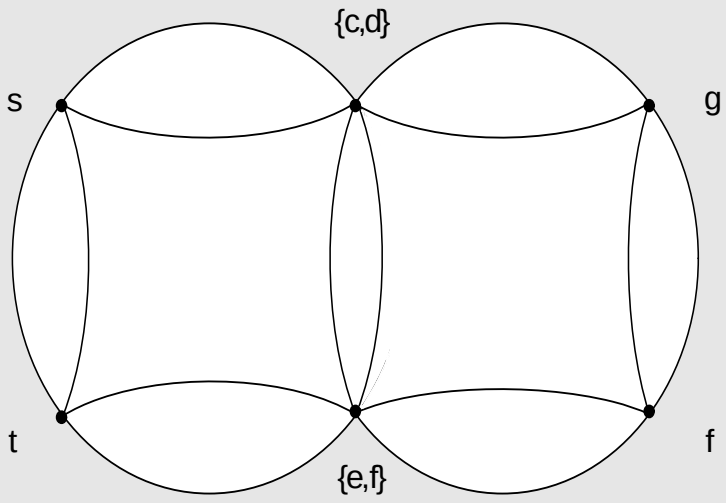
Da außerdem gilt $C_{eff}(s,t) = C_{eff}(t,s)$, folgt schließlich auch

$$\frac{P_{esc}(s \to t)}{P_{esc}(t \to s)} = \frac{C_{eff}(s,t)/C_s}{C_{eff}(t,s)/C_t} = \frac{C_t}{C_s}.$$

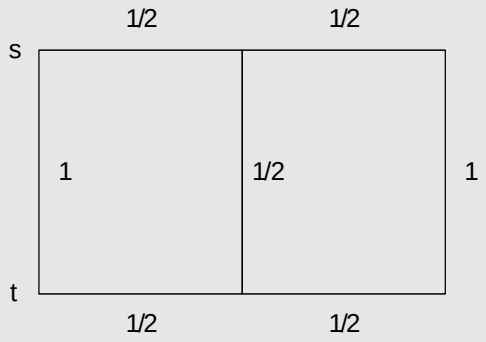
**Beispiel 11.3-1:**
Wir wollen den effektiven Widerstand zwischen zwei benachbarten Ecken des Einheitswürfels aus Einheitswiderständen bestimmen (siehe folgende Grafik).

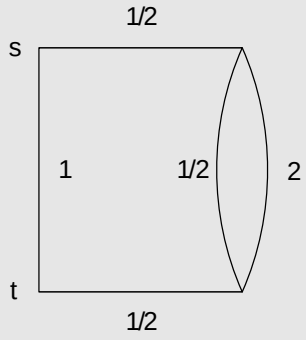


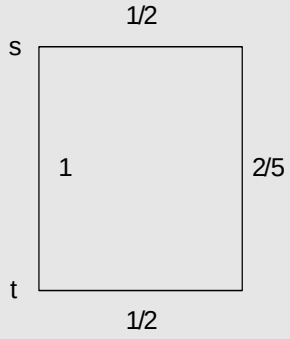

Wir legen eine 1-Volt-Spannung zwischen $s$ und $t$. Aus Symmetriegründen folgt zunächst, dass die Spannungen in $c$ und $d$ gleich sind, ebenso in $e$ und $f$. Folglich ist unser Schaltkreis äquivalent zu dem in der folgenden Darstellung.

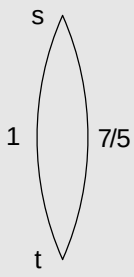


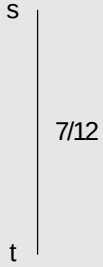

Durch die Anwendung der üblichen Gesetze für die Serien- und Parallelschaltung von Widerständen gelangt man zu der nachfolgend dargestellten Abfolge von Vereinfachungen, an deren Ende das Resultat steht: Der effektive Widerstand zwischen $s$ und $t$ beträgt $\frac{7}{12}$ Ohm.


1/2
1/2
s
1
1/2
1
t
1/2
1/2

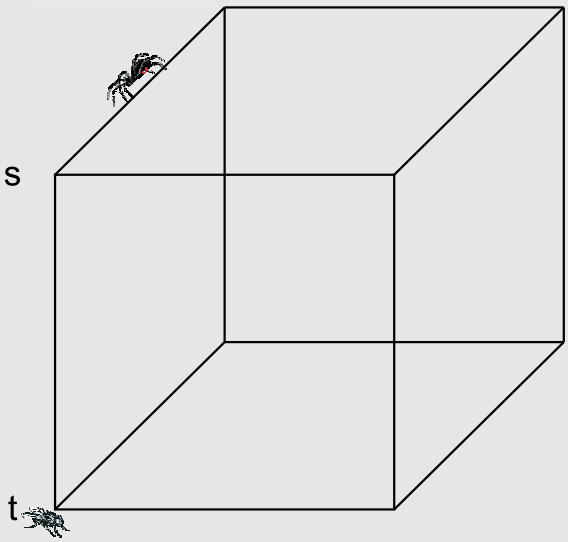

1/2
s
1
1/2
2
t
1/2


1/2
s
1
2/5
t
1/2


s
1
7/5
t


s
7/12
t

**Beispiel 11.3-2:**
Eine Spinne krabbelt zufällig auf dem Einheitswürfel herum (siehe folgende Grafik).




Falls sie von $s$ aus startet - wie groß ist die Wahrscheinlichkeit, dass sie die tote Fliege bei $t$ erreicht, bevor sie zu $s$ zurückkehrt?

Aufgrund des vorigen Beispiels wissen wir:

$$C_{eff}(s,t) = \frac{12}{7}$$

Mit Satz 11.3-3 ergibt sich damit

$$P_{esc}(s \to t) = C_{eff}(s,t)/C_s = \frac{12}{7}/3 = \frac{4}{7}.$$

Als nächstes fragen wir nach der erwarteten Anzahl von Schritten, in einem Random Walk von einer bestimmten Ecke zu einer anderen zu gelangen.

Ein beliebiger Graph $G = (E, K)$ sei gegeben.

Mit $H_{uv}$ bezeichnen wir die erwartete Anzahl von Schritten eines Random Walk in $G$, der bei der Ecke $u$ startet und bei der Ecke $v$ stoppt.

$C_{uv}$ sei $H_{uv} + H_{vu}$.

$R_{uv}$ sei der effektive Widerstand zwischen $u$ und $v$ in dem elektrischen Netzwerk $(G, r)$ mit $m$ vilen Kanten, in dem jede Kante von $G$ einen Widerstand von 1 $\Omega$ darstellt.

**Satz 11.3-4:** *Es gilt stets* $C_{uv} = 2mR_{uv}$.

**Beweis:** Mit $\gamma(x)$ wird wieder der Eckengrad einer Ecke bezeichnet, also $\gamma(x) = |N(x)|$. $\Phi_{uv}$ sei die Spannung bei $u$ bezüglich $v$, falls ein Strom der Stärke $\gamma(x)$ in jeden Knoten $x \in E$ hineinfließt und den Knoten $v$ ein Strom der Stärke $2m$ verlässt.

Zuerst wird gezeigt, dass für alle Ecken $u$ gilt

$$H_{uv} = \Phi_{uv}.$$

Aufgrund der Kirchhoffschen Gesetze und des Ohmschen Gesetzes folgt nämlich

$$\gamma(u) = \sum_{w \in N(u)} (\Phi_{uv} - \Phi_{wv})$$

für alle $u \neq v$.

Dies lässt sich umformen zu

$$\Phi_{uv} = \sum_{w \in N(u)} \frac{1}{d(u)}(1 + \Phi_{wv})(*).$$

Andererseits kann mit elementarer Wahrscheinlichkeitstheorie geschlossen werden

$$H_{uv} = \sum_{w \in N(u)} \frac{1}{\gamma(u)}(1 + H_{wv})(**)$$

für alle $u \neq v$.

Fast man die $\Phi_{uv}$ bzw. $H_{uv}$ als Unbekannte auf, so hat man mit $(*)$ und $(**)$ zwei identische lineare Gleichungssysteme vorliegen, die eine eindeutige Lösung besitzen - dies zeigt, dass $H_{uv} = \Phi_{uv}$ gelten muss.

Wir stellen nun fest, dass also $H_{vu}$ gleich der Spannung $\Phi_{vu}$ bei $v$ in $(G, r)$ bezüglich $u$ ist, wenn Ströme in alle anderen Knoten hineingeführt und bei $u$ hinausgelassen werden. Werden die Vorzeichen geändert, so lässt sich damit auch sagen: $\Phi_{vu}$ ist die Spannung bei $u$ beüglich $v$, wenn ein Strom der Stärke $2m$ in $u$ eingeführt wird und bei jedem $v \neq u$ in der Größe $\gamma(v)$ wieder abfließt. Da elektrische Netzwerke linear überlagert werden können, kann nun $C_{uv} = H_{uv} + H_{vu}$ bestimmt werden: Da sich bei allen Knoten außer $u$ und $v$ die eingeführten bzw. abgezweigten Ströme aufheben, wird so $C_{uv}$ die Spannung zwischen $u$ und $v$, wenn ein Strom der Stärke $2m$ bei $u$ eingeführt und bei $v$ abgezweigt wird. Die Aussage des Satzes folgt nun mit dem Ohmschen Gesetz.

**Rayleigh's Monotoniegesetz**

Zum Ende dieses Abschnitts kommen wir noch auf **Rayleigh's Monotoniegesetz** zu sprechen kommen. Es besagt im wesentlichen, dass sich durch die Vergrößerung von Einzelwiderständen der effektive Widerstand auch nur vergrößern kann:

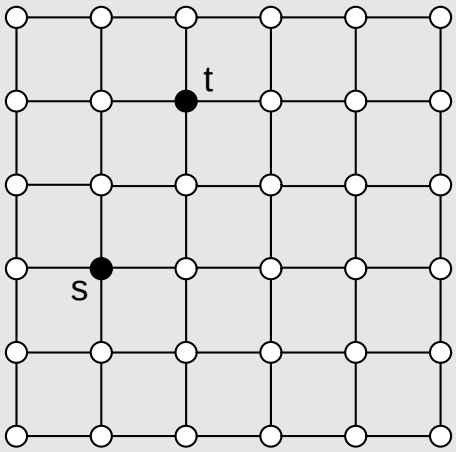
**Satz 11.3-5:** *$(G, r)$ und $(G, r')$ seien elektrische Netzwerke auf dem Graphen $G$, und es sei*

$$r'(k) \geq r(k)$$

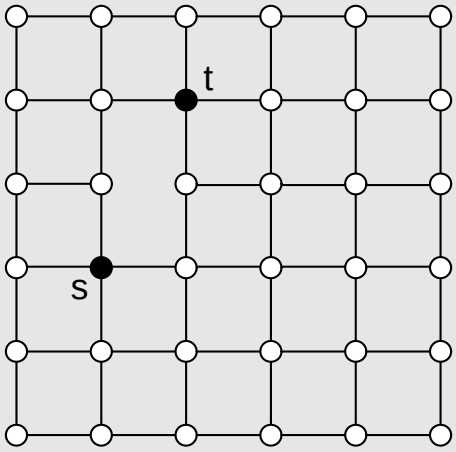
*für jede Kante $k$ in $G$. Dann gilt für beliebige Ecken $s$ und $t$ in $G$ für die effektiven Widerstände*

$$R'_{eff}(s,t) \geq R_{eff}(s,t).$$

**Beispiel 11.3-3:**
Gegeben sei der folgende Graph, mit einheitlicher Kantenbewertung 1.




In der Sprache der Random Walks ist $P_{esc}(s \to t)$ die Wahrscheinlichkeit, dass ein bei $s$ startender RW $t$ erreicht, bevor er wieder zu $s$ zurückkehrt. Entfernt man nun aus dem Graphen eine Kante, so kann dies interpretiert werden als Erhöhung ihres Widerstandes auf "unendlich" (siehe folgende Darstellung).



Nach Rayleigh's Monotoniegesetz gilt nun

$$P'_{esc}(s \to t) \leq P_{esc}(s \to t).$$

Die Interpretation lautet: **Es ist schwieriger geworden, von $s$ nach $t$ zu entkommen**.

In der Sprache der elektrischen Netzwerke stellt sich die Ungleichung mit Satz 11.3-3 dar als:

$$R'_{eff}(s,t) \geq R_{eff}(s,t).$$

Das Rayleighsche Monotoniegesetz beinhaltet auch die entsprechende Aussage für die Verkleinerung von Einzelwiderständen, als deren Resultat sich die effektiven Widerstände ebenfalls verkleinern.

Obwohl die Ausssagen des Rayleighschen Monotoniegesetzes recht plausibel sind, ist das Gesetz nicht einfach zu beweisen. In der Literatur sind unterschiedliche Beweise zu finden (siehe z. B. [Doy84] oder [Bol98]), wobei man sowohl in der Sprache der Random Walks als auch in der elektrischer Netzwerke argumentieren kann.

Auf die Darstellung eines Beweises wollen wir an dieser Stelle verzichten.

Die Bedeutung des Rayleighschen Gesetzes liegt vor allem darin, dass man mit seiner Hilfe effektive Widerstände bzw. Wahrscheinlichkeiten in großen komplexen Netzwerken bzw. Graphen abschätzen kann, indem man durch Hinzufügen oder Entfernen von Kanten zu bekannteren Strukturen kommt, deren Parameter man bereits kennt. Man spricht von **Rayleigh's short-cut Methode**.

Die Methode kann insbesondere für den Beweis des Polyaschen Satzes über Random Walks in unendlichen Gittern angewendet werden (siehe mehr dazu im nächsten Abschnitt).

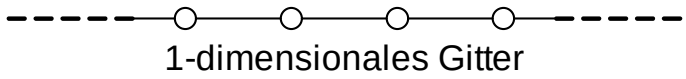
**Selbsttestaufgabe 11.3-1:**

*Erläutern Sie den Begriff des effektiven Widerstands zwischen zwei beliebig gewählten Ecken eines endlichen Graphen.*

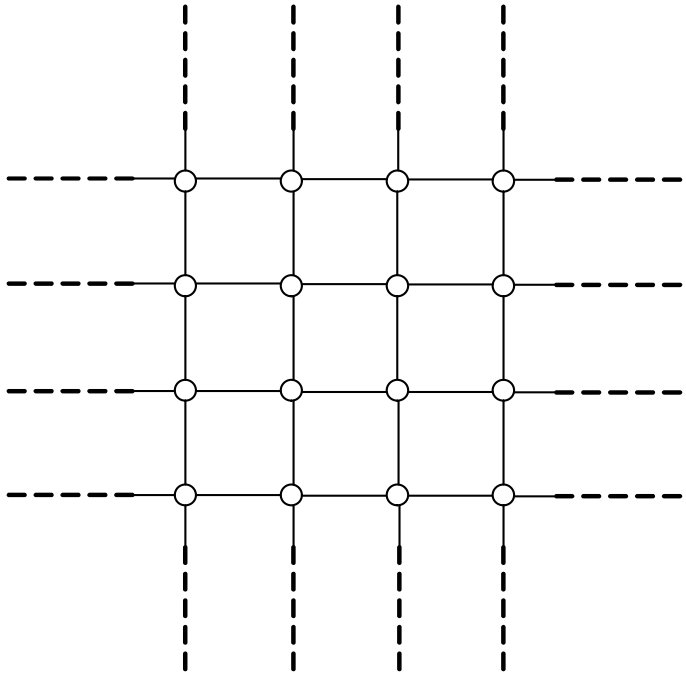
## 11.4 Random Walks in unendlichen Graphen

Wir wollen uns in diesem Abschnitt beschränken auf das "klassische" Problem der Random Walks in unendlichen Gittern, welches zuerst von George Polya im Jahre 1921 untersucht wurde (siehe [Pol21]). Erweiterungen der Ergebnisse auf allgemeine unendliche Graphen, die vorwiegend auf eine geschickte Anwendung der

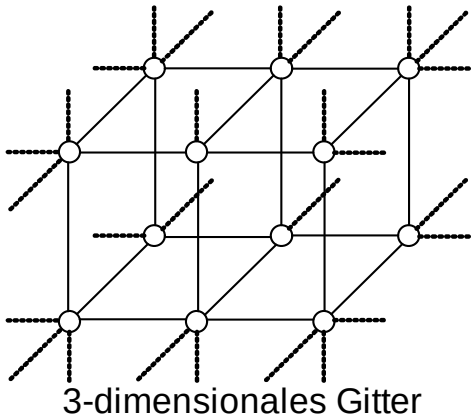
Rayleighschen short-cut-Methode hinauslaufen (vgl. dazu im vorigen Abschnitt), werden hier jedoch nicht betrachtet.



1-dimensionales Gitter



2-dimensionales Gitter



3-dimensionales Gitter

In der obigen Abbildung sind $d$-dimensionale unendliche Gitter für $d = 1, 2$ und $3$ dargestellt. Die Punkte kann man sich als $d$-dimensionale Vektoren $x \in \mathbb{Z}^d$ mit ganzzahligen Komponenten vorstellen. (Aus Gründen der Übersichtlichkeit ist die Vektordarstellung in den Bildern nicht eingetragen.) Kanten existieren zwischen Punkten dann, wenn sich die Vektoren in einer Komponente um den Betrag 1 unterscheiden und ansonsten gleich sind; beispielsweise gibt es also im 3-dimensionalen Gitter eine Kante zwischen $(-1, 2, 1)$ und $(-1, 3, 1)$.

Einen Random Walk im zweidimensionalen Gitter kann man sich als zufälliges Bewegen in einem unendlichen Straßennetz vorstellen. Im Falle $d = 3$ ist eine Interpretation als zufällige Bewegung eines Moleküls in einem unendlich großen räumlichen Kristall möglich.

Die zentrale Frage, die Polya in der bereits zitierten Arbeit stellt, lautet:

- Kehrt der RW mit Sicherheit (also mit Wahrscheinlichkeit 1) irgendwann zu seinem Ausgangspunkt zurück?

Falls dies der Fall ist, wird der RW als **rekurrent** bezeichnet, andernfalls als **transient**.

**rekurrent**
**transient**

**Definition 11.4-1:** *Für einen Random Walk in einem $d$-dimensionalen Gitter sei*

$$P_{escape}$$

*die Wahrscheinlichkeit, dass der RW niemals zu seinem Ausgangspunkt zurückkehrt.*

Der RW heißt

- **rekurrent**, falls $P_{escape} = 0$,
- **transient**, falls $P_{escape} > 0$.

Das Problem, festzustellen, ob ein RW (bzw. ein Graph) rekurrent oder transient ist, bezeichnet man als das **Typ-Problem**.

Der berühmte **Satz von Polya** lautet:

**Satz 11.4-1:** *Ein Random Walk auf einem $d$-dimensionalen Gitter ist rekurrent für $d = 1$ oder $2$ und transient für $d > 2$.*

In der Literatur findet man mehrere Beweise für diesen Satz. Eine Möglichkeit besteht darin zu zeigen, dass - in der Sprache der elektrischen Netzwerke - der effektive Widerstand zwischen einem beliebigen Anfangspunkt und einem "unendlich fernen" Punkt im Falle $d \leq 2$ unendlich groß ist (weshalb der RW nicht entweichen kann), im Falle $d > 2$ hingegen einen endlichen Wert hat. Auf diese Beweise können wir jedoch nicht eingehen.

Stattdessen wollen wir zum Abschluss Polyas Originalbeweis für den einfachsten Fall $d = 1$ präsentieren.

**Beweis:** für den Fall $d = 1$

Wir bezeichnen mit $u$ die Wahrscheinlichkeit, dass ein im Nullpunkt startender Random Walk dorthin zurückkkehrt. Die Wahrscheinlichkeit, dass der RW sich dort genau $k$ mal aufhält, ist dann offenbar $u^{k-1}(1-u)$. Bezeichnet man mit $m$ die erwartete Anzahl von Aufenthalten im Nullpunkt, so gilt nun

$$m = \sum_{k=1}^{\infty} k u^{k-1}(1-u) = \frac{1}{1-u}.$$

Ist $m = \infty$ bzw. $u = 1$, so ist der RW rekurrent, im Falle $m < \infty$ und $u < 1$ transient.

Es wird nun noch ein anderer Ausdruck für $m$ abgeleitet.

$u_n$ bezeichne die Wahrscheinlichkeit, dass der RW, welcher beim Punkt 0 startet, nach dem $n$-ten Schritt wieder bei 0 angekommen ist. Da bei 0 begonnen wird, ist $u_0 = 1$.

Sei nun $e_n$ eine Zufallsvariable, die den Wert 1 annimmt, falls sich der RW nach dem $n$-ten Schritt im Punkt 0 befindet, und den Wert 0 sonst. Dann ist

$$T = \sum_{n=0}^{\infty} e_n$$

die gesamte Anzahl aller Aufenthalte im Punkt 0, und

$$m = E(T) = \sum_{n=0}^{\infty} E(e_n).$$

Andererseits ist $E(e_n) = 1 \cdot u_n + 0 \cdot (1 - u_n)$, und man hat schließlich

$$m = \sum_{n=0}^{\infty} u_n.$$

Damit ist nun klar:

- Der RW ist rekurrent, falls die Reihe $\sum_{n=0}^{\infty} u_n$ divergiert, und transient, falls sie konvergiert.

Um im eindimensionalen Gitter zum Nullpunkt zurückzukehren, muss der RW sich gleich oft nach links und nach rechts bewegt haben - es sind also in diesen Fällen nur gerade Anzahlen von absolvierten Schritten möglich. Es geht folglich darum, die Wahrscheinlichkeiten $u_{2n}$ zu berechnen.

Jeder einzelne Weg, der nach $2n$ Schritten zurückkehrt, besitzt die Wahrscheinlichkeit $\frac{1}{2^{2n}}$. Die Anzahl dieser Wege entspricht der Anzahl von Möglichkeiten, aus den $2n$ Schritten $n$ viele auszuwählen, bei denen nach rechts gegangen wird. Dies bedeutet:

$$u_{2n} = \binom{2n}{n} \frac{1}{2^{2n}}$$

Das Ziel ist zu zeigen:

$$\sum_{n} u_{2n} = \infty$$

Dazu wird die bekannte Stirlingsche Formel benutzt:

$$n! \sim \sqrt{2\pi n} e^{-n} n^n$$

Damit folgt nämlich:

$$u_{2n} = \frac{(2n)!}{n!n!}\frac{1}{2^{2n}} \sim \frac{\sqrt{2\pi \cdot 2n}e^{-2n}(2n)^{2n}}{(\sqrt{2\pi n}e^{-n}n^n)^2 2^{2n}}$$

$$= \frac{1}{\sqrt{\pi n}}$$

Dies zeigt $\sum_n u_{2n} = \infty$, und der Beweis ist erbracht.

**Selbsttestaufgabe 11.4-1:**

*Erläutern Sie das Grundprinzip der Anwendung der Rayleighschen short-cut-Methode auf unendliche Graphen.*

# A Kurze Übersicht der verwendeten Begriffe und Symbole aus der Mengenlehre

**Menge**

Eine **Menge** $M$ kann durch eine Eigenschaft oder - falls sie nur endlich viele Elemente hat - durch die Aufzählung ihrer Elemente beschrieben werden: z. B. ist $M = \{1, 3, 5, 7, 9\}$ identisch mit $M = \{x | x\ ist\ eine\ ungerade\ natürliche\ Zahl, x < 10\}$.

**Element**

$x \in M$ steht für "x ist **Element** der Menge $M$".

**Teilmenge**

$M \subseteq N$ steht für "$M$ ist **Teilmenge** von $N$", d. h. für jedes $x$ mit $x \in M$ gilt auch $x \in N$.

**echte Teilmenge**

$M \subset N$ bedeutet "$M$ ist **echte Teilmenge** von $N$", also $M \subseteq N$ und $M \neq N$.

**leere Menge**

Die **leere Menge**, d. h. die Menge ohne Elemente, wird mit dem Symbol $\emptyset$ bezeichnet.
Für Mengen $M$ und $N$ wird gesetzt

**Durchschnitt**

$M \cap N = \{x | x \in M\ und\ x \in N\}$

**Vereinigung**

$M \cup N = \{x | x \in M\ oder\ x \in N\}$

$M \cap N$ heißt **Durchschnitt**, $M \cup N$ **Vereinigung** von $M$ und $N$.

**disjunkte Mengen**

**disjunkte Vereinigung**

Ist $M \cap N = \emptyset$, so heißen $M$ und $N$ **disjunkte Mengen**, $M \cup N$ die **disjunkte Vereinigung** von $M$ und $N$.
Allgemein ist für eine beliebige nicht-leere Indexmenge $I$, falls für jedes $i \in I$ eine Menge $M_i$ gegeben ist,

$$\bigcap_{i \in I} M_i = \{x | für\ alle\ i \in I\ ist\ x \in M_i\}$$

und

$$\bigcup_{i \in I} M_i = \{x | es\ gibt\ ein\ i \in I\ mit\ x \in M_i\}.$$

**Mengendifferenz**

Für Mengen $M$ und $N$ ist die **Mengendifferenz** von $M$ und $N$

$$M \backslash N = \{x | x \in M\ und\ x \notin N\}.$$

**Komplement**

**Potenzmenge**

Ist $N \subseteq M$, so heißt $M \backslash N$ auch das **Komplement** von $N$ in $M$ und wird mit $\overline{N}_M$ oder, falls die übergreifende Menge nicht benannt werden muss, einfach mit $\overline{N}$ bezeichnet. $P(M)$, die **Potenzmenge** von $M$, ist die Menge aller Teilmengen von $M$.

Ist $n$ eine natürliche Zahl, und sind $M_1, M_2, \ldots, M_n$ Mengen, dann heißt die Menge

$$X_{i=1}^{n} M_i = \{(x_1, x_2, \ldots, x_n) | für\ alle\ 1 \leq i \leq n\ ist\ x_i \in M_i\}$$

**n-faches kartesisches Produkt**

das **n-faches kartesisches Produkt** der $M_i$ für $i = 1$ bis $n$.
Für $n = 1$ wird unter $X_{i=1}^{n} M_i$ einfach die Menge $M_1$ verstanden.
Für $n = 2$ schreibt man statt $X_{i=1}^{n} M_i$ einfacher $M_1 \times M_2$.
Sind alle $M_i$ gleich einer Menge $M$, so wird $X_{i=1}^{n} M_i$ mit $M^n$ bezeichnet.

**geordnetes n-Tupel** **Tripel**

Die Elemente von $X_{i=1}^{n} M_i$ heißen ( **geordnetes n-Tupel**, im Falle $n = 3$ auch **Tripel**.

**Folge**

Ein beliebiges Element von $\bigcup_{i \in N} M^i$ wird auch als (endliche) **Folge** von Elementen aus $M$ bezeichnet.

**Relation**

Sind $M$ und $N$ Mengen, so wird eine Teilmenge $R \subseteq M \times N$ auch eine **Relation** zwischen $M$ und $N$ genannt.

**eindeutige Relation** **Funktion**

R ist eine **eindeutige Relation** oder **Funktion**, wenn für $m \in M$ und $n, n' \in N$ aus $(m, n) \in R$ und $(m, n') \in R$ stets $n = n'$ folgt.

**Abbildung**

Gilt für eine eindeutige Relation $R \subseteq M \times N$, dass zu jedem $m \in M$ ein (wegen Eindeutigkeit dann genau ein) $n \in N$ mit $(m, n) \in R$ existiert, so heißt R **Abbildung** von $M$ in $N$ (oder: von $M$ nach $N$).
Für Abbildungen werden meist kleine lateinische oder aber griechische Buchstaben als Namen verwendet, z. B. f (statt R). Auch schreibt man statt $f \subseteq M \times N$ üblicherweise $f : M \to N$ oder $M \xrightarrow{f} N$ , um herauszuheben, dass durch f jedem $m \in M$ (genau) ein $n \in N$ zugeordnet wird, dieses Element von $N$ wird auch mit $f(n)$ bezeichnet. Ist $f : M \to N$ eine Abbildung und $\tilde{M} \subseteq M$ eine Teilmenge von $M$, so wird die **Einschränkung** von $f$ auf $\tilde{M}$ mit $f/\tilde{M}$ bezeichnet, d. h. man hat die Abbildung $f/\tilde{M} : \tilde{M} \to N$ .

**Einschränkung einer Abbildung**

**injektiv** **surjektiv** **bijektiv**

Eine Abbildung $f : M \to N$ heißt **injektiv**, wenn aus $m_1 \neq m_2$ in $M$ stets $f(m_1) \neq f(m_2)$ in $N$ folgt. f ist **surjektiv**, falls es zu jedem $n \in N$ (mindestens) ein $m \in M$ gibt mit $f(m) = n$. Schließlich heißt f **bijektiv**, wenn $f$ injektiv und surjektiv ist.

Für eine Relation $R \subseteq M \times M$ auf einer Menge $M$ schreibt man statt $(x, y) \in R$ auch $xRy$.

Man definiert:

**reflexiv**

- ist **reflexiv** auf $M$, wenn für alle $x \in M$ gilt $xRx$.

**symmetrisch**

- $R$ ist **symmetrisch** auf $M$, wenn aus $xRy$ stets $yRx$ folgt.

**transitiv**

- ist **transitiv** auf $M$, wenn aus $xRy$ und $yRz$ stets $xRz$ folgt.

**Äquivalenzrelation**

- $R$ ist **Äquivalenzrelation** auf $M$, wenn $R$ reflexiv, symmetrisch und transitiv auf $M$ ist.

**Äquivalenzklassen**

Eine Äquivalenzrelation gibt Anlass zu einer disjunkten Zerlegung der Menge $M$ in die $R-$**Äquivalenzklassen**. Innerhalb einer solchen Klasse stehen je zwei Elemente in der Relation $R$, und umgekehrt gehören zwei in der Relation stehende Elemente von $M$ stets zur selben Klasse.

# B Kurze Erläuterung der verwendeten Begriffe aus der Linearen Algebra

**Körper**

Eine Menge $F$, in der man in der üblichen Weise addieren, subtrahieren, multiplizieren und dividieren kann, wird **Körper** genannt. Dabei geht man bei der axiomatischen Definition eines Körpers zunächst nur von den Operationen $+$ und $\cdot$ aus sowie zwei besonderen Elementen $0$ und $1$. (Ein Körper hat also stets mindestens zwei Elemente.)

Die Axiome sind die folgenden:

**Axiom: A1. Assoziativgesetz:**
$a + (b + c) = (a + b) + c$ *für alle* $a, b, c \in F$

**Axiom: A2. Kommutativgesetz:**

$a + b = b + a$ für alle $a, b \in F$;

**Axiom: A3.** $0$ **als neutrales Element:**

$a + 0 = a$ für alle $a \in F$;

**Axiom: A4. Existenz inverser Elemente:**

Zu jedem $a \in F$ existiert ein
$b \in F$ mit $a + b = 0$.

Für die Multiplikation gilt:

**Axiom: M1. Assoziativgesetz:**

$a \cdot (b \cdot c) = (a \cdot b) \cdot c$ für alle $a, b, c \in F$;

**Axiom: M2. Kommutativgesetz:**

$a \cdot b = b \cdot a$
für alle $a, b \in F$;

**Axiom: M3.** $1$ **als neutrales Element:**

$a \cdot 1 = a$ für alle $a \in F$;

**Axiom: M4. Existenz inverser Elemente:**

Zu jedem $a \in F$, $a \neq 0$, existiert ein $b \in F$ mit $a \cdot b = 1$.

Für die Addition und Multiplikation gilt das

**Axiom: D. Distributivgesetz:**

$(a + b) \cdot c = (a \cdot c) + (b \cdot c)$
für alle $a, b, c \in F$.

**Körper** $\mathbb{R}$ In Anwendungen hat man es meist mit dem **Körper** $\mathbb{R}$ der reellen Zahlen zu tun. Für diesen Kurs ist auch der **Körper** $\mathbb{F}_2$ von besonderer Wichtigkeit: es handelt sich um die Menge $\mathbb{F}_2 = \{0, 1\}$ mit der in den folgenden Tafeln beschriebenen Addition und Multiplikation. Man beachte die Besonderheit $1 + 1 = 0$.

| + | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| · | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Es sei $F$ ein beliebiger Körper. Ein **Vektorraum** über $F$ besteht aus einer Menge $V$ (den Vektoren) mit einem besonderen Element $0$ und folgenden Eigenschaften: In $V$ ist eine Addition $+$ für Vektoren erklärt, so dass dafür $A1$ bis $A4$ (mit $0$ als neutralem Element) gelten. Ferner ist eine sogenannte **skalare Multiplikation** von Vektoren mit Körperelementen definiert, für die gilt:

**skalare Multiplikation**

**Axiom: V1. Distributivgesetze:**
$(\alpha + \beta) \cdot a = (\alpha \cdot a) + (\beta \cdot a)$ *und* $\alpha \cdot (a + b) = (\alpha \cdot a) + (\alpha \cdot b)$ *für alle* $a, b \in V$ *und* $\alpha, \beta \in F$*;*

**Axiom: V2. Assoziativgesetz:**
$\alpha \cdot (\beta \cdot a) = (\alpha \cdot \beta) \cdot a$ *für alle* $a \in V$ *und* $\alpha, \beta \in F$*;*

**Axiom: V3. Multiplikation mit** $1 \in F$**:**
$1 \cdot a = a$ *für alle* $a \in V$.

In dem soeben beschriebenen Sinne wird die Potenzmenge $\mathbf{P}(M)$ einer beliebigen Menge $M$ zu einem Vektorraum über $\mathbb{F}_2$ (mit $\emptyset$ als Nullelement), wenn man $+$ als die **symmetrische Differenz** definiert, d. h. für $A, B \in \mathbf{P}(M)$ setzt

**symmetrische Differenz**

$$A + B := (A \cup B) \backslash (A \cap B).$$

Es gilt dann immer $A + A = \emptyset$. Für die skalare Multiplikation mit den Elementen $\mathbb{F}_2$ setzt man $0 \cdot A = \emptyset$ und $1 \cdot A = A$.

**Untervektorraum** Ist $V$ ein Vektorraum über einem Körper $F$, so ist $U \subseteq V$ ein **Untervektorraum** von $V$, wenn $U$ unter den algebraischen Operationen abgeschlossen ist, d. h. wenn aus $u, v \in U$ und $\lambda \in F$ stets $u + v \in U$ und $\lambda \cdot u \in U$ folgt.

In einer Potenzmenge $\mathbf{P}(M)$ bilden z. B. alle Teilmengen von $M$ mit gerader Elementeanzahl einen Untervektorraum, denn haben $A, B \in \mathbf{P}(M)$ eine gerade Anzahl von Elementen, so gilt dies auch für $A + B$.

Gilt in einem Vektorraum $V$ über dem Körper $F$ die Beziehung
$v = \alpha_1 \cdot v_1 + \ldots + \alpha_n v_n$ mit $v, v_1, \ldots, v_n \in V$ und $\alpha_1, \ldots, \alpha_n \in F$, so heißt $v$ eine **Linearkombination** von $v_1, \ldots, v_n$.

**Linearkombination**

Elemente $v_1, \ldots, v_n$ eines Vektorraums $V$ bilden eine **Basis** von $V$, wenn $v_1, \ldots, v_n$**linear unabhängig** sind (d. h. kein $v_i$ lässt sich als Linearkombination anderer $v_j$ erhalten) und wenn jedes Element $v \in V$ sich als Linearkombination gewisser $v_j$ darstellen lässt.

**Basis**

**linear unabhängig**

Ist $V$ Vektorraum über $\mathsf{IF}_2$ und hat eine Basis von $n$ Elementen, so gibt es insgesamt $2^n$ viele Elemente (Vektoren) in $V$.

Für eine $n$-elementige Menge $M = \{m_1, \ldots, m_n\}$ bilden z. B. die einelementigen Teilmengen $v_1 = \{m_1\}, \ldots, v_n = \{m_n\}$ eine Basis des Vektorraums $P(M)$.

Ein Vektorraum kann viele Basen haben, jedoch ist die Anzahl der Vektoren in einer Basis eindeutig und heißt **Dimension** des Vektorraums.

**Dimension**

In einem Vektorraum $V$ (über $F$) kann auch eine Multiplikation zwischen Vektoren mit einem Körperelement als Ergebnis definiert werden, man spricht dann von einem **Skalarprodukt**. In dem Vektorraum $F^n$ der $n$-tupel über dem Körper $F$ setzt man z. B. üblicherweise für Vektoren $u, v \in F^n$

**Skalarprodukt**

$$(u_1, \ldots, u_n) \cdot (v_1, \ldots, v_n) := u_1 \cdot v_1 + \ldots + u_n \cdot v_n,$$

wobei die rechtsstehenden Multiplikationen und Additionen in $F$ auszuführen sind. Solche Skalarprodukte haben Eigenschaften, wie man sie vom dreidimensionalen reellen Raum kennt (z. B. gilt $u \cdot v = v \cdot u$ und $u \cdot (v + w) = u \cdot v + u \cdot w$).

Auf die präzise (axiomatische) Definition eines Skalarprodukts soll hier verzichtet werden.[1]

Setzt man in dem Vektorraum $P(M)$ (über $\mathsf{IF}_2$) für $A, B \in P(M)$

$$A \cdot B := \begin{cases} 0, & \text{falls } |A \cap B| \text{ gerade ist,} \\ 1, & \text{falls } |A \cap B| \text{ ungerade ist,} \end{cases}$$

so hat die so definierte Multiplikation von Vektoren (mit Werten in $\mathsf{IF}_2$) ebenfalls die Eigenschaften eines Skalarprodukts.

Die Analogie wird aus folgendem klarer: hat man in $P(M)$ die Basis $v_i = \{m_i\}$ (s. o.) gewählt, und haben $A$ und $B$ die Darstellungen $A = a_1 \cdot v_1 + \ldots + a_n \cdot v_n$, $B = b_1 \cdot v_1 + \ldots + b_n \cdot v_n (a_i, b_i \in \mathsf{IF}_2)$, so ist die obige Definition der Skalarmultiplikation gleichbedeutend mit

$$A \cdot B = a_1 \cdot b_1 + \ldots + a_n \cdot b_n,$$

wobei die rechtsstehenden Multiplikationen und Additionen in $\mathsf{IF}_2$ auszuführen sind.

1 Es sollte beachtet werden, dass für verschiedene Additionen bzw. Multiplikationen stets die gleichen Zeichen $+$ und $\cdot$ verwendet werden; z. B. hat man bei dem Vektorraum $P(M)$ über $\mathsf{IF}_2$ die Multiplikation in $\mathsf{IF}_2$, die Multiplikation von Vektoren mit Körperelementen und die Skalarmultiplikation zweier Vektoren.

**orthogonal**

Vektoren $u, v$ in einem Vektorraum $V$, in dem ein Skalarprodukt definiert ist, heißen **orthogonal** oder **zueinander senkrecht** (in Zeichen $u \perp v$), wenn $u \cdot v = 0$ gilt. Untervektorräume $U_1, U_2 \subseteq V$ sind orthogonal ($U_1 \perp U_2$), wenn für beliebige $u_1 \in U_1$ und $u_2 \in U_2$ stets $u_1 \perp u_2$ ist.

$V$ und $W$ seien Verktorräume über dem Körper $F$. Eine Abbildung

$$f : V \to W$$

**lineare Abbildung**

heißt **linear**, wenn für alle $u, v \in V$ und $\alpha \in F$ gilt

$$f(u + v) = f(u) + f(v)$$

$$f(\alpha \cdot u) = \alpha \cdot f(u).$$

(Dabei sind die Operationen in $V$ und $W$ mit den gleichen Symbolen bezeichnet.)

**Kern**
**Bild**

Ferner setzt man die Untervektorräume Kern $f$ von $V$ und Bild $f$ von $W$ folgendermaßen fest:

$$\text{Kern} f := \{a \in V | f(a) = 0\},$$

$$\text{Bild} f := \{b \in W | \text{ es gibt ein } a \in V \text{ mit } f(a) = b\}.$$

# C Kurze Erläuterung der verwendeten Begriffe aus der Theorie der Matrizen

**Matrix über einem Körper**

Eine $(n \times m)$-**Matrix über einem Körper** $F$ besteht aus $n \cdot m$ vielen Elementen des Körpers $F$, die in einem rechteckigen Schema mit $n$ Zeilen und $m$ Spalten angeordnet sind. Um die einzelnen Einträge in der Matrix $A$ benennen zu können, ist die Schreibweise

$$A = (a_{ij})$$

**Indizierung**

üblich, wobei in der doppelten **Indizierung** $i$ die Zeilen und $j$ die Spalten durchläuft. $a_{ij}$ bezeichnet also den Eintrag der Matrix in Zeile $i$ und Spalte $j$, im Bild:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix}$$

Für Matrizen sind gewisse Rechenoperationen erklärt, die ihrerseits gewissen Rechenregeln genügen. Die Eigenschaften von Matrizen und des Rechnens mit ihnen werden in der Linearen Algebra nutzbringend angewandt, z. B. beim Lösen linearer Gleichungssysteme.

Die **Multiplikation von Matrizen** ist folgendermaßen definiert. Gegeben seien eine $(n \times m)$-Matrix $A$ und eine $(m \times p)$-Matrix $B$ (über $F$). Das **Produkt** $C = A \cdot B$ der Matrizen $A$ und $B$ ist die $(n \times p)$-Matrix $(c_{ij})$ mit

$$c_{ij} = \sum_{k=1}^{m} a_{ik} \cdot b_{kj} \qquad (1 \leq i \leq n, 1 \leq j \leq p).$$

Anschaulich gesprochen werden die $j$-te Spalte von $B$ und die $i$-te Zeile von $A$ "übereinandergelegt" (beide bestehen aus $m$ Elementen von $F$), die in Paaren gruppierten Zahlen jeweils multipliziert und die Ergebnisse aufsummiert, wobei diese Rechnung im Körper $F$ ausgeführt wird.

Sind drei Matrizen $A, B, C$ gegeben und (aufgrund der Zeilen- und Spaltenanzahlen) $A \cdot B$ und $B \cdot C$ ausführbar, so sind auch $(A \cdot B) \cdot C$ und $A \cdot (B \cdot C)$ erklärt, und es gilt das **Assoziativgesetz**

**Assoziativgesetz**

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C.$$

Eine **quadratische Matrix** hat die gleiche Zeilen- wie Spaltenanzahl. Eine quadratische Matrix $A$ kann mit sich selbst multipliziert werden, man erhält $A \cdot A$, $(A \cdot A) \cdot A$ u.s.w.. Da wegen des Assoziativgesetzes die Klammerung (also die Reihenfolge der Einzelmultiplikationen) keine Rolle spielt, macht es Sinn, einfach von $A^2, A^3$, allgemein von der $n$-ten **Potenz** $A^n$ von $A$ zu sprechen.

Die Zeilen bzw. Spalten einer $(n \times m)$-Matrix $A$ können als Vektoren des Vektorraums $F^m$ bzw. $F^n$ aufgefasst werden. Die maximale Anzahl linear unabhängiger Zeilen von $A$ heißt **Zeilenrang** von $A$, die maximale Anzahl linear unabhängiger Spalten **Spaltenrang** von $A$. Trivialerweise kann der Zeilenrang höchstens $n$ und der Spaltenrang höchstens $m$ sein. Ein wichtiger Satz besagt nun, dass der Zeilenrang einer Matrix stets gleich ihrem Spaltenrang ist. Diese Zahl wird einfach **Rang** genannt. Man schreibt $rg(A)$ für den Rang der Matrix $A$ oder $rg_F(A)$, wenn der Bezug zum Körper $F$ herausgestellt werden soll.

Zu beachten ist, dass die Bestimmung des Rangs einer Matrix zu verschiedenen Ergebnissen führen kann, wenn man verschiedene Körper zugrundelegt. Beispielsweise kann die folgende Matrix $A$, die als Einträge nur $0$ oder $1$ hat, sowohl als Matrix über dem zweielementigen Körper $\mathsf{IF}_2$ als auch als Matrix über dem Körper $\mathsf{IR}$ der reellen Zahlen aufgefasst werden:

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

Es gilt $rg_{\mathsf{IR}}(A) = 3$, aber $rg_2(A) = 2$. (Der Rang über $\mathsf{IF}_2$ wird der Einfachheit halber mit $rg_2$ bezeichnet.)

Die **Transponierte** einer Matrix $A$ wird mit $A^t$ bezeichnet. Sie wird durch Vertauschung der Zeilen mit den Spalten aus $A$ erhalten. $A^t$ ist also eine $(m \times n)$-Matrix, wenn $A$ eine $(n \times m)$-Matrix ist. Aufgrund des oben Gesagten ist klar, dass

$$rg_F(A) = rg_F(A^t)$$

gilt.

Die **Determinante** einer Matrix $A$, $det(A)$, ist ein Element von $F$, welches einer beliebigen quadratischen Matrix zugeordnet werden kann. Genauer kann man die Determinante für beliebige $(n \times n)$-Matrizen z. B. auf folgende Weise induktiv definieren:

- ist $n = 1$, so ist $det(A) = a_{11}$;
- ist $n > 1$, so ist $det(A) = \sum_{j=1}^{n} (-1)^{j+1} \cdot a_{1j} \cdot det(A_{1j})$,
  wobei $A_{1j}$ die $((n-1) \times (n-1))$-Matrix ist, die aus $A$ durch Streichen der ersten Zeile und der $j$-ten Spalte entsteht. (Eigentlich muss man auch hier $det_F(A)$ schreiben).

Es sind zwei Bemerkungen nötig:

1. Meist wird die Determinante in der Literatur auf andere Weise definiert, wozu aber als Vorbereitung mehr theoretischer Hintergrund nötig ist. Die obige, als Definition verwendete Gleichung muss dann als Satz bewiesen werden.

2. Bei der obigen Formel sagt man, man habe $det(A)$ "nach der ersten Zeile entwickelt". Es kann gezeigt werden, dass Entwicklung nach jeder anderen

Zeile (oder Spalte) genauso - mit der nötigen Abwandlung der Formel - möglich ist.

Eine $(n \times n)$-Matrix $A$ (über $F$) heißt **regulär**, wenn $rg_F(A) = n$ ist, andernfalls ist sie **singulär**. Eine wichtige Eigenschaft der Determinante ist es, dass mit ihrer Hilfe die Regularität einer quadratischen Matrix getestet werden kann: $A$ ist genau dann regulär, wenn $det(A) \neq 0$ gilt.

$A$ sei eine $(n \times m)$-Matrix über $F$, $x$ ein Element des Vektorraums $F^m$ (also ein $m$-tupel $(x_1, \ldots, x_m)$ mit $x_i \in F$). Fasst man $x$ als $(m \times 1)$-Matrix auf, so ergibt das Produkt $A \cdot x$ eine $(n \times 1)$-Matrix bzw. einen Vektor $y$ aus $F^n$, im Bild:

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1m} \\ a_{21} & a_{22} & \ldots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nm} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Man hat also eine Abbildung $f : F^m \to F^n$ durch $f(x) = A \cdot x$ definiert. Es ist leicht nachzurechnen, dass diese Abbildung stets linear ist.
Umgekehrt lässt sich sogar zeigen, dass jede lineare Abbildung von $F^m$ nach $F^n$ auf diese Weise (also durch die Wahl einer geeigneten Matrix) beschrieben werden kann.

# D Pascal-Programme zu den Algorithmen von Dijkstra und Kruskal

Im Folgenden werden für die Algorithmen von Dijkstra und von Kruskal ablauffähige Pascal-Programme vorgestellt. Die Art der gemachten Einschränkungen (z. B. auf maximal 100 Ecken des Graphen) kann natürlich je nach benutztem Rechner verschieden sein.

Die Programme erheben nicht den Anspruch, besonders gute Umsetzungen der Algorithmen zu sein. Im Programmieren geübte Leser werden ermuntert, uns bessere Lösungen zuzusenden.

**Algorithmus**: Program Dijkstra

```
PROGRAM dijkstra(input, output);

{  Konstantendeklarationen                          }

CONST
   anz_ecke  = 100;
   anz_kante = 100;
   infinity  = 2147483647;

TYPE
   t_ecke    = 0..anz_ecke;
   t_matrix  = ARRAY[1..anz_ecke,1..anz_ecke] of real;
   t_nodelist= ARRAY[1..anz_ecke] of t_ecke;

VAR    laenge        : real;
       e, s, t, ende : t_ecke;
       a             : t_matrix;
       path          : t_nodelist;

PROCEDURE init;

VAR i,j      : integer;

BEGIN

  writeln;
  writeln;
  write('Bitte die Anzahl der Ecken eingeben : ');
  readln(e);

  writeln;
  writeln('Bitte die Inzidenzmatrix zeilenweise eingeben : ');

  FOR i:=1 TO e DO
  BEGIN
    writeln;
```

```
    FOR j:=1 TO e DO
       read(a[i,j]);
  END;

  writeln;
  write('Bitte den Startpunkt eingeben : ');
  readln(t);

  writeln;
  write('Bitte den Endpunkt eingeben : ');
  readln(s);

END; { init }

PROCEDURE shortestpath;

{ Diese Prozedur findet den kuerzesten Weg von s nach t in der Matrix}
{ A und gibt ihn in PATH zurueck                                      }

TYPE   lab       = (perm,tent); { is label permanent or tentative ? }
       NodeLabel = RECORD
                     predecessor : t_ecke;
                     length      : real;
                     labl        : lab;
                   END;
       GraphState = ARRAY[1..anz_ecke] OF NodeLabel;
VAR    state : GraphState;
       i, k  : t_ecke;
       min   : real;

BEGIN

   FOR i := 1 TO e do
       WITH state[i] DO
       BEGIN
         predecessor := 0;
         length      := infinity;
         labl        := tent;
       END;
   state[t].length := 0;
   state[t].labl   := perm;
   k := t;    { k ist die Anfangsecke }

   repeat  { Gibt es einen besseren Weg von k aus ? }
     FOR i:=1 TO e DO
     BEGIN
       IF ( a[k,i] <> 0) AND (state[i].labl = tent) then
          IF state[k].length + a[k,i] < state[i].length then
          BEGIN
            state[i].predecessor := k;
            state[i].length      := state[k].length + a[k,i];
          END;
    END;

       { Finden der unbenutzten Ecke mit der kuerzesten Kante }
```

```
        min := infinity;
        k   := 0;
        FOR i:=1 TO e DO
        BEGIN
          IF (state[i].labl = tent) AND (state[i].length < min) THEN
          BEGIN
            min := state[i].length;
            k   := i;
          END;
        END;
        state[k].labl := perm;
    UNTIL k = s;

    { Kopieren des Weges in den Ausgabevektor }

    k := s;
    i := 0;

    laenge := state[k].length;

    REPEAT
      i := i + 1;
      path[i] := k;
      k := state[k].predecessor;
    UNTIL k = 0;

    ende := i;

END; { shortestpath }

PROCEDURE ausgabe;

VAR i   : integer;

BEGIN

  writeln;
  writeln;
  writeln('Der kuerzeste Weg von Ecke ',t:3,' nach Ecke ',
                  s:3,' lautet folgendermassen :');
  FOR i:=1 TO ende DO
      write(path[ende+1-i]:3,'  ');
  writeln;
  writeln('Die Laenge des Weges ist : ',laenge:10);

END; { ausgabe }

BEGIN

  init;
  shortestpath;

  ausgabe;
```

```
END. { dijkstra }
```

**Algorithmus**: Program Kruskal

```
PROGRAM kruskal(input, output);

{  Konstantendeklarationen                          }

CONST
   anz_ecke  = 100;
   anz_kante = 100;
   anz_komp  = 100;
   anz_k_min = 100;
{ Typedeklaration                                   }

TYPE
   t_e          =-1..anz_ecke;
   t_k          =0..anz_ecke;
   t_komp       =1..anz_ecke;
   t_fehler     =0..3;
   t_zaehler    =0..anz_k_min;
   t_grenze     =1..anz_k_min;
   t_bewertung  =ARRAY[t_grenze]of real;
   t_kante_bew  =ARRAY[t_k]of real;
   t_inzidenz   =ARRAY[t_e,1..2]of integer;
   t_komp_verz  =ARRAY[t_grenze]of integer;
   t_geruest    =ARRAY[t_grenze,1..2]of integer;

VAR
   e            :      t_e;          { Ecke }
   k            :      t_k;          { Kante }
   bewertet     :      boolean;      { Graph bewertet  }
   kante_bew     :      t_kante_bew;  { Kantenbewertung }
   inzidenz     :      t_inzidenz;
   geruest      :      t_geruest;
   bewertung    :      t_bewertung;  { Ausgabevektor   }
   komp_verz    :      t_komp_verz;  { Verzeichnis der }
                                     { Komponenten     }
   fehler       :      t_fehler;     { Fehlercode      }
   zaehler      :      t_zaehler;

PROCEDURE error(fehler: t_fehler);

{ Diese Prozedur wertet den Fehlercode aus, und gibt einen entsprechende }
{ Fehlermeldung aus. Je nach Fehler wird die Verarbeitung abgebrochen    }

BEGIN

   CASE fehler OF
   1 : writeln('Es sind keine Ecken angegeben worden!!');
   2 : BEGIN
        writeln('Bitte J oder N fuer die Anfrage nach der',
                'Bewertung des Graphen angeben.');
```

```
        writeln('Bitte erneute Eingabe');
      END;
   END;
END; { ERROR }



PROCEDURE init(VAR e         : t_e;
               VAR k         : t_k;
               VAR bewertet  : boolean;
               VAR kante_bew : t_kante_bew;
               VAR inzidenz  : t_inzidenz;
               VAR fehler    : t_fehler);

{ Diese Prozedur liest alle Daten ein, die fuer den Algorithmus  }
{ relevant sind. Zudem werden alle Variablen initialisiert.      }

VAR entscheid : char;
    i         : integer;
    flag      : boolean;

BEGIN
  FOR i:=1 TO anz_kante DO
    kante_bew[i]:=1;

  writeln;
  write('Bitte die Anzahl der Ecken eingeben : ');
  readln(e);
  IF ( e = 0 )
     THEN fehler := 1
     ELSE IF ( e > 0 )
     THEN BEGIN
       writeln;
       writeln('Bitte die Anzahl der Kanten angeben : ');
       readln(k);
       writeln;
       writeln;
       writeln('Ist der Graph bewertet ?');
       write('Bitte J oder N angeben:');
       readln(entscheid);

       CASE entscheid OF

       'J','j' : bewertet := TRUE;
       'N','n' : bewertet := FALSE;
       ELSE
         fehler := 2;
       END;

       { Initialisieren des Komponentenverzeichnisses }

       IF ( fehler = 0 ) THEN
       BEGIN
         FOR i := 1 TO anz_k_min DO
```

```
            komp_verz[i] := i;

          FOR i := 1 TO K DO
          BEGIN
            IF bewertet THEN
            BEGIN
              writeln;
              writeln('Bitte die Bewegung der n„chsten Kanten angeben');
              readln(kante_bew[i]);
            END;

            { Einlesen der inzidierten Ecken }

            writeln;
            writeln('Bitte die inzidenten Eckpunkte angeben :');

            readln(inzidenz[i,1],inzidenz[i,2]);
          END; { FOR i := 1 .. }

        END; { IF fehler = 0 }

      END; { THEN BEGIN }

  END; { INIT }

PROCEDURE sort(VAR kante_bew : t_kante_bew;
               VAR inzidenz  : t_inzidenz;
               k             : t_k);

{ Diese Prozedur sortiert den Graphen nach aufsteigender   }
{ Kantenbwertung                                           }

VAR i,l,ll  : integer;
    j,jj    : integer;
    min,kk  : real;

BEGIN
    FOR i := 1 TO (k-1) DO
    BEGIN
      min := 214748364;
      FOR j := i TO k DO
      BEGIN
        IF ( kante_bew[j] - min < 0 )
        THEN BEGIN
          jj := j;
          min := kante_bew[j];
        END;
      END;
    kk            := kante_bew[i];
    kante_bew[i]  := kante_bew[j];
    FOR l := 1 TO 2 DO
    BEGIN
    ll            := inzidenz[i,l];
    inzidenz[i,l] := inzidenz[jj,l];
    inzidenz[jj,l]:= ll;
```

```
    END; { FOR l := ...}
    END; { FOR i := ...}
END; { BEGIN }

PROCEDURE algorithmus(k             : t_k;
                      e             : t_e;
                      kante_bew     : t_kante_bew;
                      inzidenz      : t_inzidenz;
                      VAR komp_verz : t_komp_verz;
                      VAR geruest   : t_geruest;
                      VAR bewertung : t_bewertung);

{ Diese Prozedur ist der Kern der Algortihmus. Sie nimmt immer dann }
{ die naechste Kante zum Geruest hinzu, wenn sie mit den bisher     }
{ gewaehlten keinen Kreis bildet.                                   }

VAR kk,ll,l,m  : t_zaehler;
    i,j        : integer;

BEGIN
    zaehler := 0;

    FOR i :=1 TO anz_k_min DO
    BEGIN
      l := inzidenz[i,1];
      m := inzidenz[i,2];
      IF ( komp_verz[m] - komp_verz[l] <> 0 )
      THEN BEGIN
        zaehler := zaehler + 1;
        ll      := komp_verz[l];
        kk      := komp_verz[m];
        geruest[zaehler,1] :=m;
        geruest[zaehler,2] :=l;
        bewertung[zaehler] := kante_bew[i];
        FOR j := 1 TO e DO
        BEGIN
          IF ( komp_verz[j] - kk = 0 )
          THEN komp_verz[j] := ll;
        END;
      END; { THEN BEGIN }
    END; { I-Schleige }

END; { Algorithmus}


PROCEDURE ausgabe( VAR komp_verz : t_komp_verz;
                   geruest       : t_geruest;
                   bewertung     : t_bewertung;
                   e             : t_e);

VAR hilf               : t_komp_verz;
    ia,js,is,z,i,nr,j  : integer;
    summe              : real;

BEGIN
```

```
  writeln;
  writeln('Komponenten');
  is := 0;
  FOR i := 1 TO e DO
  BEGIN
    IF ( komp_verz[i] <> 0 )
    THEN BEGIN
      ia := komp_verz[i];
      js := 0;
      FOR j :=1 TO e DO
      BEGIN
        IF ( komp_verz[i] - ia = 0 )
        THEN BEGIN
          js           := js + 1;
          hilf[js]     := j;
          komp_verz[j] := 0;
        END;
      END; { FOR j ..}

      is := is + 1;
      writeln;
      writeln(is,'-te','  ');
      writeln;

      FOR z := 1 TO js DO
        write('            ',hilf[z]:3,'  ');
    END;
  END; { FOR i:= .. }

  writeln;
  writeln;
  writeln('Geruest');
  writeln;
  nr := e - is;
  FOR i := 1 TO nr DO
  BEGIN
    write(' Kante (',geruest[i,1]:3,',',geruest[i,2]:3,')');
    writeln('mit der Kantenbewertung:',bewertung[i]:10:6);
  END;

{ Berechnung der Summe der Kantenbewertungen des Minimalgeruestes }

  summe := 0;
  FOR i := 1 TO anz_k_min DO
    summe := summe + bewertung[i];

  writeln;
  writeln('Die Summe der Kantenbewertungen des Minimalgeruestes',
              ' lautet: ',summe:10:6);
  END; { Ausgabe }


BEGIN { Hauptprogramm }

 e := 0;
```

```
 writeln;

 WHILE ( e>= 0) DO
 BEGIN
   writeln('Der Algorithmus ist mit -1 fuer die Anzahl ',
               'der Ecken abzubrechen !');

   fehler := 0;
   init(e,k,bewertet,kante_bew,inzidenz,fehler);
   IF ( fehler <> 0)
     THEN error(fehler)
     ELSE BEGIN
       IF bewertet
          THEN sort(kante_bew,inzidenz,k);

       algorithmus(k,e,kante_bew,inzidenz,komp_verz,geruest,
                   bewertung);
       ausgabe(komp_verz,geruest,bewertung,e);

       END;
   END;
END. { Hauptprogramm }
```

# E Pascal-Programm zum Algorithmus von Ford und Fulkerson

Im folgenden wird für den Algorithmus von Ford und Fulkerson ein ablauffähiges Pascal-Programm vorgestellt.

Als Einschränkung soll hier der gegebene Digraph nicht mehr als $50$ Ecken haben. Die Eingabe des mit der Kantenbewertung $c$ (Kapazität) versehenen Digraphen erfolgt durch die bewertete Adjazenzmatrix: an der $j$-ten Stelle der $i$-ten Zeile wird $c(ij)$ eingetragen, falls es eine gerichtete Kante $ij$ von Ecke $i$ zu Ecke $j$ gibt, ansonsten der Wert $0$.

Der Leser sollte sich davon überzeugen, dass es sich hier nicht um eine Implementierung des Algorithmus von Edmonds und Karp handelt.

**Algorithmus**: Algorithmus von Ford und Fulkerson

```
PROGRAM ford_fulkerson(input, output);

{ Dieses Programm berechnet einen maximalen Fluss bzw. einen Schnitt }
{ minimaler Kapazitaet in einem Netzwerk.                             }
{ Die grundlegende Idee des Algorithmus von Ford und Fulkerson ist    }
{ es, von q ausgehend schrittweise diejenigen Ecken zu markieren,     }
{ zu denen es von q aus einen zunehmenden Weg gibt. Wird s mar-       }
{ kiert, so kann der vorliegende Fluss vergroessert werden.           }

CONST
  anz_ecken = 50;
  unendlich =32767;      { Fr Compiler mit 16-Bit Integerarithmetik }
           {=2147483647;   Fr Compiler mit 32-Bit Integerarithmetik }

VAR
  i, g       : integer;      { Zaehlervariable }
  e          : 1..anz_ecken; { Anzahl der Ecken des Netzes }
  q          : integer;      { Quelle }
  s          : integer;      { Senke }
  fluss     : integer;      { aktueller Fluss }
  ecke       : 1..anz_ecken; { Ecke, von der aus weitere Wege gesucht werden }

  wert1      : integer; { Vergleichswert fuer die Suche des Minimums: }
                        { Hier: Kapazitaet - Fluss einer Kante        }
  wert2      : integer; { Vergleichswert fuer die Suche des Minimums: }
                        { Hier: moegliche Flussvergroesserung bis zur }
                        { Vorgaengerecke }
  minimum    : integer; { Kleinerer Wert von wert1 und wert2 }

  { Die Markierung soll fuer jede Ecke ein 3-Tupel sein. In der ersten  }
  { Komponente steht der Vorgaenger der Ecke, in der zweiten Komponente }
  { gibt es drei Moeglichkeiten:                                        }
  {      0 : Initialwert                                                }
  {      1 : positiver Fluss, also in Laufrichtung                      }
  {      2 : negativer Fluss, also in Umkehrrichtung                    }
```

```
  {                                                                    }
  { In der dritten Komponente steht der Wert, um den der Fluss bis zur  }
  { betreffenden Ecke verbessert werden kann                           }

  markierung : ARRAY[1..anz_ecken,1..3] of integer;

 { Das Netzwerk, das in der 1. Komponente die Kapazitaet der Kante und  }
 { in der 2. Komponente den Fluss enthaelt }

  netz    : ARRAY[1..anz_ecken,1..anz_ecken,1..2] of integer;
  u       : ARRAY[1..anz_ecken] of integer;  { gibt an, welche der      }
{ markierten Ecken bereits }
{ abgesucht wurden         }
  d       : ARRAY[1..anz_ecken] of integer;  { moegliche Flussverbes-   }
{ serung bis zur           }
                                             { zur Vorgaengerecke       }
  abbruch : BOOLEAN; { Abbruchbedingung fuer die Suche nach Wegen       }


PROCEDURE initialisierung;

VAR
   i, j : integer;

BEGIN { initialisierung }

  q     := 0;
  s     := 0;
  fluss := 0;

  FOR i:=1 TO anz_ecken DO
  BEGIN
    u[i]            := 0;
    d[i]            := unendlich;
    markierung[i,1] := 0;
    markierung[i,2] := 0;
    markierung[i,3] := unendlich;
  END;

  write('Bitte die Anzahl der Ecken eingeben: ');
  readln(e);

  FOR i:=1 TO e DO
  BEGIN
    writeln('Bitte die ',i:2,' -te Zeile der bew. Netzmatrix eingeben: ');
    writeln;
    FOR j:=1 TO e DO
    BEGIN
      netz[i,j,2] := 0;
      read(netz[i,j,1]);
    END;
  END;

  writeln;
  writeln;
```

```
  write('Bitte die Quelle eingeben: ');
  readln(q);

  writeln;
  writeln;
  write('Bitte die Senke eingeben: ');
  readln(s);

  { Die Quelle q wird mit (-1,0,unendlich) markiert }

  markierung[q,1] := -1; { Quelle hat keinen Vorgaenger }
  markierung[q,2] := 2;
  markierung[q,3] := unendlich;

END;  { initialisierung }

PROCEDURE wahl_markierte_n_abgesuche_ecke;

VAR
   i        : integer;
   gefunden : BOOLEAN;

BEGIN
  gefunden := FALSE;
  i        := 1;
  REPEAT

    { Ist die Ecke markiert, aber noch nicht abgesucht }

    IF (markierung[i,1] <> 0) AND (u[i] = 0) THEN
    BEGIN
      ecke     := i;
      gefunden := TRUE;
    END
    ELSE i := i + 1;
  UNTIL gefunden;
END;

PROCEDURE markiere(zu_markierende_ecke : integer;
                   vorgaenger          : integer;
                   flussrichtung       : integer;
                   differenz           : integer);

BEGIN
  markierung[zu_markierende_ecke,1] := vorgaenger;
  markierung[zu_markierende_ecke,2] := flussrichtung;
  markierung[zu_markierende_ecke,3] := differenz;
END;


PROCEDURE absuchen_der_vorwaertskanten;

VAR
  g :integer;
```

```
BEGIN

  { Untersuche alle Kanten, die von der Ecke ECKE weggehen, unter der }
  { Bedingung, dass die adjazente Ecke noch nicht markiert ist und    }
  { der Fluss auf dieser Kante noch verbessert werden kann, d.h. er   }
  { ist bisher kleiner als die Kapazitaet der Kante                   }

  FOR g:=1 TO e DO
  BEGIN
    { Alle zu ecke adjazenten Ecken, nicht die Ecke ecke selbst }
    IF (g <> ecke) AND (netz[ecke,g,1] > 0) THEN
    BEGIN
      { Fuer Ecke g ist noch kein Vorgaenger eingetragen und der bis-  }
      { Fluss auf der Kante [ecke,g] ist kleiner als deren Kapazitaet  }

      IF (markierung[g,1] = 0)
         AND (netz[ecke,g,2] < netz[ecke,g,1]) THEN
      BEGIN
        wert1 := netz[ecke,g,1] - netz[ecke,g,2];
        wert2 := markierung[ecke,3];
        IF wert1 < wert2 THEN minimum := wert1
                         ELSE minimum := wert2;
        d[g] := minimum;
        markiere(g,ecke,1,d[g]);
      END;
    END; { if g <> ecke }
  END;{ Durchsuche alle Kanten }

END; { positiven_fluss_suchen }

PROCEDURE absuchen_der_rueckwaertskanten;

VAR
   g :integer;

BEGIN

  { Untersuche alle Kanten, die zu der Ecke ECKE hinfuehren, unter der }
  { Bedingung, dass die adjazente Ecke noch nicht markiert ist und der }
  { bisherige Fluss auf dieser Kante groesser Null ist.                }

  FOR g:=1 TO e DO
  BEGIN
    { Alle zu ecke adjazenten Ecken, nicht die Ecke ecke selbst }
    IF (g <> ecke) AND (netz[g,ecke,1] > 0) THEN
    BEGIN
      { Fuer Ecke g ist noch kein Vorgaenger eingetragen und der Fluss }
      { der Kante [g,ecke] ist groesser Null                           }

      IF (markierung[g,1] = 0)
         AND (netz[g,ecke,2] > 0) THEN
      BEGIN
        wert1 := netz[g,ecke,2];
        wert2 := markierung[ecke,3];
        IF wert1 < wert2 THEN minimum := wert1
```

```
                        ELSE minimum := wert2;
        d[g] := minimum;
        markiere(g,ecke,2,d[g]);
      END;
    END; { Kante noch nicht markiert + der Fluss < Kapazitaet der Kante }
  END; { alle Ecken fuer den negativen Fluss durchsuchen }
END; { negativen_fluss_suchen }

PROCEDURE fluss_vergroessern;

VAR
   g, i : integer;

BEGIN
  ecke := s;
  WHILE ecke <> q DO
  BEGIN
    g := markierung[ecke,1];
    IF markierung[ecke,2] = 1
       THEN  netz[g,ecke,2] := netz[g,ecke,2] + markierung[s,3]
       ELSE  netz[ecke,g,2] := netz[ecke,g,2] - markierung[s,3];

    ecke := g;
  END; { while }

  fluss := fluss + markierung[s,3];


  { Alle Markierungen loeschen, ausser der von q und Zuruecksetzen }
  { der Arrays u und d                                             }

  FOR i:=1 TO e DO
  BEGIN
    IF i <> q THEN
    BEGIN
      markierung[i,1] := 0;
      markierung[i,2] := 0;
      markierung[i,3] := unendlich;
    END;
    u[i] := 0;
    d[i] := unendlich;
  END;
END;


PROCEDURE ausgabe;

VAR
   i, j : integer;

BEGIN
  writeln;
  writeln('Eingegebenes Netz: ');
  writeln;
  FOR i:=1 TO e DO
```

```
    BEGIN
      writeln;
      FOR j:=1 TO e DO
      BEGIN
        write(netz[i,j,1]:4,' ');
      END;
    END;

    writeln;
    writeln;
    writeln('Der maximale Fluss lautet: ',fluss:4);
    writeln;

    writeln('Flussmatrix: ');
    writeln;
    FOR i:=1 TO e DO
    BEGIN
      writeln;
      FOR j:=1 TO e DO
      BEGIN
        write(netz[i,j,2]:4,' ');
      END;
    END;

  END; { Ausgabe }

  BEGIN { HP }

    initialisierung;

    REPEAT
      wahl_markierte_n_abgesuche_ecke;

      absuchen_der_vorwaertskanten;

      absuchen_der_rueckwaertskanten;

      { Ecke ECKE als bereits abgesucht markieren }

      u[ecke] := 1;

      { Ist die Senke markiert? }

      IF markierung[s,3] <> unendlich THEN fluss_vergroessern;

      { Abbruchbedingung fuer die Repeat-Schleife. Abbruch wird generell  }
      { zuerst auf TRUE gesetzt. Die Schleife soll abbrechen, wenn fuer   }
      { alle markierten Ecken u[e] =1 ist, d.h. bereits abgesucht.        }

      abbruch := TRUE;
      i       := 0;
      REPEAT
        i := i + 1;
        IF (markierung[i,1] <> 0) AND (u[i] = 0) THEN abbruch := FALSE;
      UNTIL (i = e) OR (NOT abbruch);
```

```
  UNTIL abbruch;
  ausgabe;
END.  { HP }
```

# F Boolesche Ausdrücke und ihre Umformung

**Boolesche Variable**

Gegeben seien **Boolesche Variablen** $x_1, ..., x_n$, die die Werte 1 (für "wahr") und 0 (für "falsch") annehmen können.

Ein **Boolescher Ausdruck** ist irgendein Ausdruck, den man aus diesen Variablen sowie den Konstanten 0 und 1 mit Hilfe der Verknüpfungen

$\wedge$ (und)
$\vee$ (oder)
$'$ (nicht)

und unter Verwendung von Klammern "in sinnvoller Weise" bilden kann. (Auf eine exakte (induktive) Definition wird hier verzichtet.)

Boolesche Ausdrücke sind z. B. $x_1 \vee x_2,\ \ x_1' \wedge (x_2 \vee (x_3 \wedge x_1)')',\ \ x_1 \vee 0$; kein Boolescher Ausdruck, da nicht sinnvoll, ist etwa $x_1(\wedge x_2 \vee' x_3)$.

Um zu verdeutlichen, dass ein Boolescher Ausdruck $f$ von den Variablen $x_1, ..., x_n$ abhängt, schreiben wir auch $f(x_1, ..., x_n)$.

Setzt man für die Variablen $x_i$ jeweils 0 oder 1 ein, so nimmt auch $f$ einen dieser beiden Werte an; dabei arbeiten die elementaren Verknüpfungen $\wedge, \vee$ und $'$ so, wie in den folgenden Tafeln dargestellt:

| $\wedge$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| $\vee$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

| $'$ | 0 | 1 |
|---|---|---|
| | 1 | 0 |

**Variablenbelegungen**

Ist z. B. $f(x_1, x_2, x_3) = x_1 \wedge (x_2 \vee x_3')$, so gibt es acht mögliche **Variablenbelegungen** $\alpha \in \{0,1\}^3$, deren Einsetzen in $f$ jeweils den Wert $f^\alpha$ (der 0 oder 1 ist) liefert. Für $\alpha = (1,0,1)$ ergibt sich hier z. B. $f^\alpha = 1 \wedge (0 \vee 1') = 0$. Allgemein wird für $f(x_1, ..., x_n)$ die Menge aller $\alpha \in \{0,1\}^n$ mit $f^\alpha = 1$ als der **Träger** von $f$ (kurz: $T(f)$) bezeichnet.

**Träger**

**Wahrheitstafel**

Um den Träger von $f(x_1, x_2, x_3) = x_1 \wedge (x_2 \vee x_3')$ systematisch zu ermitteln, kann man eine **Wahrheitstafel** für $f$ aufstellen, in der man alle möglichen $\alpha$ und die zugehörigen $f^\alpha$ einträgt. Die Spalten 2 und 3 haben hier nur Hilfsfunktion:

| $\alpha = (x_1, x_2, x_3)$ | $x_3'$ | $x_2 \vee x_3'$ | $f^\alpha$ |
|---|---|---|---|
| 0 0 0 | 1 | 1 | 0 |
| 0 0 1 | 0 | 0 | 0 |
| 0 1 0 | 1 | 1 | 0 |
| 0 1 1 | 0 | 1 | 0 |
| 1 0 0 | 1 | 1 | 1 |
| 1 0 1 | 0 | 0 | 0 |
| 1 1 0 | 1 | 1 | 1 |
| 1 1 1 | 0 | 1 | 1 |

Man liest ab, dass in diesem Beispiel gilt

$$T(f) \;=\; \{(1,0,0),(1,1,0),(1,1,1)\}.$$

Der Aufwand zur Ermittlung des Trägers mittels einer Wahrheitstafel steigt offenbar exponentiell mit der Anzahl $n$ der vorkommenden Variablen. Jedoch muss jeder andere Algorithmus, der die Träger Boolescher Ausdrücke berechnet und niederschreibt, ebenfalls exponentiell sein, weil es $2^n$ viele mögliche Elemente des gesuchten Trägers gibt.

**äquivalente Boolesche Ausdrücke**

Verschiedene Boolesche Ausdrücke $f(x_1, ..., x_n)$ und $g(x_1, ..., x_n)$ können **äquivalent** sein in dem Sinne, dass jede Variablenbelegung für $f$ und $g$ dasselbe Resultat liefert: $f^\alpha = g^\alpha$ für alle $\alpha \in \{0,1\}^n$ ; wir schreiben dann $f \equiv g$.

Es gilt z. B.

$$\begin{aligned} x_1 \wedge (x_2 \vee x_3) &\equiv (x_1 \wedge x_2) \vee (x_1 \wedge x_3)\,, \\ x_1 \vee 0 &\equiv x_1\,, \\ x_2''' &\equiv x_2'\,. \end{aligned}$$

**Axiome der Booleschen Algebra**

Man kann beweisen, dass zwei Boolesche Ausdrücke genau dann äquivalent sind, wenn sie sich durch Anwendung der folgenden Regeln ineinander überführen lassen. Diese Regeln stellen die **Axiome der Booleschen Algebra** dar:

$$\begin{aligned} x \vee x &\equiv x \\ x \vee y &\equiv y \vee x \\ x \vee (y \vee z) &\equiv (x \vee y) \vee z \\ x \vee (x \wedge y) &\equiv x \\ x \vee (y \wedge z) &\equiv (x \vee y) \wedge (x \vee z) \\ x \vee 0 &\equiv x \\ x \vee 1 &\equiv 1 \\ x \vee x' &\equiv 1 \end{aligned} \qquad \begin{aligned} x \wedge x &\equiv x \\ x \wedge y &\equiv y \wedge x \\ x \wedge (y \wedge z) &\equiv (x \wedge y) \wedge z \\ x \wedge (x \vee y) &\equiv x \\ x \wedge (y \vee z) &\equiv (x \wedge y) \vee (x \wedge z) \\ x \wedge 1 &\equiv x \\ x \wedge 0 &\equiv 0 \\ x \wedge x' &\equiv 0 \end{aligned}$$

**de Morgansche Regeln**

Besonders wichtig für das Rechnen mit Booleschen Ausdrücken sind noch die **de Morganschen Regeln**

$$(x \vee y)' \equiv x' \wedge y' \qquad\qquad (x \wedge y)' \equiv x' \vee y'$$

**Regel für die doppelte Verneinung**

sowie die **Regel für die doppelte Verneinung**

$$x'' \equiv x\ ,$$

die sämtlich aus obigen Axiomen ableitbar sind.

Oft besteht der Wunsch, alle vorkommenden Booleschen Ausdrücke in äquivalente Ausdrücke einer einheitlichen und besonders einfachen Form umzuwandeln.

**Disjunktive Normalform DNF**

Eine herausragende Stellung nehmen hier die **Disjunktiven Normalformen** (DNF) ein. Ein Boolescher Ausdruck ist eine DNF, wenn er aus einer Disjunktion ($\vee$ -Verknüpfung) von Konjunktionen ($\wedge$-Verknüpfungen) von (eventuell komplementierten) Variablen besteht.

DNF sind zum Beispiel

$$(x_1 \wedge x_2 \wedge x_3') \vee (x_2 \wedge x_5), (x_1 \wedge x_2) \vee (x_1 \wedge x_2'), (x_1 \wedge x_2) \vee x_3$$

jedoch ist

$$x_1 \wedge (x_2 \vee (x_3 \wedge x_4))$$

oder

$$(x_1 \wedge x_2'') \vee x_3$$

keine DNF.

**DDNF**

Eine DNF heißt **DDNF** (für disjunkte DNF), wenn die einzelnen Konjunktionen paarweise disjunkt sind (d. h. ihre $\wedge$-Verknüpfung 0 ergibt). Z.B. ist die DNF $(x_1 \wedge x_2) \vee (x_1 \wedge x_2')$ eine DDNF, denn $(x_1 \wedge x_2) \ \wedge (x_1 \wedge x_2') \equiv 0$.

**kanonische DNF**

Von besonderer Bedeutung ist noch die jedem Booleschen Ausdruck eindeutig zugehörige **kanonische DNF** – dies ist eine DDNF, in der in jeder Konjunktion alle Variablen $x_1, ..., x_n$ (eventuell komplementiert) vorkommen.

Es sei der Boolesche Ausdruck $f(x_1, x_2, x_3) = (x_1 \wedge x_2'') \vee x_3$ gegeben. Eine dazu äquivalente DNF ist $(x_1 \wedge x_2) \vee x_3$ , eine andere (die sogar DDNF ist) $(x_1 \wedge x_2 \wedge x_3') \vee x_3$. Die zugehörige kanonische DNF lautet
$(x_1 \wedge x_2 \wedge x_3') \vee (x_1 \wedge x_2 \wedge x_3) \vee (x_1' \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2' \wedge x_3) \vee (x_1' \wedge x_2' \wedge x_3)$.

Die kanonische DNF lässt sich über den Träger ermitteln.
Zu $\alpha \in \{0,1\}^n$ sei

$$k^\alpha := x_1^{\alpha(1)} \wedge ... \wedge x_n^{\alpha(n)},$$

wobei $x_i^1 = x_i$ und $x_i^0 = x_i'$ vereinbart sein soll.

Damit gilt:

**Satz:** *Für einen beliebigen Booleschen Ausdruck $f(x_1, ..., x_n)$ ist*

$$\bigvee_{\alpha \in T(f)} k^\alpha$$

*die zugehörige kanonische DNF.*

Es leuchtet sofort ein, dass man jeden Booleschen Ausdruck mit Hilfe der oben aufgeführten Axiome – insbesondere sind Distributivgesetze, de Morgansche Regeln und Regel für die doppelte Verneinung interessant – in eine zu ihm äquivalente DNF umwandeln kann. Ein Algorithmus zur Berechnung der kanonischen DNF wird in jedem Falle exonentiell sein. In Anwendungen ist man allerdings oft an einer DDNF interessiert, die – im Gegensatz zur kanonischen DNF – auch noch möglichst kurz sein sollte.

In diesem Zusammenhang ist auch das folgende Komplexitätsresultat von Interesse (vgl. [GAR]):

Das Entscheidungsproblem, ob bei gegebenen Variablen $x_1, ..., x_n$, Teilmenge $A \subseteq \{0,1\}^n$ von Variablenbelegungen und natürlicher Zahl $k > 0$ es eine DNF aus höchstens $k$ vielen Konjunktionen und mit $A$ als Träger gibt, ist NP-vollständig.

Literaturempfehlungen zu dieser Thematik sind [GUM] und [WEG].

**Boolesche Ausdrücke und Wahrscheinlichkeiten**

Es seien unabhängige Ereignisse $E_1, ..., E_n$ eines Wahrscheinlichkeitsraumes mit bekannten Wahrscheinlichkeiten $P\{E_i\}$ $(1 \leq i \leq n)$ gegeben. Bildet man aus den $E_i$ durch Anwendungen der logischen Operationen "und", "oder" und "nicht" ein neues Ereignis $E$, so lässt sich dies auch so auffassen: man hat in einen Booleschen Ausdruck $f(x_1, ..., x_n)$ für die Variablen $x_i$ die Ereignisse $E_i$ eingesetzt, um $E = f(E_1, ..., E_n)$ zu erhalten.

Es stellt sich nun die Frage, wie man $P\{E\}$ bzw. $P\{f(E_1, ..., E_n)\}$ berechnen kann.

Diese Frage hat im Prinzip eine einfache Antwort, die in den folgenden beiden Gesetzen der Wahrscheinlichkeit begründet ist:

I. Für sich gegenseitig ausschließende Ereignisse $F_1, ..., F_k$ gilt stets

$$P\{F_1 \vee ... \vee F_k\} = \sum_{i=1}^{k} P\{F_i\}.$$

II. Für unabhängige Ereignisse $F_1, ...F_k$ gilt stets

$$P\{F_1 \wedge ... \wedge F_k\} = \prod_{i=1}^{k} P\{F_i\}.$$

Daraus wird klar, dass man $P\{f(E_1, ..., E_n)\}$ über eine zu $f$ äquivalente DDNF unmittelbar berechnen kann.

Als Beispiel seien unabhängige Ereignisse $E_1, E_2, E_3$ gegeben mit bekannten Wahrscheinlichkeiten $P\{E_i\} = p_i$. Soll $P\{E\}$ für das Ereignis $E = (E_1 \wedge E_2) \vee E_3$

bestimmt werden, so betrachtet man zunächst den Booleschen Ausdruck $f = (x_1 \wedge x_2) \vee x_3$. Eine zu $f$ äquivalente DDNF ist, wie oben angemerkt,

$$(x_1 \wedge x_2 \wedge x_3') \vee x_3.$$

Also gilt $P\{E\} = p_1 p_2 (1 - p_3) + p_3$.

Die soeben skizzierte Methode zur Bestimmung von $P\{E\}$ birgt, wie bereits gesagt, das Problem in sich, die kanonische DNF zu bestimmen, was i.a. sehr aufwändig ist. Mitunter gibt man sich jedoch mit Abschätzungen für $P\{E\}$ zufrieden. Ein typischer Fall für eine solche Abschätzung ist folgender:

**Siebformel**
**Prinzip der Inklusion – Exklusion**

Gegeben seien (nicht als unabhängig vorausgesetzte) Ereignisse $E_1, ..., E_n$, es sei $E$ das Ereignis $E_1 \vee ... \vee E_n$. Die **Siebformel** (auch als **Prinzip der Inklusion – Exklusion** bekannt) besagt:

$$P\{E_1 \vee ... \vee E_n\} = \sum_i P\{E_i\} - \sum_{i,j} P\{E_i \wedge E_j\} + \sum_{i,j,k} P\{E_i \wedge E_j \wedge E_k\} - ...$$
$$+(-1)^{n+1}\, P\{E_1 \wedge ... \wedge E_n\}$$

Man beachte, dass $n = 2$ die bekannte Formel

$$P\{E_1 \vee E_2\} = P\{E_1\} + P\{E_2\} - P\{E_1 \wedge E_2\}$$

liefert.

Ist es nun bei einem konkreten Problem so, dass die Wahrscheinlichkeiten der UND-Verknüpfungen ab einer gewissen Anzahl beteiligter $E_i$ verschwindend klein werden, so kann der ab diesem Summenzeichen "abgeschnittene" Ausdruck u. U. als Schätzwert für $P\{E\}$ verwendet werden.

Ist also z. B. für alle $i, j, k$ $P\{E_i \wedge E_j \wedge E_k\}$ sehr klein, so gilt (unter geeigneten Bedingungen)

$$P\{E_1 \vee ... \vee E_n\} \approx \sum_i P\{E_i\} - \sum_{i,j} P\{E_i \wedge E_j\}.$$

Genauer gesagt: die Abweichung dieses Schätzwertes vom korrekten Wert $P\{E_1 \vee ... \vee E_n\}$ kann höchstens so groß sein wie der erste weggelassene Summand, welches hier

$$\sum_{i,j,k} P\{E_i \wedge E_j \wedge E_k\}$$

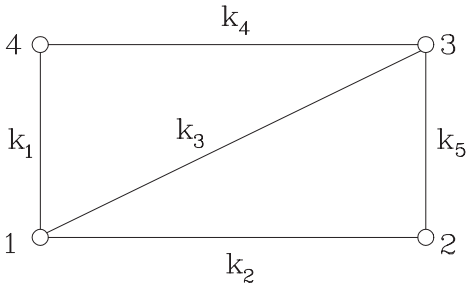
ist.

**Boolesche Ausdrücke, Funktions- und Fehlerbäume**

**Blockschaltbild**
**Funktionsbaum**
**Fehlerbaum**

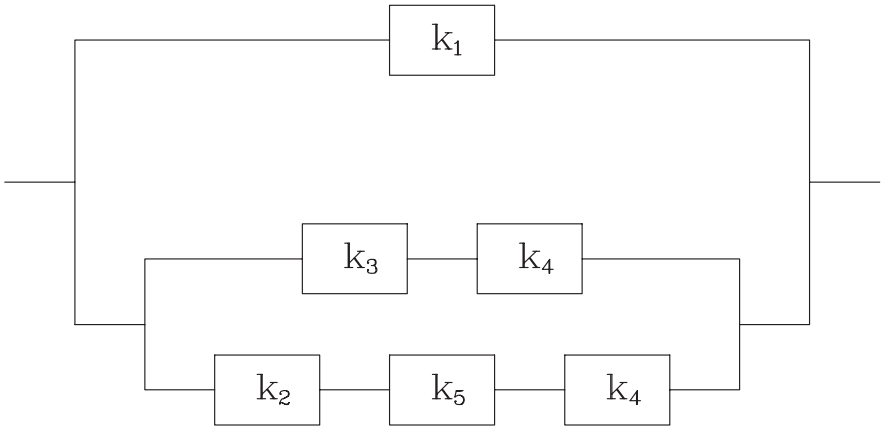
In der technischen Zuverlässigkeitstheorie ist es üblich, den die Funktionsfähigkeit eines Systems darstellenden Booleschen Ausdruck mit Hilfe von **Blockschaltbildern**, **Funktions**- oder **Fehlerbäumen** darzustellen. Diese Begriffe sollen an einem Beispiel illustriert werden. An einer genaueren Darstellung interessierte Leser werden auf [SCH1] oder [SCH2] verwiesen.

Es wird wieder das folgende Netz betrachtet:



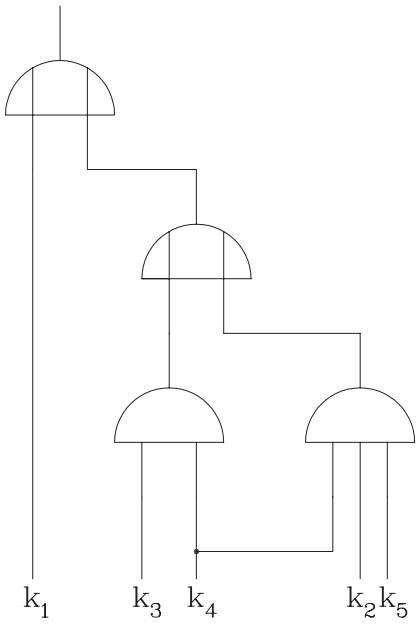

Die Kanten $k_i$ sind die ausfallgefährdeten Systemkomponenten. Die Funktionsfähigkeit des Systems sei gegeben dadurch, dass die Stationen 1 und 4 längs irgendeines Weges miteinander kommunizieren können.

Das die Funktionsfähigkeit charakterisierende Blockschaltbild sieht folgendermaßen aus:

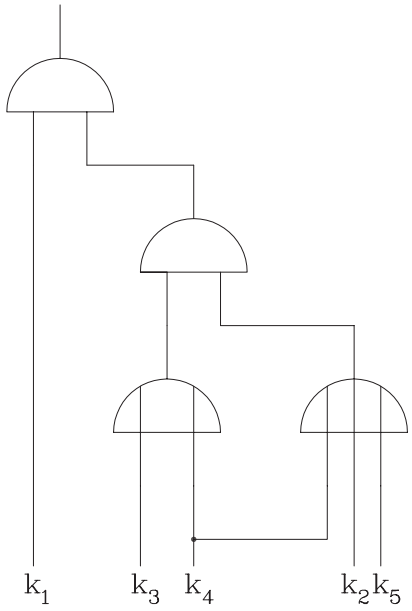



Mit der üblichen Vereinbarung, dass die UND- und ODER-Schaltungen durch die Diagramme


dargestellt werden, hat man dazu folgenden Funktionsbaum:




Als Fehlerbaum ergibt sich entsprechend:




Zur Bestimmung der interessierenden Wahrscheinlichkeiten gibt es in der Zuverlässigkeitstheorie eigene Methoden, die an diesen graphischen Darstellungen orientiert sind, inhaltlich jedoch den oben geschilderten Methoden entsprechen.

# G Gerüste eines Graphen

In Kapitel 6 hatten wir gezeigt, dass ein vollständiger Graph mit $n$ Ecken $n^{n-2}$ viele verschiedene Gerüste enthält. Für beliebige Graphen $G = (E, K))$ mit $n$ Ecken und $m$ Kanten gibt es keine geschlossene Formel für die Anzahl der Gerüste, jedoch lässt sich diese Anzahl mit dem folgenden Verfahren leicht bestimmen.

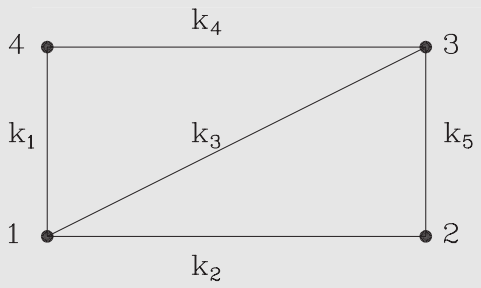
Die Ecken von $G$ seien durchnummeriert, $E = \{e_1, ..., e_n\}$. Es wird nun die negative Adjazenzmatrix gebildet (vgl. Kapitel 2), auf der Hauptdiagonalen werden zusätzlich die Eckengrade eingetragen (vgl. Kapitel 1). Für die resultierende Matrix $M = (m_{ij})$ gilt also

$$m_{ij} = \begin{cases} \gamma(e_i) & , \quad falls \quad i = j \\ -1 & , \quad falls \quad ij \in K \\ 0 & , \quad sonst. \end{cases}$$

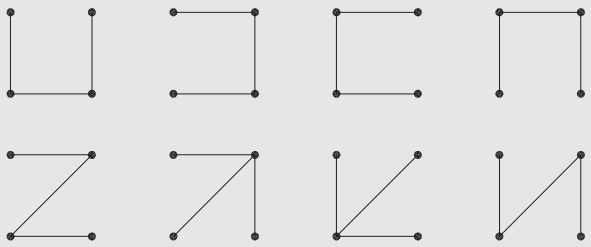
Durch Streichen der letzten Zeile und Spalte von $M$ ergibt sich eine Matrix $M'$. Es gilt nun: die Determinante von $M'$, $\det(M')$, ist die Anzahl der Gerüste von $G$.

Dieser Zusammenhang ist bereits im Jahre 1847 von G. Kirchhoff entdeckt worden (siehe [KIR]). In [BRO] wurde die Aussage formal aufgestellt und bewiesen.

**Beispiel:**
Wir betrachten wieder den folgenden Graphen $G$ :




Es wurde bereits an früherer Stelle festgestellt, dass $G$ 8 Gerüste besitzt. Die Gerüste sind hier dargestellt:



Für die Matrix $M$ ergibt sich

$$M = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix},$$

also ist

$$M' = \begin{pmatrix} 3 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 3 \end{pmatrix}.$$

Wie man leicht nachrechnet, gilt tatsächlich $det(M') = 8$.

Das soeben beschriebene Verfahren zur Ermittlung der Anzahl der Gerüste benötigt nur polynomialen Aufwand, denn Determinanten können effizient berechnet werden. Für die Aufzählung der Gerüste selbst kann es keinen polynomialen Algorithmus geben, da die mögliche Anzahl mit $n$ exponentiell wächst. Wir wollen ein Verfahren beschreiben, das die Gerüste auf elegante Weise bestimmt. Das Verfahren stammt von M. Piekarski (vgl. [PIE]) und ist auch in [DÖR] dargestellt.

Es werden auch die Kanten durchnummeriert, $K = \{k_1, ..., k_m\}$, und es wird die Inzidenzmatrix I gebildet (vgl. Kapitel 2).

Um die weitere Darstellung des Verfahrens zu veranschaulichen, wollen wir die Schritte immer gleich anhand des obigen Beispielgraphen vollziehen. Hier ergibt sich bisher

$$I = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Zu den Zeilen von I (außer der letzten) werden jetzt Matrizen $D_1, ..., D_{n-1}$ gebildet; $D_i$ ist die Matrix der Basisrektoren, deren Summe gerade die $i-te$ Zeile von I ergibt.

In unserem Beispiel erhalten wir

$$D_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

$$D_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$D_3 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Nun wird zunächst aus $D_1$ und $D_2$ eine neue Matrix $D_{1,2}$ gebildet, indem jede Zeile von $D_1$ zu jeder Zeile von $D_2$ addiert und das Ergebnis als Zeile von $D_{1,2}$ genommen wird; $D_{1,2}$ wird noch "reduziert", indem diejenigen Zeilen gestrichen werden,

die weniger als zwei Einsen enthalten oder in gerader Anzahl auftreten. Entsprechend wird danach $D_3$ zu $D_{1,2}$ "addiert" usw., bis alle $D_i (1 \leq i \leq n-1)$ abgearbeitet sind. Die Zeilen von $D_{1,2,\ldots,n-1}$, gelesen als Kantenmengen, korrespondieren dann zu den Gerüsten von $G$.

Für das Beispiel ergibt sich

$$D_{1,2} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{array}{l} \leftarrow \text{ 1. Zeile } D_1 + \text{ 1. Zeile } D_2 \\ \\ . \\ . \\ . \\ \\ \end{array}$$

bzw. nach Reduktion

$$D_{1,2} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Hinzuaddieren von $D_3$ führt zu

$$D_{1,2,3} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ \cancel{1} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} \\ \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} \\ 0 & 1 & 0 & 1 & 1 \\ \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{0} \\ \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{0} \\ 0 & 1 & 1 & 1 & 0 \\ \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} \\ \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{1} \\ 0 & 0 & 1 & 1 & 1 \\ \cancel{0} & \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} \end{pmatrix}$$

bzw. nach Reduktion

$$D_{1,2,3} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Die acht Zeilen dieser Matrix ergeben genau die acht oben dargestellten Gerüste.

Der entscheidende Schritt beim Beweis der Korrektheit dieses Verfahrens besteht darin, die folgende Aussage zu zeigen: jede Zeile im $D_{1,2,\ldots,n-1}$ , die einem nicht kreislosen Teilgraphen entspricht, tritt in gerader Anzahl auf.

# H Pascal-Programm zur Berechnung des Zuverlässigkeitspolynoms

Eckenzahl $n$ und Kantenzahl $m$ (die hier auf 6 und 15 stehen) müssen zu Anfang neu eingetragen werden. Sodann müssen nach Programmstart die Kanten eingegeben werden.

Der prinzipielle Programmablauf ist dann folgendermaßen:

es werden die Teilmengen der Menge aller Kanten abgesucht. Hat eine Teilmenge mindestens $n-1$ viele Elemente, so wird geprüft, ob der zugehörige Teilgraph zusammenhängend ist. Im positiven Falle wird ein Zähler erhöht, wobei für jede mögliche Kantenzahl $i$ $(n-1 \leq i \leq m)$ ein solcher Zähler geführt wird. Zum Schluss stehen also die Werte $F_i$ in diesen Zählern.

**Algorithmus**:

```
program
 graphzuv;

uses crt,tkbtestu;


const
    n = 6;
    m = 10;
    l = m-n+2;

type
    paare = record
            x: byte;
            y: byte;
            END;
var
   kanten          : array[1..m] of paare;
   i,i2,i3,i4,l1   : longint;
   za              : array[1..l] of longint;
   a,b,msk         : longint;
   anz,t_i         : byte;
   x               : array[1..anz_m] of byte;
   t               : array[1..n] of byte;
   st,lt           : byte;
   zus_fl,clr_kante_fl,noeintr_fl,loesch_fl,keinzu_fl: boolean;

begin
  clrscr;
  a := 1;
  b := 1;
  for i := 1 to (n-1) do
     a := 2*a;
  a := a-1;
  for i := 1 to m do
```

```
   b:= 2*b;
b := b-1;


for i := 1 to l do
    za[i] := 0;
for i := 1 to m do
    begin
    write(i,'. Kante (x,y) : ');
    read(kanten[i].x);
    gotoxy(25,wherey-1);
    readln(kanten[i].y);
    end;


laufzeit;
for i := a to b do
    begin
    for lt := 1 to m do
       x[lt] := 0;
    anz:=0;
    msk := 1;
    st := 1;
    lt := 1;
    while msk <= (b+1)/2 do
          begin
          if (msk and i) <> 0 then
            begin
            inc(anz);
            x[lt] := st;
            inc(lt);
            end;
    msk := 2*msk;
    inc(st);
    end;
zus_fl := false;
noeintr_fl := false;
if anz >= (n-1) then
   begin
   for i2 := 1 to n do
       t[i2] := 0;
   t[1] := kanten[x[1]].x;
   t[2] := kanten[x[1]].y;
   x[1] := 0;
   t_i := 2;
   loesch_fl := true;
   while loesch_fl do
         begin
         loesch_fl := false;
         for i2 := 2 to m do
             begin
             if x[i2] <> 0 then
                begin
                for i3 := 1 to t_i do
                    begin
```

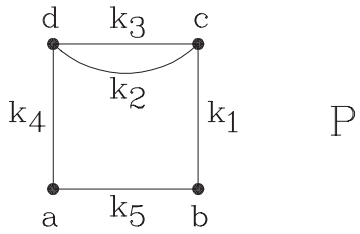
```
                    clr_kante_fl := false;
                    if x[i2] <> 0 then
                       begin
                       if (t[i3] = kanten[x[i2]].x) then
                          begin
                          clr_kante_fl := true;
                          noeintr_fl := false;
                          for i4 := 1 to t_i do
                              if t[i4] = kanten[x[i2]].y then
                                 noeintr_fl := true;
                              if not(noeintr_fl) then
                                 begin
                                 inc(t_i);
                                 t[t_i] :=  kanten[x[i2]].y;
                                 end;
                          end
                         else
                             begin
                             if (t[i3] = kanten[x[i2]].y) then
                                begin
                                clr_kante_fl := true;
                                keinzu_fl := false;
                                for l1 := 1 to t_i do
                                    begin
                                    if t[l1] = kanten[x[i2]].x then
                                       keinzu_fl := true;
                                    end;
                                if not keinzu_fl then
                                   begin
                                   inc(t_i);
                                   t[t_i] :=  kanten[x[i2]].x;
                                   end;
                             end;
                          end;
                    end;
                 if clr_kante_fl then
                    begin
                    x[i2] := 0;
                    loesch_fl := true;
                    end;
                 end;
              end;
           end;
     end;
  if t_i = n then
  zus_fl := true;
  end;
  if zus_fl then
     inc(za[anz+2-n]);
end;
i3 := 0;
i2 := 0;
clrscr;
for i := 1 to l do
    begin
```

```
    inc(i3);
    if i3 > 25 then
       begin
       i2 := i2 + 20;
       gotoxy(1,i2);
       end;
    writeln(' ',i+n-2,' K: ',za[i]);
    end;
laufzeit;
readln;
end.
```

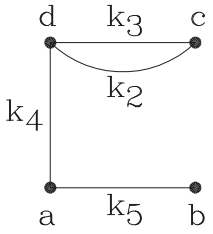
# Lösungshinweise zu Selbsttestaufgaben

***Lösung zu Selbsttestaufgabe 1.1-1:***

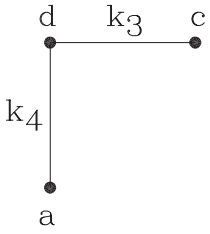
Bei einem Teilgraphen eines Pseudographen $P$ ist nur verlangt, dass er mit jeder Kante von $P$ auch die mit ihr in $P$ inzidenten Ecken enthält. Ein Untergraph muss umgekehrt auch mit zwei zu ihm gehörenden Ecken von $P$ alle Kanten zwischen diesen beiden Ecken enthalten. Im folgenden Beispiel ist von den Teilgraphen $P_1, P_2, P_3$ nur $P_3$ ein Untergraph von $P$: bei $P_1$ fehlt $k_1$, bei $P_2$ fehlt $k_2$.



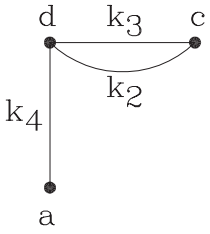

P




P1




P2

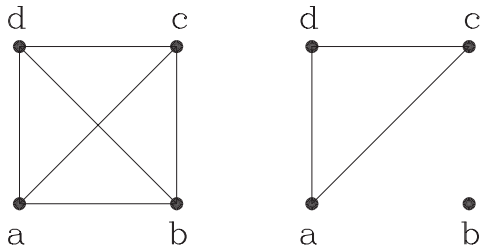



P3

***Lösung zu Selbsttestaufgabe 1.2-1:***

i. Ein Kreis ist definiert als geschlossener Kantenzug, der (von Anfangs- und Endecke abgesehen) keine Ecke zweimal enthält. Sind Durchlaufrichtung und Festlegung von Anfangs- bzw. Endecke nicht von Interesse, so kann ein Kreis in einem Pseudographen auch als Teilgraph mit bestimmten Eigenschaften aufgefasst werden; z. B. kann man einen Kreis dadurch charakterisieren, dass er ein zusammenhängender Teilgraph ist, in dem jede Ecke den Grad 2 hat.
Ein Baum ist ein zusammenhängender Graph, der keinen Kreis enthält.

ii. Jedes Gerüst von $K_4$ hat drei Kanten (und enthält alle vier Ecken). Es gibt $20$ Möglichkeiten, von den sechs Kanten von $K_4$ drei auszuwählen. Nur vier dieser Möglichkeiten führen zu Teilgraphen, die nicht zusammenhängend sind - eine ist unten im zweiten Diagramm dargestellt. Keine der anderen Möglichkeiten enthält einen Kreis. Also hat $K_4$ $16$ Gerüste.




$K_4$

***Lösung zu Selbsttestaufgabe 1.3-1:***

Besteht der Multigraph $M$ aus $k$ vielen Komponenten, so hat jeder spannende Wald $\rho(M) = n - k$ viele Kanten. Die Bedingungen $\mu(M) = m - n + k = 0$ ist gleichbedeutend mit $n - k = m$ und folglich genau dann erfüllt, wenn $M$ selbst ein Wald ist.

***Lösung zu Selbsttestaufgabe 2.1-1:***

Eine planare Darstellung eines Graphen ist eine Einbettung des Graphen in die Zeichenebene derart, dass sich keine der Linien, die die Kanten darstellen, überkreuzen. Ein Graph heißt planar, wenn für ihn eine solche planare Darstellung existiert. Dies bedeutet jedoch nicht, dass ein Diagramm eines planaren Graphen stets eine planare Darstellung sein muss.

***Lösung zu Selbsttestaufgabe 2.2-1:***

Streichen der ersten Zeile in $I$ (s. Beispiel 2.2-4) führt zu

$$I_r = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

mit $det\,(I_r) = 1$.

***Lösung zu Selbsttestaufgabe 2.3-1:***

Wählt man als Gerüst den Baum mit den Kanten $k_1, k_3, k_4$, so ergeben sich die fundamentalen Kreise $C_1 = \{k_1, k_2, k_3, k_4\}$, $C_2 = \{k_1, k_4, k_5\}$, $C_3 = \{k_1, k_4, k_6\}$. Weitere Elemente von $K$ sind $C_4 = C_1 + C_2, C_5 = C_1 + C_3, C_6 = C_2 + C_3$ und $C_7 = C_1 + C_2 + C_3$. Im Ergebnis hat man die Matrix

$$C(M) = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

***Lösung zu Selbsttestaufgabe 3.2-1:***

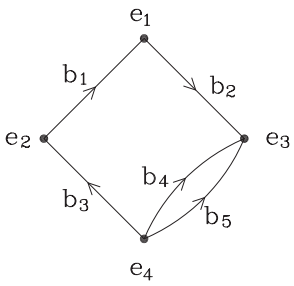
Die Komplexität $T_A(n)$ eines Algorithmus $A$ ist die Anzahl der im schlechtesten Fall auszuführenden elementaren Rechenoperationen bei einer Anwendung von $A$ auf einen Einzelfall von Größe $n$.

***Lösung zu Selbsttestaufgabe 3.3-1:***

Für die Lösung gewisser Probleme hat man keine polynomialen Algorithmen - sei es, dass solche mit Sicherheit nicht existieren, sei es, dass man bisher keine solchen finden konnte. In solchen Fällen setzt man sich häufig das Ziel, einen approximativen Algorithmus zu entwerfen, der keine hohe Komplexität hat und gute Näherungslösungen liefert.

***Lösung zu Selbsttestaufgabe 4.1-1:***

Ein Pseudodigraph ist azyklisch, wenn er keinen Zyklus (gerichteten Kreis) enthält, und kreisfrei, wenn der zugrundeliegende (ungerichtete) Pseudograph keinen Kreis enthält. Der folgende Multidigraph ist azyklisch, enthält jedoch Kreise, z. B. bilden die Kanten $b_4$ und $b_5$ einen Kreis:

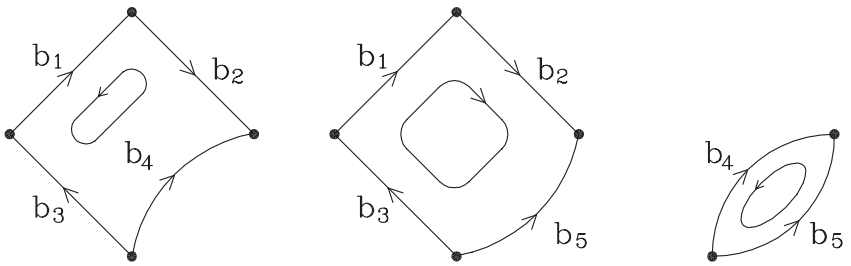



***Lösung zu Selbsttestaufgabe 4.2-1:***

Da Ecken und Bögen numeriert sind, kann die Inzidenzmatrix unmittelbar hingeschrieben werden:

$$I = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & -1 & -1 & -1 \end{pmatrix}$$

Das folgende Bild zeigt die drei Elemente von $C$, die sämtlich Kreise sind, jeweils mit einer Orientierung:




Damit erhält man die Kreis-Bogen-Matrix

$$C = \begin{pmatrix} -1 & -1 & -1 & 1 & 0 \\ 1 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

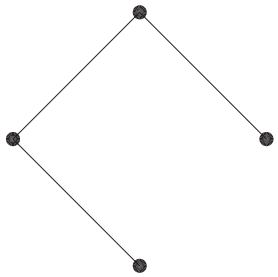
Schließlich rechnet man aus:

$$I \cdot C^t = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} -1 & 1 & 0 \\ -1 & 1 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

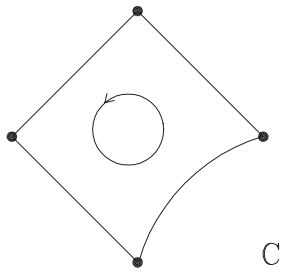
$$= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

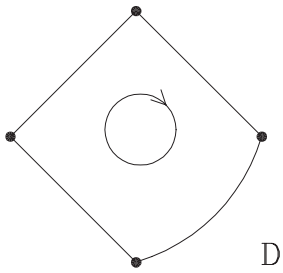
***Lösung zu Selbsttestaufgabe 5.2-1:***

Wir wählen das hier dargestellte Gerüst:

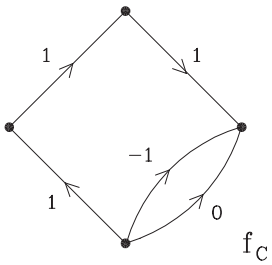


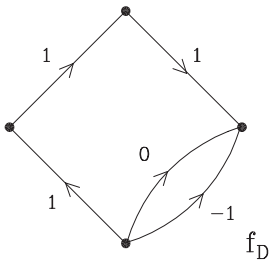
Dies führt zu den fundamentalen Kreisen $C$ und $D$ (mit einer beliebig gewählten Orientierung):




Somit bilden die Zirkulationen $f_C$ und $f_D$ eine Basis von $\mathbf{C}(\mathbb{R})$.




***Lösung zu Selbsttestaufgabe 6.1-1:***

Enthält ein Netz einen Kreis negativer Länge (im gerichteten Fall einen gerichteten Kreis bzw. Zyklus), und sind $a$ und $b$ auf diesem Kreis liegende Ecken, so gibt es von $a$ nach $b$ beliebig kurze Kantenfolgen, denn durch wiederholtes Durchlaufen des Kreises erhält man immer kürzere Kantenfolgen, ohne an eine untere Schranke zu geraten. Betrachtet man jedoch nur Wege von $a$ nach $b$, d. h. keine Ecke oder Kante darf mehrfach verwendet werden, so gibt es stets einen kürzesten Weg von $a$ nach $b$. Allerdings hat man bisher für diese Situation keinen guten Algorithmus zur Bestimmung der kürzesten Wege gefunden. Wenn das gegebene Netz keinen Kreis negativer Länge enthält, so gibt es stets kürzeste Kantenfolgen, und eine kürzeste Kantenfolge ist auch immer ein Weg (und folglich ein kürzester Weg).

***Lösung zu Selbsttestaufgabe 6.2-1:***

Die Algorithmen von Prim und Kruskal bestimmen jeweils ein minimales Gerüst für ein gegebenes Netz. Der Algorithmus von Prim geht folgendermaßen vor: Ausgehend von einer Kante mit kleinstem Gewicht wird iterativ jeweils eine Ecke mit einer Kante so hinzugenommen, dass diese Kante unter den zu neuen Ecken führenden Kanten minimales Gewicht hat; auf diese Weise ”wächst” ein Baum zu einem minimalen Gerüst. Beim Algorithmus von Kruskal müssen die Kanten zunächst ”der Größe nach” (bezüglich der gegebenen Bewertung) geordnet werden; dann wählt man jeweils die kleinste Kante und erklärt sie als zum aufzubauenden Gerüst gehörig, sofern sie mit den bereits gewählten Kanten keinen Kreis bildet. Beim Algorithmus von Kruskal können die Zwischenergebnisse Wälder sein, die sich erst zum Schluss zum Gerüst zusammenfügen.

***Lösung zu Selbsttestaufgabe 7.1-1:***

Ist $f$ ein Fluss auf einem Fluss-Netz $N = (G, c, q, s)$, so kann man den Digraphen $G = (E, K)$ durch Hinzunahme einer zusätzlichen Kante $k_{sq}$ zu einem Digraphen $G' = (E, K')$ erweitern und eine Kantenbewertung $f'$ auf $K'$ erklären durch

$$f'(k) := \begin{cases} f(k), & \text{falls } k \in K \\ w(f), & \text{falls } k \in k_{sq}\,; \end{cases}$$

$f'$ ist dann eine Zirkulation auf $G'$. Erweitert man auch die Abbildung $c$ zur Abbildung $c'$, indem man z. B. $c'(k_{sq}) := \sum\limits_{k \in K} |c(k)|$ setzt, so ergibt sich für den Digraphen $G'$ mit der Kapazitätsbeschränkung $c'$ für die Kanten, dass $f'$ eine zulässige Zirkulation ist. Sind umgekehrt ein Digraph $G = (E, K)$ mit einer Kapazitätsabbildung $c$ für die Kanten und darauf eine zulässige Zirkulation $f$ gegeben, und ist $k$ irgendeine Kante in $G$, so ergibt sich durch Herausnehmen von $k$ aus $G$ ein Fluss von $k^+$ nach $k^-$, genauer gesagt: $f/K\backslash\{k\}$ ist ein Fluss in dem Fluss-Netz $(G\backslash k,\ c/K\backslash\{k\}, k^+, k^-)$.

***Lösung zu Selbsttestaufgabe 7.2-1:***

Für jeden Schnitt $[Q, S]$ von $N$ muss gelten $q \in Q$ und $s \in S$. Es gibt die folgenden vier Möglichkeiten:

$$\begin{array}{lrl} Q = \{q\}, & S = \{a, b, s\} \text{ mit} & c(Q, S) = 6; \\ Q = \{q, a\}, & S = \{b, s\} \text{ mit} & c(Q, S) = 3; \\ Q = \{q, b\}, & S = \{a, s\} \text{ mit} & c(Q, S) = 10; \\ Q = \{q, a, b\}, & S = \{s\} \text{ mit} & c(Q, S) = 6. \end{array}$$

Offenbar ist der zweitgenannte Schnitt $[\{q, a\}, \{b, s\}]$ minimal, und nach dem "max-flow min-cut"-Theorem ist der maximale Fluss-Wert auf diesem Fluss-Netz $3$.

Die Betrachtung aller Schnitte, um den Wert eines maximalen Flusses zu ermitteln, führt nicht zu einem polynomialen Algorithmus, denn es gibt $2^{n-2}$ viele Schnitte eines Fluss-Netzes, wenn $n$ die Anzahl der Ecken ist.

***Lösung zu Selbsttestaufgabe 7.3-1:***

Der Algorithmus von Edmonds und Karp zur Bestimmung eines maximalen Flusses auf einem Fluss-Netz ist bei beliebigen reellen Kapazitätsbeschränkungen für die Kanten anwendbar, außerdem ist er polynomial (bezüglich der Ecken- bzw. Kantenanzahl des zugrundeliegenden Digraphen). Der Algorithmus von Ford und Fulkerson hat keine dieser Eigenschaften.

***Lösung zu Selbsttestaufgabe 7.4-1:***

Da die Ergebnisse des vorliegenden Kapitels sämlich für Digraphen formuliert sind, geht man von $G$ zunächst zu dem Digraphen $\tilde{G}$ über, indem man jede Kante von $G$ durch zwei entgegengesetzt gerichtete Kanten ersetzt (vgl. Beispiel 4.2-2 in Kapitel 4). Im Digraphen $\tilde{G}$ ordnet man nun jeder gerichteten Kante $k$ die Kapazität $c(k) = 1$ zu. Man kann nun für das Fluss-Netz $(\tilde{G}, c, A, B)$ mit dem Algorithmus von Edmonds und Karp einen maximalen $0$-$1$-Fluss ermitteln. Dessen Wert $w(f)$ stimmt mit der maximalen Anzahl kantendisjunkter Wege von $A$ nach $B$ in $\tilde{G}$ überein. Dabei muss es auch $w(f)$ viele kantendisjunkte Wege von $A$ nach $B$ in $\tilde{G}$ geben derart, dass für jede Kante $\{x, y\}$ von $G$ nur eine der gerichteten Kanten $xy$ bzw. $yx$ von irgendeinem dieser Wege genutzt wird. Es ist dann $w(f) - 1$ auch die maximale Anzahl von Kanten bzw. Leitungen, die in $G$ ausfallen dürfen, so dass immer noch ein Weg von $A$ nach $B$ intakt ist.

***Lösung zu Selbsttestaufgabe 7.5-1:***

Der vorgegebene bipartite Graph $G = (E_1 \cup E_2, K)$ wird durch Hinzunahme neuer Ecken $q$ und $s$ sowie gerichteter Kanten $qe$ (für $e \in E_1$) und $es$ (für $e \in E_2$) zu einem Digraphen $\tilde{G}$ erweitert; die alten Kanten aus $K$ werden von $E_1$ nach $E_2$ hin gerichtet. In $\tilde{G}$ wird jede Kante $\tilde{k}$ mit $c(\tilde{k}) = 1$ bewertet. Nun wird mit dem Algorithmus von Ford und Fulkerson ein maximaler ($0$-$1$-) Fluss $f_{max}$ bestimmt. Die Kanten $k \in K$ mit $f_{max}(k) = 1$ bilden dann eine maximale Korrespondenz in $G$.

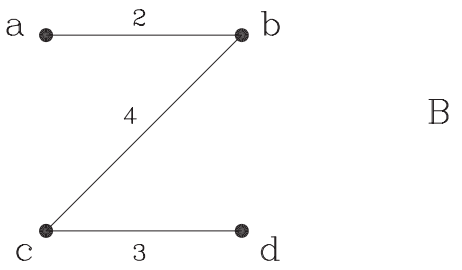
***Lösung zu Selbsttestaufgabe 7.6-1:***

Hat man für ein Fluss-Netz $(G, c, q, s)$ zusätzlich eine untere Kapazität $b : K \to \mathbb{R}_0^+$ gegeben, die von den Flüssen zu respektieren ist, so braucht es keinen zulässigen Fluss von $q$ nach $s$ zu geben, also auch keinen maximalen Fluss. Existiert jedoch irgendein zulässiger Fluss, so gibt es auch einen maximalen.

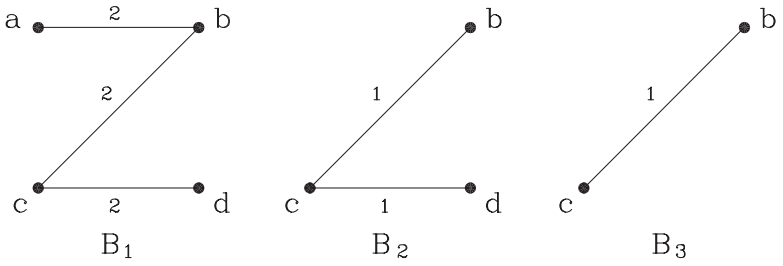
Zur effektiven Bestimmung eines maximalen Flusses muss zunächst irgendein zulässiger Fluss gefunden werden, dann wird dieser mit Hilfe zunehmender Wege zu einem maximalen Fluss erweitert. Beim ersten Problem kann man sich des Algorithmus von Ford und Fulkerson bedienen, der auf ein geeignet abgewandeltes Netz anzuwenden ist; für das zweite Problem kann eine Erweiterung des Algorithmus von Ford und Fulkerson verwendet werden, in der auch die unteren Kapazitätsschranken berücksichtigt werden.

***Lösung zu Selbsttestaufgabe 7.7-1:***

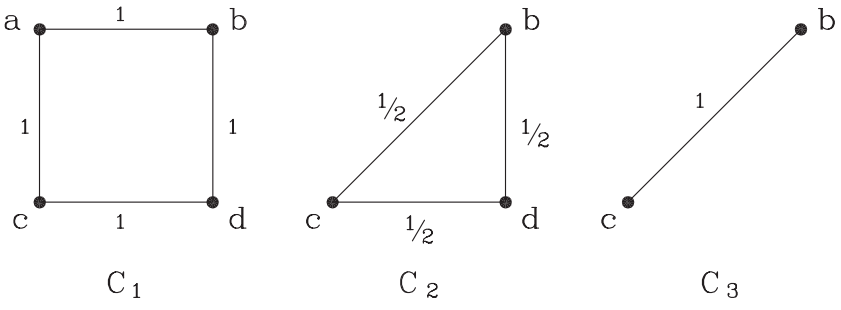
Es wird die bei der Besprechung des Beispiel 7.7-2 beschriebene Konstruktion auf das vorliegende Beispiel angewandt. Bei diesem Beispiel gibt es einen eindeutigen maximalen Baum:



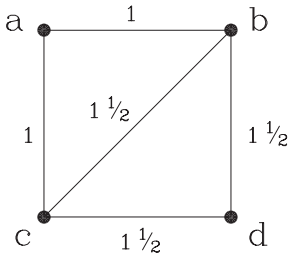

Die folgenden Diagramme zeigen die Zerlegung von $B$ in uniforme Bäume:




Man kann so zu folgenden Kreisen kommen. (Bei $C_1$ könnte auch irgendein anderer einfacher Kreis mit den Ecken $a, b, c, d$ gebildet werden.)




Als minimales (zulässiges) Netz erhält man also:




(G,c)

***Lösung zu Selbsttestaufgabe 8.1-1:***

Zunächst mag es mehrere kürzeste Wege gleicher Länge geben, so dass stets eine Auswahl zu treffen ist. Im Fall von Änderungen des Netzzustandes (z. B. Ausfall einer Leitung) können sich für bestimmte Knotenpaare die kürzesten Wege zwischen ihnen ändern; es muss festgelegt werden, welche Instanz die neuen kürzesten Wege ermittelt (zentral oder verteilt), wie die sich daraus ergebenden Routing-Informationen im Netz weitergegeben werden usw. In dem Fall, dass Informationen über Zweitwege (und Drittwege usw.) bereits von vornherein in den Knoten gespeichert sind, müssen diese "statischen" Daten zu Beginn des Netzbetriebs sorgfältig so bestimmt werden, dass in keiner Situation Nachrichten im Netz kreisen können und eine Reihe weiterer Kriterien erfüllt sind.

***Lösung zu Selbsttestaufgabe 8.2-1:***

Durch die verteilte Ausführung des Algorithmus berechnet jeder Netzknoten selbst die für ihn aktuell günstigsten Wege, es muss dies kein Knoten zentral für alle anderen tun und anschließend die Informationen verteilen. Auch müssen die einzelnen Netzknoten keine Kenntnis der gesamten Netztopologie haben. Der entscheidende Vorteil des asynchronen Algorithmus ist, dass auf eine Synchronität der in den verschiedenen Netzknoten ausgeführten Iterationsschritte verzichtet werden kann. Jeder Knoten muss nur immer wieder aufgrund der neuesten ihm vorliegenden Informationen seine kürzesten Abstände zu den anderen Knoten berechnen und die Ergebnisse seinen Nachbarn mitteilen. Dies im ganzen Netz synchron zu realisieren, würde hingegen einen hohen zusätzlichen Aufwand bedeuten.

***Lösung zu Selbsttestaufgabe 8.3-1:***

Oszillationen beim Routing gehen einher mit einer ungleichmäßigen Auslastung von verschiedenen Routen und so auch mit einer erhöhten Gefahr von Überlastsituationen in Teilen des Netzes, was letztlich auf Kosten des Durchsatzes geht. Häufiges Umlenken von Paketen wird auch die Paketlaufzeiten erhöhen, außerdem werden durch den dazu nötigen Koordinationsaufwand Teile der Ressourcen unnötig verbraucht.

***Lösung zu Selbsttestaufgabe 8.4-1:***

B ignoriert die von A erhaltene Nachricht mit Folgenummer 2. Dies hat jedoch keine ernsten Konsequenzen, wenn C die Nachricht von A mit Folgenummer 5 korrekt empfängt, denn C versorgt danach u. a. auch B mit der korrekten Nachricht mit Folgenummer 5.

***Lösung zu Selbsttestaufgabe 8.5-1:***

Obwohl das Routing im Internet sicherstellt, dass im Normalfall die Pakete auf kürzesten Wegen zu ihren Zielen laufen, ist z. B. folgender Fall möglich: aufgrund des Ausfalls einer Kante wurde die Entscheidung getroffen, ein Paket umzuleiten, jedoch ist die defekte Kante unmittelbar danach wiederhergestellt, so dass rein physikalisch der optimale Weg wieder verfügbar wäre.

***Lösung zu Selbsttestaufgabe 8.6-1:***

Man kann sich z. B. des Algorithmus von Dijkstra bedienen, der mit einer Komplexität von O($n^2$) die kürzesten Abstände von einem Knoten zu allen anderen bestimmen kann.

Es sei zunächst eine Ecke $j$ des Graphen fest gewählt. Mit einmaliger Anwendung des Algorithmus von Dijktra können alle Nachbarn $c(1, j, x)$ für $x \neq j$ bestimmt werden. Durch Streichen jeder einzelnen an $j$ anliegenden Kante und jeweilige anschließende Anwendung des Algorithmus von Dijkstra kann man all $c(2, j, x)$ mit $x \neq j$ bestimmen.

Da $j$ bis zu $n-1$ viele Nachbarn haben kann, ist auf diese Weise bei festem $j$ ein Aufwand von O($n^3$) nötig, mithin hat man für den gesamten Algorithmus zur Erstellung des Routing-Plans eine Komplexität von O($n^4$).

***Lösung zu Selbsttestaufgabe 8.7-1:***

Die Schwierigkeiten beginnen schon damit, dass es durchaus nicht klar ist, wie genau ”optimales Routing” definiert werden sollte. Doch selbst wenn man sich auf eine Definition einlässt, indem man verschiedene Ziele des Routing in einer zu optimierenden Zielfunktion zusammenzufassen versucht – dies ist stets nur unter vielen vereinfachenden Annahmen möglich, so stößt man dann in der Regel auf ein sehr schwierig zu lösendes mathematisches Optimierungsproblem.

***Lösung zu Selbsttestaufgabe 9.2-1:***

Eine Möglichkeit wäre es, alle Koeffizienten $g(5,b)$ mit $4 \leq b \leq 10$ gemäß der Vorgehensweise in Beispiel 9.2-5 mit einem linearen Gleichungssystem zu bestimmen. Die Frage nach $g(5,4)$ lässt sich jedoch schneller beantworten: da $g(5,4)$ die Anzahl der Gerüste des $K_5$ angibt, gilt nach den Formeln in Kapitel 6$g(5,4) = 5^3 = 125$.

***Lösung zu Selbsttestaufgabe 9.3-1:***

$G$ besitze $n$ viele Knoten. Wird mit $F_i(G)$ bzw. $F_i(G')$ die jeweilige Anzahl funktionsfähiger Zustände mit $i$ vielen Kanten bezeichnet , so gilt offenbar $F_i(G') \geq F_i(G)$ für alle $i$, denn ein fuktionsfähiger Zustand in $G$ ist dies stets auch in $G'$.

Dies zeigt $z_v(G') \geq z_v(G)$.

Es gilt sogar $z_v(G') > z_v(G)$, denn es ist .z. B. $F_{n-1}(G') > F_{n-1}(G)$ : ein beliebiges Gerüst von $G'$, welches die "neue" Kante erhält, ist nämlich funktionsfähig in $G'$, jedoch gibt es keine Entsprechung in $G$.

***Lösung zu Selbsttestaufgabe 9.4-1:***

Für ein Zuverlässigkeitsproblem mit der Menge $\mathcal{F} \subseteq \mathcal{P}(K)$ funktionsfähiger Zustände hat das Zuverlässigkeitspolynom als Funktion der Kantenwahrscheinlichkeit stets die Form

$$z(G) = \sum_{i=0}^{m} F_i \cdot p^i \, (1-p)^{m-i} \, ,$$

wobei $F_i$ die Anzahl funktionsfähiger Zustände mit $i$ vielen Kanten ist. Gelingt es, diese Koeffizienten effizient zu bestimmen, so ist es gerechtfertigt, das Zuverlässigkeitsproblem als gelöst zu betrachten. Hat man umgekehrt ein Verfahren, zu beliebigen $p_0$ den Wert $z(G)\,(p_0)$ effizient zu errechnen, so können damit auch leicht die Koeffizienten $F_i$ bestimmt werden.

***Lösung zu Selbsttestaufgabe 9.5-1:***

Um die Sperner-Schranken anwenden zu können, muss die Menge $\mathcal{F}$ der funktionsfähigen Zustände folgende Eigenschaft haben: jede Obermenge eines Elements von $\mathcal{F}$ gehört wieder zu $\mathcal{F}$ - anders gesagt: das Hinzufügen von Kanten zu einem funktionsfähigen Zustand kann nie zu Nicht-Funktionsfähigkeit führen. Äquivalent dazu ist auch die folgende Aussage: die Komplemente der Elemente von $\mathcal{F}$ (als Teilmengen der Kantenmenge $K$) bilden ein erbliches Mengensystem.

***Lösung zu Selbsttestaufgabe 9.6-1:***

In Man hat zunächst die Beziehung

$$z(\mathcal{N})(p) - \check{z}(\mathcal{N})(p) \leq \binom{C_2}{2} q^3.$$

Nun ist $C_2 \leq \binom{m}{2}$ und $m \leq \binom{n}{2}$, also $C_2 \leq \binom{\binom{n}{2}}{2} < \frac{n^4}{8}$ und $\binom{C_2}{2} < \frac{n^8}{128}$.

Damit folgt

$z(\mathcal{N})(p) - \check{z}(\mathcal{N})(p) < \frac{n^8}{128} \cdot 128 \cdot n^{-8} \cdot 10^{-t} = 10^{-t}$.

***Lösung zu Selbsttestaufgabe 9.7-1:***

Ist die Kantenausfallwahrscheinlichkeit $q = 1 - p$ sehr klein (z. B. ist $10^{-4}$ oder kleiner nicht unrealistisch), so wird in der Näherung

$$z(G)(p) \approx 1 - C_c \cdot q^c \cdot p^{m-c}$$

bei nicht zu großen Graphen (und damit auch beschränktem $C_c$) die Größe von $C_c \cdot q^c \cdot p^{m-c}$ vor allem von der Größe von $c$ bestimmt. Dies rechtfertigt das Ziel, ein möglichst großes $c$ anzustreben und dann für dieses $c$ die Zahl $C_c$ zu minimieren.

***Lösung zu Selbsttestaufgabe 10.2-1:***

Wegen $z = p(n-1)$ muss $p = \frac{1000}{6{,}5 \cdot 10^9} \approx 1,5 \cdot 10^{-7}$ gewählt werden. Also folgt für den erwarteten mittleren Abstand $l \approx \frac{\log 6{,}5 \cdot 10^9}{1000} \approx \frac{9{,}81}{3} = 3,27$ und für den erwarteten Cluster-Koeffizienten $C \approx 1,5 \cdot 10^{-7}$.

***Lösung zu Selbsttestaufgabe 10.3-1:***

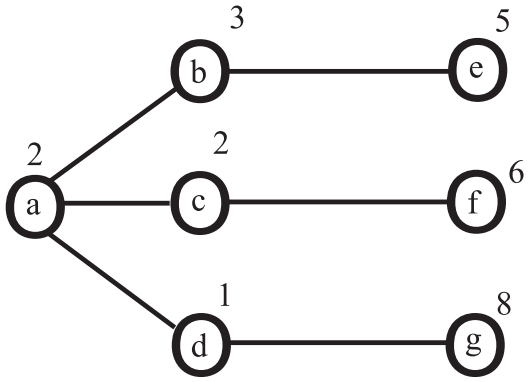
Für wachsendes $n$ bleibt bei festem $z$ der Cluster-Koeffizient in den Zufallskreisen nach Watts-Strogatz recht hoch – ausgehend von $\frac{3}{4} \cdot \frac{z-2}{z-1}$ bei $K_{n,z}$ erfolgt eine Abnahme lediglich mit dem Faktor $(1-p)^3$. Demgegenüber beträgt der Cluster-Koeffizient in $\mathfrak{G}(n,p)$ nur $p$ und nimmt umgekehrt proportional zu $n$ direkt ab.

Der erwartete mittlere Abstand kommt bei den Zufallsgraphen nach Watts-Strogatz schon bei kleinem $p$ in die Nähe von $\log n$, dem Wert für die Zufallsgraphen der Klasse $\mathfrak{G}(n,p)$.

***Lösung zu Selbsttestaufgabe 10.4-1:***

Ein Greedy-Algorithmus wählt stets die hinsichlich einer Zielsetzung aktuell am günstigsten erscheinende Alternative. Bei dem folgenden Problem führt dies nicht zum Ziel.

Der Graph $G = (E, K)$ mit einer Knotenbewertung wie im untenstehenden Diagramm sei gegeben. Knoten $a$ ist aktuell markiert. Durch Markierung eines bisher nicht markierten Nachbarn möchte man schrittweise auf einen Knoten mit möglichst großer Markierung stoßen. Wählt man nun im ersten Schritt von $a$ ausgehend den Nachbarn $b$ (das wäre "greedy"), so wird man den wahren Gewinner $g$ niemals finden.




***Lösung zu Selbsttestaufgabe 10.5-1:***

Angenommen, es existierte eine Kante von $o \in OUT$ nach $i \in IN$. Da man ohnehin (durch gerichtete Wege) von $i$ zu jedem Knoten von $SCC$ und von jedem Knoten von $SCC$ zu $o$ kommt, käme man dann auch von $o$ zu jedem Knoten von $SCC$ und von jedem Knoten von $SCC$ zu $i$, d. h. es wären $i, o \in SCC$.

***Lösung zu Selbsttestaufgabe 10.6-1:***

Nach $t$ vielen Schritten hat ein nach dem Modell von Barabasi, Albert und Jeong erzeugter zufälliger Graph $m_0 + t$ viele Ecken und $mt$ viele Kanten. Da jede Kante den Eckengrad zweier Ecken um Eins erhöht, führt dies zu einem durchschnittlichen Eckengrad von

$$2\frac{mt}{m_0 + t}$$

was sich für $t \to \infty$ dem Wert $2m$ nähert.

***Lösung zu Selbsttestaufgabe 10.7-1:***

In einem endlichen Graphen mit $n$ vielen Ecken kann der Grad einer Ecke höchstens $n-1$ betragen, d. h. ab $i \geq n$ gilt für den Anteil der Ecken vom Grad $i$ stets $p_i = 0$. Falls ein Potenzgesetz für die Eckengrade exakt gilt, nimmt $p_i$ jedoch niemals den Wert Null an.

***Lösung zu Selbsttestaufgabe 11.2-1:***

Die Grundidee der Methode der Relaxationen besteht darin, die für die gesuchten $p(x)$-Werte geltenden Gleichungen (Jeder Wert ist das arithmetische Mittel der zu den Nachbarknoten gehörenden Werte) als Iterationsanweisungen aufzufassen, wobei die Randwerte als Startwerte genommen werden und der $p(x)$-Wert eines inneren Punktes $x$ mit Null beginnt.

Dass dieses Verfahren stets konvergiert, kann mit den Methoden der Numerischen Mathematik gezeigt werden.

***Lösung zu Selbsttestaufgabe 11.3-1:***

Gegeben sei ein einfacher zusammenhängender Graph $G$, $s$ und $t$ seien Ecken. Der effektive Widerstand $R_{eff}(s,t)$ ergibt sich beispielsweise als Kehrwert von $i_s$, wobei $i_s$ den Strom bezeichnet, der in die Ecke $s$ hinein fließt, wenn bei $s$ die Spannung $v_s = 1$ und bei $tv_t = 0$ gesetzt wird.

Eine andere Formulierung lautet: $R_{eff}(s,t)$ ist die Potentialdifferenz zwischen $s$ und $t$, wenn in $s$ von außen ein Strom $i_s = 1$ hinein und bei $t$ ein Strom $i_t = 1$ nach außen abfließt. (Bei der Verwendung der physikalischen Begriffe ist immer vorausgesetzt, dass die Kirchhoffschen Gesetze gelten.)

***Lösung zu Selbsttestaufgabe 11.4-1:***

Aus dem Rayleighschen Monotoniegesetz folgt:

- Lässt man in einem unendlichen rekurrenten Graphen eine Kante weg, so ist auch der entstehende Graph rekurrent.
- Fügt man in einem unendlichen transienten Graphen eine zusätzliche Kante hinzu, so ist auch der entstehende Graph transient.

Man kann mit anderen Worten unter Umständen durch Weglassen oder Hinzufügen von Kanten auf bereits untersuchte Graphen treffen und so "rekurrent" bzw. "transient" zeigen.

# Übungsaufgaben

### Übungsaufgaben zum Kapitel "Grundbegriffe"

**Aufgabe 1**: Beweisen Sie Satz 1.2-5. **4 Pkt**

**Aufgabe 2**: Ein Graph $G$ mit $n$ Ecken und $m$ Kanten sei gegeben. **5 Pkt**
Zeigen Sie: sind irgendwelche zwei der drei folgenden Eigenschaften erfüllt, so ist $G$ ein Baum, und es ist auch die dritte Eigenschaft erfüllt.

i. $n - 1 = m$.

ii. $G$ enthält keinen Kreis.

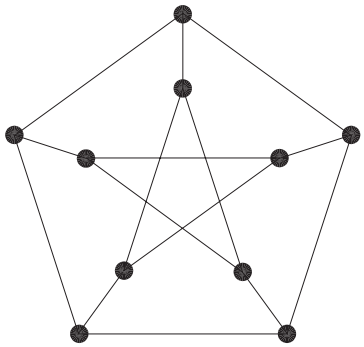
iii. $G$ ist zusammenhängend.

**Aufgabe 3**: Es sei $G$ ein Graph mit $n$ Ecken, für jede Ecke $e$ gelte $\gamma(e) \geq \frac{n-1}{2}$. **5 Pkt**
Zeigen Sie: $G$ ist zusammenhängend.

**Aufgabe 4**: Man zeige: Ist $M$ ein Multigraph und $B$ ein spannender Wald in $M$, so **5 Pkt**
bilden die $\rho(M)$ vielen fundamentalen Schnitte eine Basis von $\mathbf{S}^*$ (s.Satz 1.3-4).

### Übungsaufgaben zum Kapitel "Darstellung von Graphen"

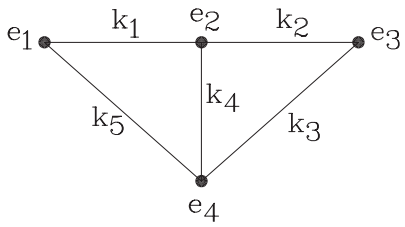
**Aufgabe 5**: Weisen Sie nach, dass der im folgenden Diagramm dargestellte Graph, **5 Pkt**
der sogenannte Petersen-Graph, nicht planar ist.



**4 Pkt** **Aufgabe 6**: Beweisen Sie die im Text aufgestellte Behauptung, dass für die Inzidenzmatrix $I$ eines Multigraphen $M$ stets gilt $rg_2(I) = \rho(M)$.

**5 Pkt** **Aufgabe 7**:

i. Stellen Sie die Kreis-Kanten-Matrix $C(G)$ des im folgenden Diagramm dargestellten Graphen $G$ auf.




ii. Stellen Sie auch die Inzidenzmatrix $I(G)$ auf, und rechnen Sie die Beziehung $I \cdot C^t = 0$ (über $\mathsf{IF}_2$) nach.

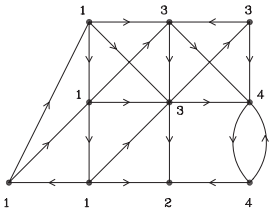
**Übungsaufgaben zum Kapitel "Algorithmen"**

**4 Pkt** **Aufgabe 8**: Entwerfen Sie einen Algorithmus von Komplexität $O(n)$, der aus $n$ gegebenen reellen Zahlen $a_1, \ldots, a_n$ die kleinste heraussucht.
(Begründen Sie die Aussage über die Komplexität.)

**5 Pkt** **Aufgabe 9**: Für einen Graphen $G = (E, K)$ bezeichnet man als Komplementgraphen $\overline{G} = (E, \overline{K})$ den Graphen mit derselben Eckenmenge, in dem genau die in $G$ nicht verbundenen Ecken benachbart sind, d. h. für $e, f \in E$ gilt $\{e, f\} \in \overline{K}$ genau dann, wenn $\{e, f\} \notin K$ ist.
Zeigen Sie: Eine Teilmenge $X \subseteq E$ ist genau dann eine Knotenüberdeckung in $G$, wenn $E \backslash X$ in $\overline{G}$ einen vollständigen Untergraphen bildet.

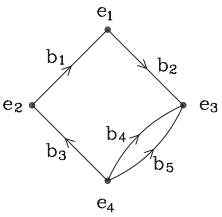
**Übungsaufgaben zum Kapitel "Pseudodigraphen"**

**Aufgabe 10**: Bestimmen Sie für den im folgenden Diagramm dargestellten Digraphen die starken Zusammenhangskomponenten: **3 Pkt**




**Übungsaufgaben zum Kapitel "Bewertungen"**

**Aufgabe 11**: Stellen Sie für den Multidigraphen $\vec{M}$ (vgl. Selbsttestaufgaben) eine Schnitt-Bogen-Matrix auf, und geben Sie eine Basis des Vektorraums $\mathbf{S}(\mathsf{IR})$ an. **6 Pkt**

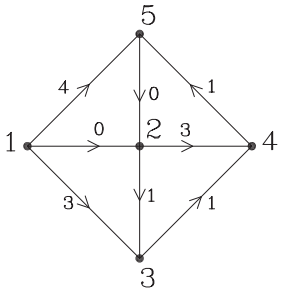



$\vec{M}$

**Aufgabe 12**: Der Digraph $\vec{G} = (E, B, v)$ sei ein Baum, $W$ sei ein beliebiger Körper. Zeigen Sie: Die Nullabbildung $f : B \to W$ (d. h. $f(b) = 0$ für alle $b \in B$) ist die einzige Zirkulation auf $\vec{G}$. **5 Pkt**

**Übungsaufgaben zum Kapitel "Kürzeste Wege und minimale Gerüste"**

**4 Pkt** **Aufgabe 13**: Bestimmen Sie für das im folgenden Diagramm dargestellte Netz $(G, w)$ die Abstände zwischen je zwei Ecken mit dem Algorithmus von Floyd-Warshall.




**5 Pkt** **Aufgabe 14**: Es sei $(G, w)$ ein Netz ohne negative Kreise, zu jeder Ecke $e \in E$ sei $N(e)$ die Menge aller Nachbar- bzw. Nachfolgerecken. Zeigen Sie, dass der folgende Algorithmus von Bellmann-Ford die Abstände von der Ecke $s$ mit Komplexität $0(|E|^3)$ berechnet:

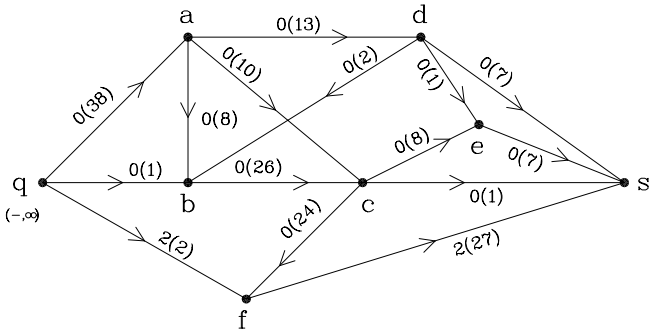
**Algorithmus**:

```
begin
   Setze d(s) := 0;
   for e ∈ N(s) do setze d(e) := w(se);
   for e ∈ E \ (N(s) ∪ { s} ) do setze d(e) := ∞;
   repeat
      for e ∈ E do setze d'(e) := d(e),
      for e ∈ E do setze
         d(e) := min {d'(e), min { d'(f) + w(fe) | fe ∈ K } }
   until d(e) =d'(e) für alle e ∈ E;
end
```

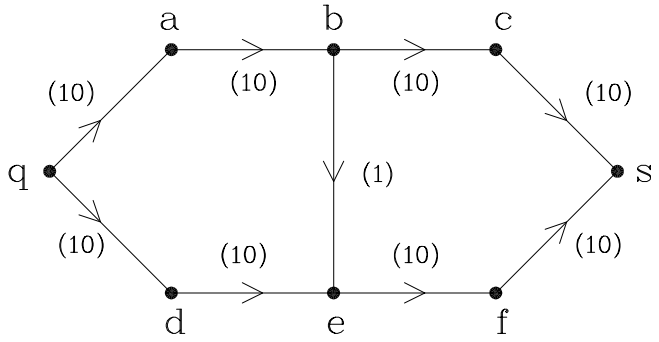
**5 Pkt** **Aufgabe 15**: Es sei $(G, w)$ ein Netz, in dem je zwei Kanten verschiedene Bewertungen haben. Zeigen Sie, dass $(G, w)$ genau ein minimales Gerüst besitzt.

**Übungsaufgaben zum Kapitel "Flüsse"**

**Aufgabe 16**: Bestimmen Sie einen maximalen Fluss für das im folgenden Diagramm dargestellte Fluss-Netz **5 Pkt**




**Aufgabe 17**: **6 Pkt**




a. Bestimmen Sie für das dargestellte Fluss-Netz einen maximalen Fluss mit dem Algorithmus von Ford und Fulkerson.

b. Verwenden Sie das Netz und die Ergebnisse aus a), um zu begründen, dass der Algorithmus von Ford und Fulkerson selbst bei ganzzahligen Kapazitäten nicht polynomial ist.

**Aufgabe 18**: Die Zusammenhangszahl $\kappa(G)$ eines Graphen $G = (E, K)$ ist wie folgt erklärt: Ist $G$ ein vollständiger Graph $K_n$, so gilt $\kappa(G) = n - 1$; andernfalls ist **5 Pkt**

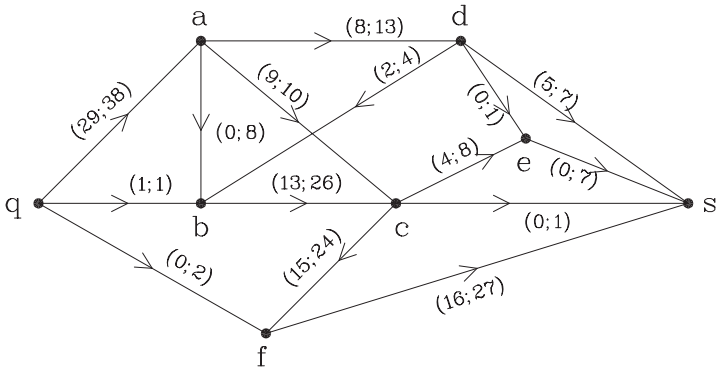
$$\kappa(G) := min\{|F| \,/ F \subseteq E,\ G \backslash F \ \text{ ist nicht zusammenhängend }\}.$$

$G$ heißt $p$-fach zusammenhängend, wenn $\kappa(G) \geq p$ gilt.

Zeigen Sie: Ein Graph $G$ ist genau dann $p$-fach zusammenhängend, wenn je zwei Ecken in $G$ durch mindestens $p$ eckendisjunkte Wege verbunden sind.

**5 Pkt** **Aufgabe 19**: Ein (ungerichteter) Graph $G$ heißt regulär, wenn alle Ecken denselben Grad $\gamma$ mit $\gamma \neq 0$ haben, d. h. jede Ecke hat $\gamma$ viele Nachbarecken. Zeigen Sie: In einem regulären bipartiten Graphen gibt es stets eine vollständige Korrespondenz.

**5 Pkt** **Aufgabe 20**: Es sei der folgende Digraph $G$ mit Kapazitäten $b$ und $c$ gegeben:

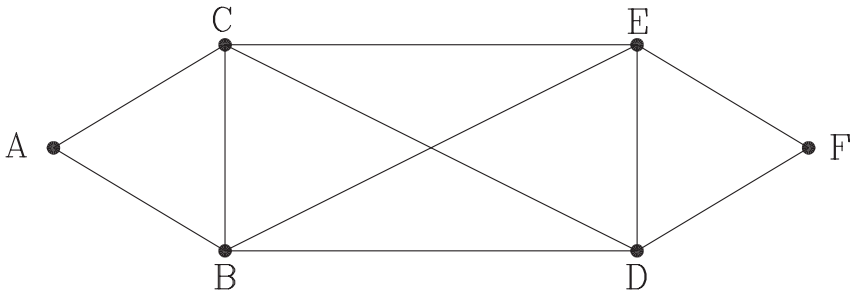



Bestimmen Sie einen maximalen zulässigen Fluss von $q$ nach $s$. Begründen Sie die Maximalität des gefundenen Flusses.

**6 Pkt** **Aufgabe 21**: Gegeben sei eine $(n \times n)$-Matrix $A$ mit nicht-negativen reellen Einträgen. Als das Zuordnungsproblem ("assignment problem") bezeichnet man das Problem, aus jeder Zeile und jeder Spalte ein Element so auszuwählen, dass die Summe der ausgewählten $n$ Einträge minimal ist. Definieren Sie ein geeignetes Fluss-Netz mit einer Kostenfunktion derart, dass ein gewisser Fluss mit minimalen Kosten eine Lösung des Zuordnungsproblems liefert.

**Übungsaufgaben zum Kapitel "Wegauswahl in Netzen"**

**Aufgabe 22**: In dem folgenden Netz werde mit Random Routing gearbeitet: **4 Pkt**



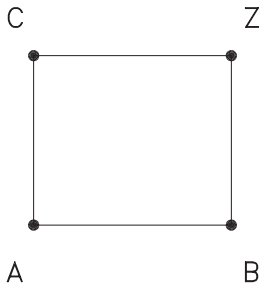

Angenommen, A möchte eine Nachricht an F schicken. Es wird von folgenden weiteren Annahmen ausgegangen:

1. Die übertragungszeit eines Nachrichtenpakets auf einer Kante beträgt stets 1 msec.
2. Die Verarbeitungszeit in einem Zwischenknoten, der das Paket weiterleitet, beträgt stets 4 msec.

Wie groß ist die Wahrscheinlichkeit, dass die Nachricht nach 11 msec beim Empfänger F ankommt?

**Aufgabe 23**: Es wird wieder das in Aufgabe 8.1 dargestellte Netz betrachtet. Angenommen, B möchte per Flooding (mit Schwellwert 2) eine Nachricht an alle anderen Knoten schicken. (Die Nachrichtenpakete brauchen also bei dieser Anwendung keine Zieladresse zu tragen.) Weiter sei angenommen, dass die Kanten BD und DE ausfallen, nachdem B seine ersten vier Meldungen erfolgreich verschickt hat. **3 Pkt**

Bestimmen Sie die Anzahl der Meldungen, die unter diesen Bedingungen durch die Nachricht im Netz entstehen.




**5 Pkt** **Aufgabe 24**:

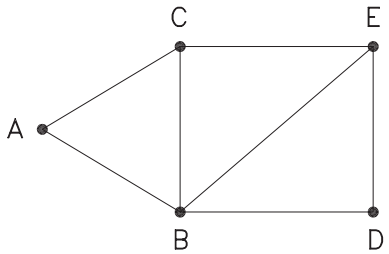
In dem hier dargestellten Paketnetz werde folgende Situation betrachtet:

- Jeder der Knoten A,B,C erzeugt pro Zeittakt T eine Datenmenge gleicher Größe für Ziel Z.

  1. Am Ende eines Zeittaktes geschieht folgendes:
  2. Die Kantenlängen $w_{ij}$ werden nun festgelegt entsprechend der im letzten Zeittakt getragenen Verkehrslast. (I. a. ist $w_{ij} \neq w_{ji}$
  3. Die Knoten informieren sich gegenseitig über die neuen Kantenlägen.
  4. Die Knoten bestimmen ihren kürzesten Weg zum Ziel Z.

- In jedem Zeittakt wird aufgrund der zuletzt bestimmten kürzesten Wege geroutet.

  1. Beschreiben Sie, wie sich das Routing mit den Zeittakten entwickelt, wenn zu Beginn des 1. Zeittaktes $w_{ij} = 1$ für alle Kanten angenommen wird.
  2. Was ändert sich, wenn ein Biasfaktor von $\alpha = 1$ eingeführt wird?
  3. Durch welche zusätzliche Maßnahme könnte man Oszillationen entgegenwirken?



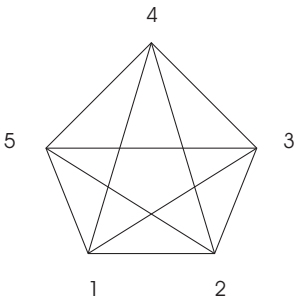

**3 Pkt** **Aufgabe 25**:

Es sei das obige Netz gegeben. Betrachtet wird die folgende Situation: A möchte eine Nachricht per Broadcasting verschicken, wobei Flooding mit Folgenummern angewendet wird.

Bestimmen Sie die Anzahl der Meldungen, die im Netz ausgetauscht werden. Zeigen Sie beispielhaft auf, dass diese Anzahl auch von der zeitlichen Reihenfolge gewisser Ereignisse abhängt.

**Aufgabe 26**: Zeigen Sie: Ist $G$ ein Graph und $C$ ein kreisfreier Routing-Plan für $G$, so kann jeder Knoten jeden anderen erreichen, d. h. in jedem $G(\rightarrow k)$ existiert ein gerichteter Weg von jedem Knoten $x \neq k$ nach $k$. **5 Pkt**

**Aufgabe 27**: Auf dem Graphen $G = K_5$ sei der dargestellte Routing-Plan $C$ mit $\pi = 2$ gegeben. **5 Pkt**

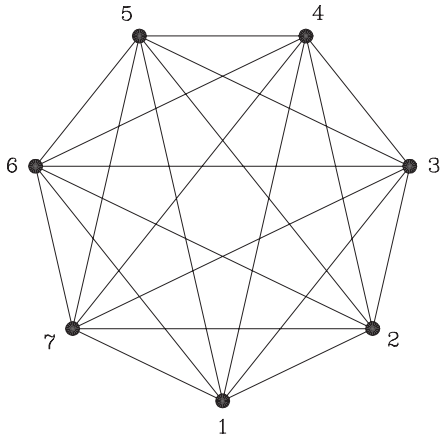



| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | - | 2;5 | 3;2 | 4;5 | 5;2 |
| 2 | 1;3 | - | 3;1 | 4;3 | 5;1 |
| 3 | 1;2 | 2;4 | - | 4;2 | 5;4 |
| 4 | 1;5 | 2;3 | 3;5 | - | 5;3 |
| 5 | 1;4 | 2;1 | 3;4 | 4;1 | - |

$C$

Ist $C$ kreisfrei bzw. bidirektional? Stellen Sie fest, ob stets ein möglichst kurzer Weg zum Ziel gewählt wird.

**7 Pkt** **Aufgabe 28**: Auf dem Graphen $G = K_7$ sei der dargestellte Routing-Plan $C$ mit $\pi = 2$ gegeben.




C

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | – | 2;3 | 3;2 | 4;7 | 5;6 | 6;5 | 7;4 |
| 2 | 1;3 | – | 3;1 | 4;6 | 5;7 | 6;4 | 7;5 |
| 3 | 1;2 | 2;1 | – | 4;5 | 5;4 | 6;7 | 7;6 |
| 4 | 1;7 | 2;6 | 3;5 | – | 5;3 | 6;2 | 7;1 |
| 5 | 1;6 | 2;7 | 3;4 | 4;3 | – | 6;1 | 7;2 |
| 6 | 1;5 | 2;4 | 3;7 | 4;2 | 5;1 | – | 7;3 |
| 7 | 1;4 | 2;5 | 3;6 | 4;1 | 5;2 | 6;3 | – |

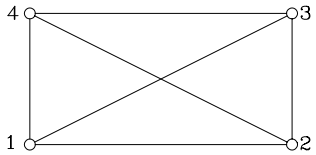
C

Ist $C$ kreisfrei bzw. bidirektional? Stellen Sie fest, ob stets ein möglichst kurzer Weg zum Ziel gewählt wird.

**Übungsaufgaben zum Kapitel "Zuverlässigkeit in Netzen"**

**4 Pkt** **Aufgabe 29**: Gegeben sei der vollständige Graph $G = K_5$ mit 5 Ecken. Jede Kante von $G$ habe eine Wahrscheinlichkeit von $p = \frac{1}{2}$ für ihre Funktionsfähigkeit. Bestimmen Sie die Wahrscheinlichkeit, dass $G$ zusammenhängend ist.

**Aufgabe 30**: Auf dem Graphen $G = K_4$ sei der folgende kreisfreie Routing-Plan $C$ mit den jeweils zwei Einträgen gegeben: **6 Pkt**



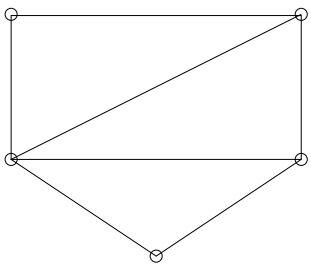

G

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | - | 2;3 | 3;2 | 4;2 |
| 2 | 1;3 | - | 3;1 | 4;1 |
| 3 | 1;2 | 2;1 | - | 4;1 |
| 4 | 1;2 | 2;1 | 3;1 | - |

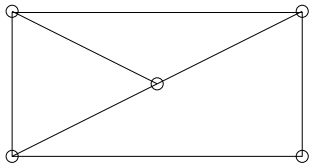
$C$

Die funktionsfähigen Zustände seien dadurch gekennzeichnet, dass jeder Knoten an jeden anderen Nachrichten versenden kann. Bestimmen sie das Zuverlässigkeitspolynom.

**Aufgabe 31**: Die beiden folgenden Graphen $G_1$ und $G_2$ mit jeweils 5 Ecken und 7 Kanten seien gegeben. Vergleichen Sie $G_1$ und $G_2$ hinsichtlich ihrer Zuverlässigkeit, wenn die Funktionsfähigkeit durch den Zusammenhang der Graphen gegeben ist. **6 Pkt**



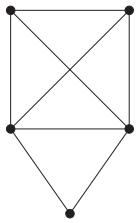
G 1



G 2

**Aufgabe 32**: Man bestimme mit der in Anhang G beschriebenen Determinantenmethode die Anzahl der Gerüste des vollständigen Graphen $K_6$ mit 6 Ecken und 15 Kanten. **5 Pkt**

**6 Pkt** **Aufgabe 33**: Gegeben sei der folgende Graph $G$:



Ermitteln Sie die einfachen und die Spernerschen Schranken für das Zuverlässigkeitspolynom.

**6 Pkt** **Aufgabe 34**: Für ein Mengensystem $\mathcal{M} \subseteq \mathcal{P}(\mathcal{M})$ über eine Menge $M$ bezeichnen wir eine Menge $T \subseteq M$ als **Transversale**, wenn $T$ mit jeder Menge aus $\mathcal{M}$ höchstens ein Element gemeinsam hat. Das **Problem der maximalen Transversalen** besteht darin, zu gegebenem $\mathcal{M}$ eine Transversale mit möglichst vielen Elementen zu finden. Zeigen Sie: Das Problem der minimalen Knotenüberdeckung ist auf das Problem der maximalen Transversalen reduzierbar.

**5 Pkt** **Aufgabe 35**: Der Durchmesser $diam(G)$ eines Graphen $G = (E, K)$ ist der maximale Abstand $\max\{|a, b| \big| a, b \in E\}$ zwischen zwei Ecken von G.

Gegeben sei der vollständige Graph $K_7$ mit 7 Ecken und 21 Kanten.

a. Bestimmen Sie ein Gerüst mit möglichst kleinem Durchmesser.

b. Zeigen Sie: Es gibt genau 7 Gerüste mit dem minimalen Durchmesser wie in a).

**Aufgabe 36**: **6 Pkt**

a. Es seien ein Graph $G = (E, K)$ und ein kreisloser Routing-Plan $C$ (mit $\pi = 2$ Einträgen) gegeben.

Zeigen Sie: Für die Anzhal $C_2$ der minimalen kritischen Kantenmengen gilt stets

$$C_2 \geq \sum_{x \in E} \lceil \frac{\gamma(x)}{2} \rceil .$$

(Dabei bezeichnet wie üblich $\gamma(x)$ den Grad der Ecke $x$ und $\lceil a \rceil$ die kleinste natürliche Zahl $l$ mit $l \geq a$.)

b. Gegeben sei der vollständige Graph $K_4$.

Zeigen Sie: SP-Routing mit Präferenz führt zu einem Routing-Plan, der optimal ist bezüglich der Approximation

$$\check{z} = 1 - C_2 \cdot q^2$$

für das Zuverlässigkeitspolynom.

**Aufgabe 37**: Für den Graphen $K_5$ sei der folgende Routing Plan gegeben: **6 Pkt**

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | - | 2;3 | 3;2 | 4;5 | 5;4 |
| 2 | 1;3 | - | 3;1 | 4;5 | 5;4 |
| 3 | 1;2 | 2;1 | - | 4;5 | 5;4 |
| 4 | 1;5 | 2;3 | 3;2 | - | 5;1 |
| 5 | 1;4 | 2;3 | 3;2 | 4;1 | - |

Bestimmen Sie die Anzahl der minimalen kritischen Kantenmengen. Interpretieren Sie das Ergebnis.

**Übungsaufgaben zum Kapitel "Kleine Welten"**

**Aufgabe 38**: In einem Graphen $G = (E, K)$ bezeichnet man die Länge eines kürzesten Kreises als *Taillenweite* $g(G)$. Zeigen Sie: **6 Pkt**

Enthält der Graph $G = (E, K)$ einen Kreis, so gilt $g(G) \leq 2 \cdot diam(G) + 1$.

**Aufgabe 39**: Wie groß ist die mittlere Kantenzahl der Graphen $G \in \mathfrak{G}(n, p)$ ? **4 Pkt**

**6 Pkt** **Aufgabe 40**: Bestimmen Sie den mittleren Abstand und den Cluster-Koeffizienten in dem Graphen $K_{10^9,10^3}$.

**4 Pkt** **Aufgabe 41**: In einem skalenfreien Graphen mit 1000 Ecken sei für $k > 20$

$$p(k) \approx 100 \cdot k^{-2}.$$

Vergleichen Sie die Wahrscheinlichkeit, dass eine beliebig herausgegriffene Ecke 100 bzw. 200 Nachbarn besitzt, mit den entsprechenden Wahrscheinlichkeiten in einem Zufallsgraphen $G \in \mathfrak{G}(1000; 0, 1)$.

**6 Pkt** **Aufgabe 42**: Bestimmen Sie die epidemische Schranke $\lambda_c$ für einen vollständigen Graphen $K_n$, in dem zum Startzeitpunkt ein Knoten infiziert ist und die Ausbreitung der Infektion entsprechend dem SIS-Modell erfolgt.

**6 Pkt** **Aufgabe 43**: Geben Sie zwei Graphen mit 5 Ecken und der Gradfolge (3,3,2,2,2) an, die unterschiedliche $s(G)$-Werte besitzen.

## Lösungshinweise zu Übungsaufgaben

**Lösung zu Aufgabe 1:**

Es wird die Kette von Implikationen i) $\rightarrow$ ii) $\rightarrow$ iii) $\rightarrow$ i) gezeigt.

i) $\rightarrow$ ii): Ein Baum ist als zusammenhängender Graph ohne Kreise definiert. Es sei also $P$ ein Baum und $k = \{e, f\}$ eine Kante, die keine Endkante ist. Wäre $k$ keine Brücke, so müsste in $P \backslash k$ ein Weg zwischen $e$ und $f$ existieren, mit $k$ zusammen ergäbe sich dann in $P$ ein Kreis. Also muss $k$ eine Brücke sein.
ii) $\rightarrow$ iii): Da $P$ als zusammenhängend vorausgesetzt ist, muss nur gezeigt werden, dass zwei Ecken $a$ und $b$ niemals durch zwei verschiedene Wege in $P$ verbindbar sind. Wäre dies aber für zwei Ecken $a$ und $b$ der Fall, so wäre jede Kante auf einem dieser Wege, die nicht zu dem anderen Weg gehört, weder Endkante noch Brücke.
iii) $\rightarrow$ i): Es folgt sofort, dass $P$ zusammenhängend ist. Würde $P$ einen Kreis enthalten, so würden Ecken $a$ und $b$ existieren, die durch mehrere Wege verbindbar wären.

**Lösung zu Aufgabe 2:**

ii) und iii) ergeben zusammen die Definition eines Baumes. Sind i) und iii) erfüllt, so ist nach Satz 1.2-3 $G$ ein Baum. Nun seien für $G$ i) und ii) erfüllt. Nach ii) ist $G$ ein Wald. $G$ habe $k$ Komponenten . Mit Hilfssatz 1.2-2 folgt, dass für die Anzahl der Kanten von $G$ gelten muss $m = n - k$. Mit i) hat man schließlich $k = 1$, d. h. $G$ ist zusammenhängend.

**Lösung zu Aufgabe 3:**

$e$ und $f$ seien beliebige Ecken von $G$. Zu zeigen ist, dass $e$ und $f$ in $G$ durch einen Weg verbunden werden können. $N_e \subseteq E$ bezeichne die Menge der Nachbarecken von $e$, $N_f \subseteq E$ entsprechend die Nachbarn von $f$. Nach Voraussetzung haben sowohl $N_e$ als auch $N_f$ mindestens $\frac{n-1}{2}$ viele Elemente. Für $N'_e = N_e \cup \{e\}$ und $N'_f = N_f \cup \{f\}$ gilt dann

$$|N'_e| \geq \frac{n-1}{2} + 1 = \frac{n+1}{2}, \qquad |N'_f| \geq \frac{n+1}{2}.$$

Also können $N'_e$ und $N'_f$ nicht disjunkt sein, denn dann hätte $N'_e \cup N'_f$ mindestens $\frac{n+1}{2} + \frac{n+1}{2} = n+1$ viele Elemente. Dies bedeutet aber, dass die Menge $N'_e \cup N'_f$ einen zusammenhängenden Untergraphen von $G$ aufspannt, $e$ und $f$ sind also verbindbar.

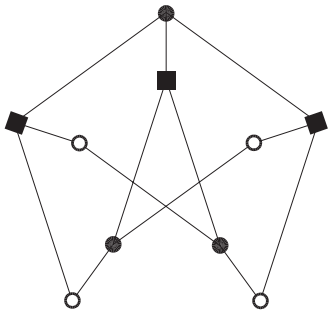
**Lösung zu Aufgabe 4:**

$M$ habe $k$ Zusammenhangskomponenten, $B_1, \ldots, B_k$ seien die Bäume, die den Wald $B$ bilden. $z_1, \ldots, z_{n-k}$ seien die Zweige (bezüglich $B$) und $S_1, \ldots, S_{n-k}$ die zugehörigen fundamentalen Schnitte. Es ist zu zeigen, dass $S_1, \ldots, S_{n-k}$ eine Basis des Vektorraums $\mathbf{S}^*$ bilden. Da jeder der Zweige $z_i$ nur zu $S_i$ und keinem der übrigen $S_j$ gehört, kann keiner der Schnitte $S_i$ als Summe der restlichen $S_j$ dargestellt werden, die Schnitte $S_1, \ldots, S_{n-k}$ sind folglich linear unabhängig. Es bleibt zu zeigen, dass jeder Schnitt $S \in \mathbf{S}$ Summe einiger dieser Schnitte $S_i (1 \leq i \leq n-k)$ ist. Sei also $S \in \mathbf{S}$ beliebig, und $z_{i_1}, \ldots, z_{i_r}$ seien diejenigen unter den Zweigen

$z_1, \ldots, z_{n-k}$, die zu $S$ gehören. Wir behaupten, dass $S = S_{i_1} + \ldots + S_{i_r}$ gilt. Da nämlich $S$ und $S_{i_1} + \ldots + S_{i_r}$ genau die gleichen Zweige enthalten, enthält $S' = S + (S_{i_1} + \ldots + S_{i_r})$ überhaupt keine Zweige. Wegen $S' \in \mathbf{S}^*$ folgt jetzt $S' = \emptyset$ und damit die Behauptung.

**Lösung zu Aufgabe 5:**

In dem folgenden Diagramm ist ein Teilgraph des Petersen-Graphen dargestellt, der zu $K_{3,3}$ homöomorph ist. Nach dem Satz von Kuratowski kann der Petersen-Graph damit nicht planar sein.



**Lösung zu Aufgabe 6:**

$M$ habe $k$ viele Komponenten. Es kann vorausgesetzt werden, dass $I$ die in Abschnitt 2.2 beschriebene diagonalähnliche Gestalt hat. Es gilt dann

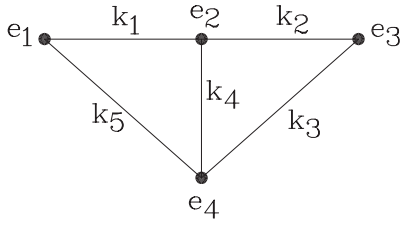
$$rg_2(I) = rg_2(I(M_1)) + \ldots + rg_2(I(M_k)).$$

Nach Folgerung 2.2-1 gilt $rg_2(I(M_j)) = \rho(M_j)$ für jede der Komponenten $M_j$, d. h. es folgt

$$rg_2(I) = \rho(M_1) + \ldots + \rho(M_k) = \rho(M).$$

**Lösung zu Aufgabe 7:**

i.




Die Elemente von **C** sind $C_1 = \{k_1, k_4, k_5\}$, $C_2 = \{k_2, k_3, k_4\}$, $C_3 = \{k_1, k_2, k_3, k_5\}$. Damit ergibt sich

$$C(G) = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

ii.

$$I(G) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Damit erhält man:

$$I \cdot \quad C^t = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

**Lösung zu Aufgabe 8:**

Gegeben seien reelle Zahlen $a_1, \ldots, a_n$. Es wird die kleinste dieser Zahlen herausgesucht.

**Algorithmus**:

```
begin
  m := a_1;
  i := 1;
  while i < n do
  begin
    i := i + 1;
    if a_i < m
      then m := a_i;
  end
end
.
```

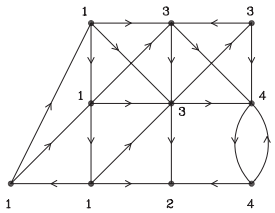
Da die while-Schleife $(n-1)$-mal durchlaufen wird und in ihr ein Vergleich und höchstens eine Zuweisung vorgenommen werden, handelt es sich um einen Algorithmus der Komplexität $O(n)$

**Lösung zu Aufgabe 9:**

Es sei $X \subseteq E$ eine Knotenüberdeckung in $G$, ferner seien $\{e, f\} \in E \backslash X$. Es folgt sofort $\{e, f\} \in \overline{K}$ bzw. $\{e, f\} \notin K$, denn wäre $\{e, f\} \in K$, so müsste $e \in X$ oder $f \in X$ sein. Umgekehrt sei nun für $X \subseteq E$ vorausgesetzt, dass $E \backslash X$ in $\overline{G}$ ein Simplex als Untergraphen bildet, ferner sei $k \in K$ mit $k = \{e, f\}$. Zu zeigen ist, dass $e \in X$ oder $f \in X$ ist. Wären aber $e, f \in E \backslash X$, so müsste gelten $\{e, f\} \in \overline{K}$ bzw. $\{e, f\} \notin K$ im Widerspruch zur Voraussetzung.

**Lösung zu Aufgabe 10:**

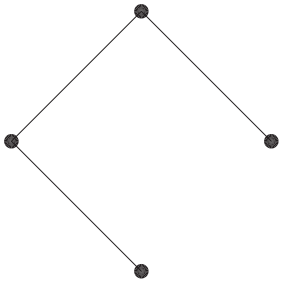
Die vier Zusammenhangskomponenten sind in dem folgenden Bild angedeutet:



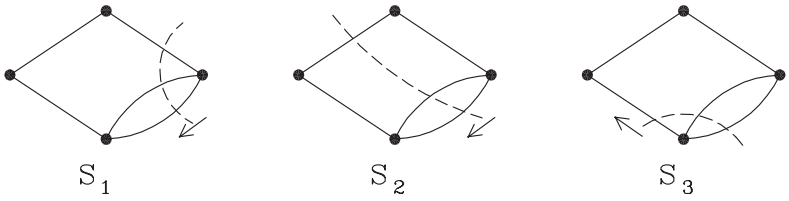

Wie man sieht, haben die zwischen zwei Komponenten laufenden Bögen stets die gleiche Richtung.

**Lösung zu Aufgabe 11:**

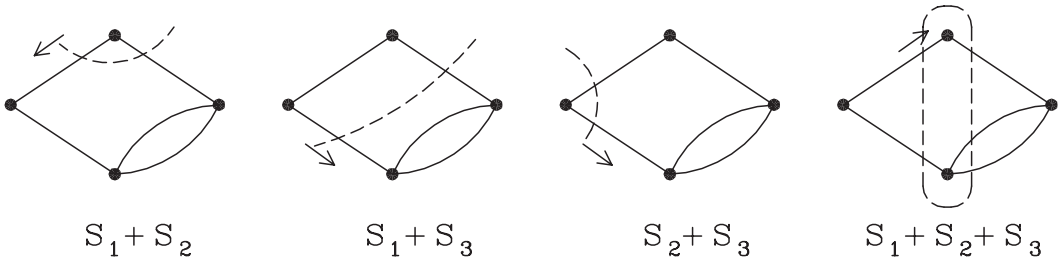
Wir wählen das hier dargestellte Gerüst:



Dies führt zu den folgenden (mit einer Orientierung versehenen) fundamentalen Schnitten:



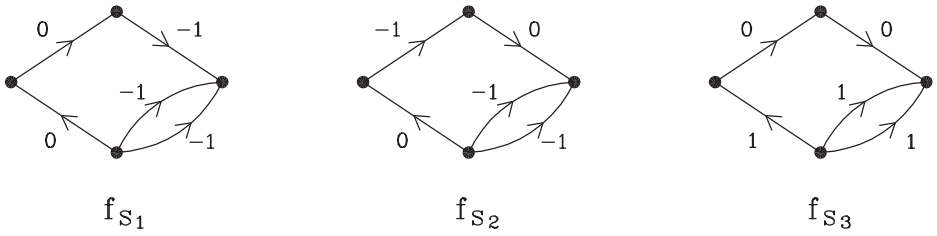

Durch Linearkombinationen ergeben sich zusätzlich die Schnitte:




Damit hat man als Schnitt-Bogen-Matrix:

$$S = \begin{pmatrix} 0 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 1 & 1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

Die Bewertungen $f_{S_1}$, $f_{S_2}$ und $f_{S_3}$ bilden eine Basis des Vektorraums $\mathbf{S}(\mathbb{R})$:




$\mathrm{f}_{S_1}$ $\mathrm{f}_{S_2}$ $\mathrm{f}_{S_3}$

**Lösung zu Aufgabe 12:**

Die einfachste Argumentation ist die folgende: Da $\vec{G}$ (bzw. $G$) ein Baum ist, gilt $\mu(G) = 0$, d. h. der Vektorraum $\mathbf{C}(W)$ hat (nach Satz 5.2-5) die Dimension Null und besteht nur aus dem Nullvektor.

**Lösung zu Aufgabe 13:**

Es ergeben sich die folgenden Matrizen:

$$D_0 = \begin{pmatrix} 0 & 0 & 3 & \infty & 4 \\ \infty & 0 & 1 & 3 & \infty \\ \infty & \infty & 0 & 1 & \infty \\ \infty & \infty & \infty & 0 & 1 \\ \infty & 0 & \infty & \infty & 0 \end{pmatrix}$$

$$D_1 = D_0$$

$$D_2 = \begin{pmatrix} 0 & 0 & 1 & 3 & 4 \\ \infty & 0 & 1 & 3 & \infty \\ \infty & \infty & 0 & 1 & \infty \\ \infty & \infty & \infty & 0 & 1 \\ \infty & 0 & 1 & 3 & 0 \end{pmatrix}$$

$$D_3 = \begin{pmatrix} 0 & 0 & 1 & 2 & 4 \\ \infty & 0 & 1 & 2 & \infty \\ \infty & \infty & 0 & 1 & \infty \\ \infty & \infty & \infty & 0 & 1 \\ \infty & 0 & 1 & 2 & 0 \end{pmatrix}$$

$$D_4 = \begin{pmatrix} 0 & 0 & 1 & 2 & 3 \\ \infty & 0 & 1 & 2 & 3 \\ \infty & \infty & 0 & 1 & 2 \\ \infty & \infty & \infty & 0 & 1 \\ \infty & 0 & 1 & 2 & 0 \end{pmatrix}$$

$$D_5 = \begin{pmatrix} 0 & 0 & 1 & 2 & 3 \\ \infty & 0 & 1 & 2 & 3 \\ \infty & 2 & 0 & 1 & 2 \\ \infty & 1 & 2 & 0 & 1 \\ \infty & 0 & 1 & 2 & 0 \end{pmatrix}$$

Aus $D_5$ können nun die Abstände abgelesen werden.

**Lösung zu Aufgabe 14:**

Die zu Anfang definierten $d$-Werte bezeichnen wir mit $d_1(e)$, die in der $(k-1)$-ten Iteration der repeat-Schleife definierten $d$-Werte mit $d_k(e)$. Mit Induktion wird jetzt zunächst überlegt, dass $d_k(e)$ die Länge eines kürzesten Weges von $s$ nach $e$ ist, der höchstens $k$ Kanten enthält. Für $k = 1$ ist dies offensichtlich. Es gelte nun

die Induktionsvoraussetzung, dass für ein beliebiges festes $k \geq 1$ $d_k(e)$ für jede Ecke $e$ die Länge eines kürzesten Weges von $s$ nach $e$ mit höchstens $k$ Kanten ist. Die Ecke $e$ sei jetzt ebenfalls fest gewählt. Nach der Definition des Algorithmus folgt nun, dass entweder $d_{k+1}(e) = d_k(e)$ ist oder es eine Vorgängerecke $f$ von $e$ gibt mit $d_{k+1}(e) = d_k(f) + w(fe) < d_k(e)$. ($f$ sei so gewählt, dass $d_k(f) + w(fe)$ minimal wird.) Da nun $d_k(f)$ nach Induktionsvoraussetzung die Länge eines kürzesten Weges mit höchstens $k$ vielen Kanten von $s$ nach $f$ ist, folgt daraus, dass $d_{k+1}(e)$ die Länge eines kürzesten Weges von $s$ nach $e$ mit höchstens $k+1$ vielen Kanten sein muss. Damit ist die Behauptung über die Werte $d_k(e)$ gezeigt.

Da $(G, w)$ keine negativen Kreise enthält, kann ein kürzester Weg höchstens $|E|-1$ viele Kanten enthalten. Somit muss die until-Bedingung in der repeat-Schleife spätestens bei $k = |E| - 1$ erfüllt sein. Es sind dann die kürzesten Abstände bestimmt, und man hat eine Komplexität von $0(|E|^3)$.

**Lösung zu Aufgabe 15:**

Angenommen, $G$ enthält zwei minimale Gerüste $B_1 = (E, K_1)$ und $B_2 = (E, K_2)$. Wir können die Kanten von $B_1$ und $B_2$ ohne Beschränkung der Allgemeinheit so numerieren, dass folgende Bedingungen erfüllt sind:

$$K_1 = \{k_1, \ldots, k_{i-1}, k_i, \ldots, k_{n-1}\},$$

$$K_2 = \{k_1, \ldots, k_{i-1}, k'_i, \ldots, k'_{n-1}\}$$

mit

$$1 \leq i \leq n-1,$$

$$w(k_1) < \ldots < w(k_{n-1})$$

und

$$w(k_1) < \ldots < w(k_i) < w(k'_i) < \ldots < w(k'_{n-1}).$$

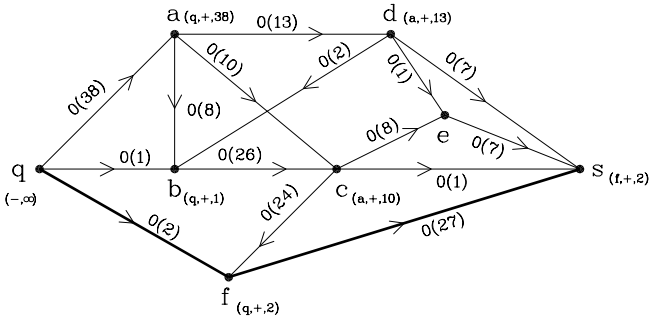
Fügen wir nun $k_i$ zu $B_2$ hinzu, so erhalten wir den Kreis $C_{B_2}(k_i)$. Nach Satz 6.2-2 gilt

$$w(k_i) \geq w(k) \text{ für jede Kante } k \text{ in} C_{B_2}(k_i).$$

Da die Gewichte alle verschieden sind, müssen die Kanten $k \neq k_i$ in $C_{B_2}(k_i)$ unter den ersten Kanten $k_1, \ldots, k_{i-1}$ von $B_2$ sein. Da diese Kanten auch zu $B_1$ gehören, enthält $B_1$ den ganzen Kreis $C_{B_2}(k_i)$, und dies ist ein Widerspruch. Damit ist der Beweis erbracht.

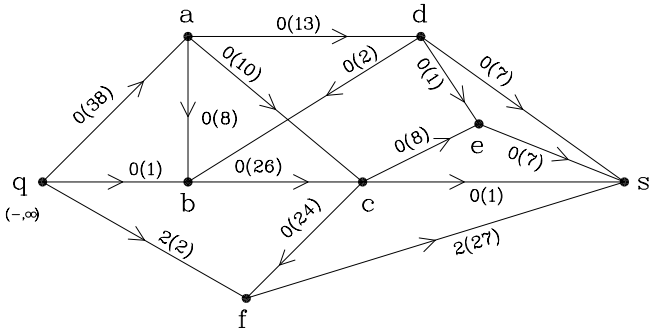
**Lösung zu Aufgabe 16:**

Ein maximaler Fluss kann mit dem Algorithmus von Edmonds und Karp bestimmt werden. Ausgehend vom Null-Fluss $f_0$ ergibt sich zuerst der zunehmende Weg $q \to f \to s$, auf dem der Fluss um zwei Einheiten erhöht werden kann:

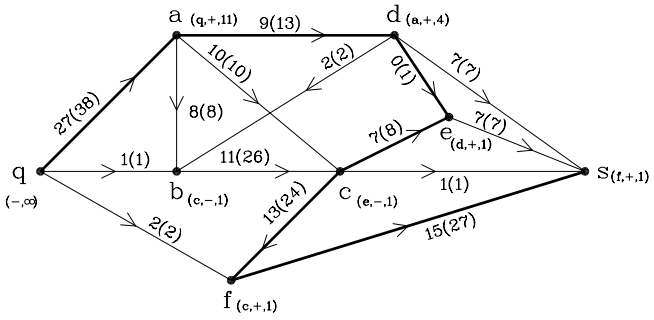



$q$ $\surd$
$a$ $\surd$
$b$ $\surd$
$f$ $\surd$
$c$
$d$
$s$

Nach der Flussvergrößerung hat man also folgendes Bild mit Fluss $f_1$:

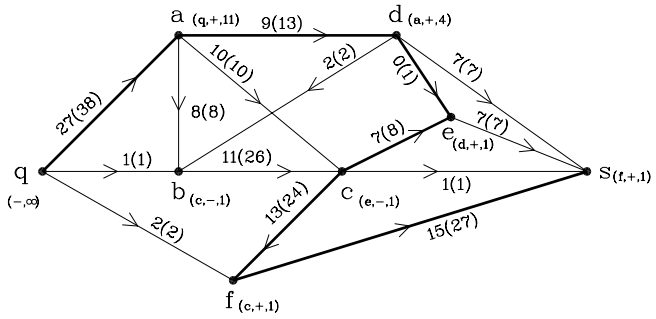



Wir wollen nun in der Lösung einige Schritte überspringen. Nach acht Flussvergrößerungen ergibt sich der in dem folgenden Diagramm dargestellte Fluss $f_8$. (Dabei sind stets, falls eine Wahl zu treffen war, die Nachbarecken einer abgesuchten Ecke in alphabetischer Reihenfolge markiert worden.)



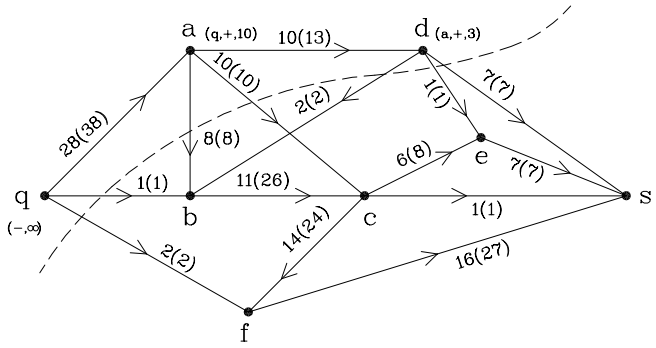

$w(f_8) = 30$

Ein weiterer Schritt des Algorithmus führt zu den folgenden Markierungen und dem herausgehobenen zunehmenden Weg:




$q$ $\surd$
$a$ $\surd$
$d$ $\surd$
$e$ $\surd$
$c$ $\surd$
$b$ $\surd$
$f$ $\surd$
$s$

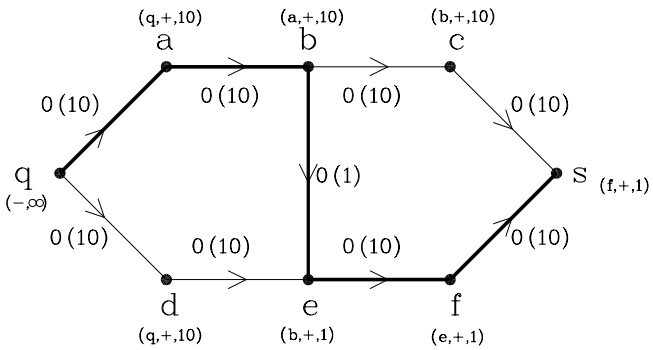
Nach der Flussvergrößerung hat man $f_9$ mit $w(f_9) = 31$. Im nächsten Schritt stoppt der Algorithmus. Das folgende Bild zeigt den maximalen Fluss $f_9$ und die Markierungen aus dem letzten Algorithmusschritt, der resultierende minimale Schnitt kann ebenfalls abgelesen werden.



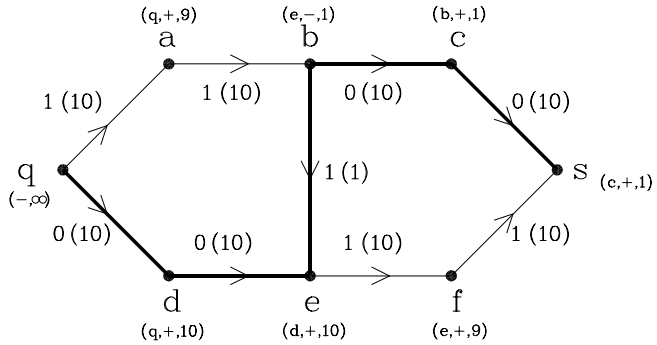

$q$ $\surd$
$a$ $\surd$
$d$ $\surd$

**Lösung zu Aufgabe 17:**

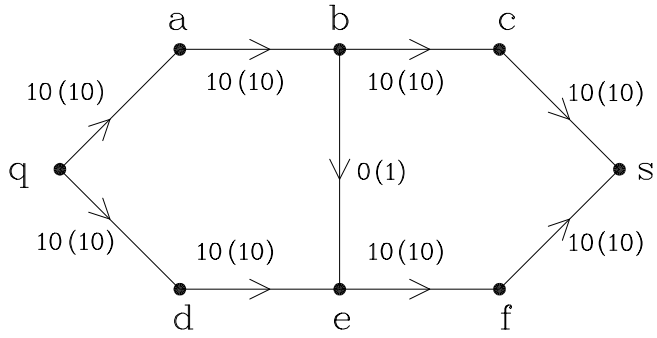
a. Je nach Absuchreihenfolge der bereits markierten Ecken, für die beim Algorithmus von Ford und Fulkerson eine gewisse Freiheit besteht, kommt man verschieden schnell zu dem (hier eindeutigen) maximalen Fluss. Sucht man z. B. im ersten Schritt in der Reihenfolge $q, a, b, e, f$ ab, so ergeben sich die folgenden Markierungen mit herausgehobenem zunehmendem Weg:




Beim nächsten Schritt ist folgendes Bild möglich




Indem man in dieser Weise fortfährt, kommt man erst nach $20$ Flussvergrößerungen zum maximalen Fluss, der hier dargestellt ist:




Mit dem Algorithmus von Edmonds und Karp bekommt man diesen Fluss schon nach zwei Flussvergrößerungen.

b. Verallgemeinert man das obige Beispiel dahingehend, dass man die Kantenkapazität $10$ durch irgendeine natürliche Zahl $p$ ersetzt, so zeigen die überlegungen aus a), dass der Algorithmus von Ford und Fulkerson bei ungeschickter Vorgehensweise $2p$ viele Flussvergrößerungen bis zur Ermittlung des maximalen Flusses braucht. Die Komplexität ist also von den Kapazitätswerten abhängig und kann folglich, da letztere beliebig groß sein können, nicht durch ein Polynom in der Anzahl $n$ der Ecken des Digraphen abgeschätzt werden.

**Lösung zu Aufgabe 18:**

Die Aussage, dass $Gp$-fach zusammenhängend ist, bedeutet: man muss mindestens $p$ Ecken aus $G$ entfernen, damit der Rest des Graphen nicht zusammenhängend ist. Wenn je zwei Ecken in $G$ durch mindestens $p$ viele eckendisjunkte Wege verbunden sind, ist dies offenbar erfüllt.

Nun sei umgekehrt $G$ als $p$-fach zusammenhängend vorausgesetzt, $q$ und $s$ seien Ecken in $G$. Jede $q$ und $s$ trennende Eckenmenge muss mindestens $p$ viele Ecken enthalten. Setzt man $q$ und $s$ als nicht-benachbart voraus, so gibt es nach dem Satz von Menger (für ungerichtete Graphen) mindestens $p$ viele eckendisjunkte Wege von $q$ nach $s$. Es bleibt der Fall zu betrachten, dass $q$ und $s$ benachbart sind. $G'$ sei der Graph, der durch Entfernen der Kante $\{q, s\}$ entsteht. $G'$ ist $(p-1)$-fach zusammenhängend, und da $q$ und $s$ in $G'$ nicht benachbart sind, kann man wie oben auf die Existenz von $p-1$ vielen eckendisjunkten Wegen von $q$ nach $s$ in $G'$ (und damit auch in $G$) schließen. Zusammen mit der Kante $\{q, s\}$ hat man in $Gp$ viele Wege.

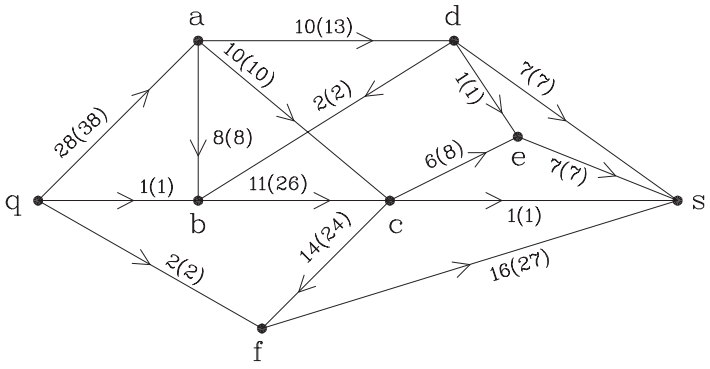
**Lösung zu Aufgabe 19:**

$G = (E_1 \cup E_2, K)$ sei ein regulärer bipartiter Graph, $\gamma \geq 1$ sei der allen Ecken gemeinsame Grad. $A \subseteq E_1$ sei eine beliebige Teilmenge, wir setzen $a := |A|$. Es gibt genau $a\gamma$ viele Kanten der Form $ef$ mit $e \in A$ und $f \in E_2$. Da jede Ecke von $E_2$ auf genau $\gamma$ vielen Kanten liegt, müssen die $a\gamma$ vielen Endecken in $E_2$ dieser $a\gamma$ vielen Kanten mindestens $a$ verschiedene Ecken enthalten, was $|\Phi(A)| \geq |A|$ bedeutet. Mit dem Satz von Hall kann nun geschlossen werden, dass $G$ eine vollständige Korrespondenz besitzt.

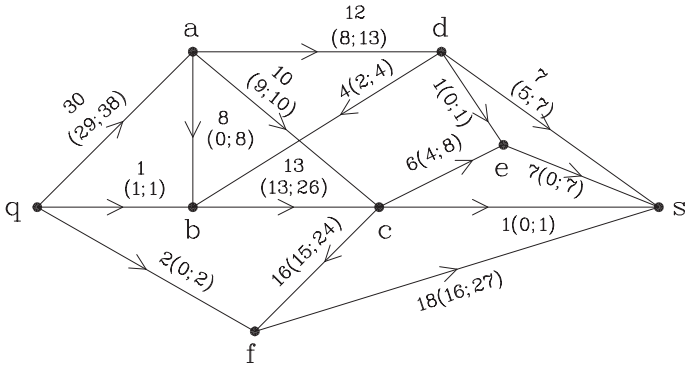
**Lösung zu Aufgabe 20:**

Man kann das Problem so lösen, wie in Abschnitt 7.6 beschrieben. Es ist dann zuerst irgendein zulässiger Fluss $f_0$ zu finden, davon ausgehend kann mit dem Algorithmus von Ford und Fulkerson (d. h. durch zunehmende Wege) ein maximaler zulässiger Fluss bestimmt werden.

Man kann sich jedoch auch die Lösung der Aufgabe 6.1 (Einsendeaufgabe zu Abschnitt 7.6) zunutze machen, wo der folgende maximale Fluss für das dort gegebene Fluss-Netz gefunden wurde:




Bei der vorliegenden Aufgabe handelt es sich um denselben Digraphen, die obere Kapazität ist nur auf der gerichteten Kante $db$ von $2$ auf $4$ geändert. Wegen der nun auch gegebenen unteren Kapazitäten ist der obige Fluss nicht zulässig. Man sieht jedoch sofort, dass auf dem Weg $q \to a \to d \to b \to c \to f \to s$ der Fluss um $2$ erhöht werden kann, Resultat ist der folgende zulässige Fluss:




Da der Fluss auf allen die Menge $Q = \{q, a, d\}$ verlassenden Kanten die obere Kapazitätsgrenze erreicht, muss dieser Fluss maximal sein.

**Lösung zu Aufgabe 21:**

Wir wählen zwei $n$-elementige Mengen $Q = \{q_1, \ldots, q_n\}$ und $S = \{s_1, \ldots, s_n\}$ und zwei weitere Elemente $q$ und $s$. Auf der Eckenmenge $E := Q \cup S \cup \{q, s\}$ definieren wir einen Digraphen $G$. Die Kantenmenge $K$ besteht aus allen $qx$, allen $xy$ und allen $ys$ (mit $x \in Q$ und $y \in S$). Jeder Kante $k \in K$ wird die Kapazität $c(k) = 1$ zugeordnet. Für das Fluss-Netz $(G, c, q, s)$ definieren wir noch eine Kostenfunktion durch $\gamma(q, x) := \gamma(y, s) := 0$ und $\gamma(q_i, s_j) := a_{ij}$, wobei wie üblich $a_{ij}$ den Eintrag in der $i$-ten Zeile und $j$-ten Spalte der Matrix $A$ bezeichnet. Nun entsprechen die ganzzahligen Flüsse vom Wert $n$ mit minimalen Kosten genau den Lösungen des Zuordnungsproblems für die gegebene Matrix $A$.

**Lösung zu Aufgabe 22:**

Die Nachricht kommt nur dann nach $11$ msec bei $F$ an, wenn einer der kürzesten Wege

$$\begin{array}{ccccccc} A & \rightarrow & B & \rightarrow & D & \rightarrow & F \\ A & \rightarrow & B & \rightarrow & E & \rightarrow & F \\ A & \rightarrow & C & \rightarrow & D & \rightarrow & F \\ A & \rightarrow & C & \rightarrow & E & \rightarrow & F \end{array}$$

genommen wird.

Jeder dieser Wege tritt mit einer Wahrscheinlichkeit von

$$\frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{18}$$

auf, mithin beträgt die gefragte Wahrscheinlichkeit $\frac{4}{18}$ bzw. $\frac{2}{9}$.

**Lösung zu Aufgabe 23:**

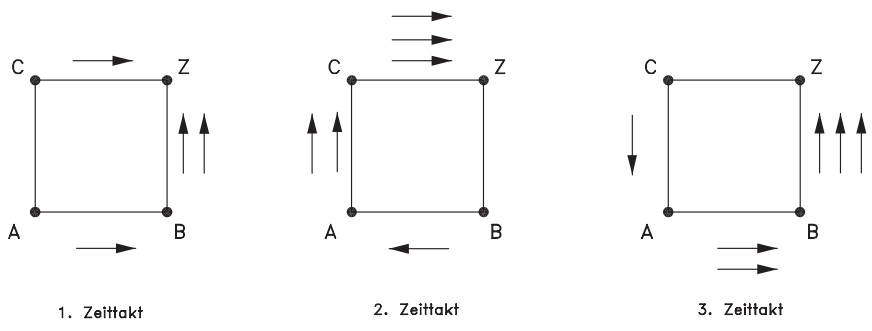
Zunächst versendet $B$ vier Meldungen (mit Zählerwert 1) an seine vier Nachbarn. Danach entstehen folgende Meldungen mit Zählerwert 2:

$$\begin{array}{ccc} A & \rightarrow & C \\ C & \rightarrow & A,D,E \\ D & \rightarrow & C,F \\ E & \rightarrow & C,F \end{array}$$

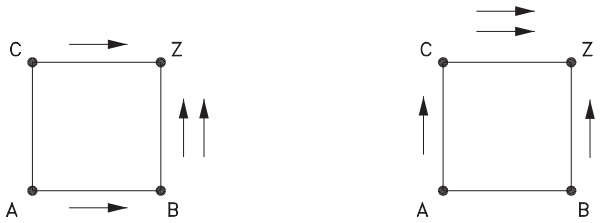
Dies ergibt in der Summe 12 Meldungen.

**Lösung zu Aufgabe 24:**

a. In den folgenden Diagrammen ist die Entwicklung in den ersten drei Zeittakten dargestellt, wobei o.B.d.A. A im ersten Takt den Nachbarn B als Zwischenknoten nutzt. Die Verkehrsbelastung einer Kante ist durch die Anzahl der Pfeile angedeutet. Im folgenden wechseln sich stets die im 2. und 3. Zeittakt vorliegenden Situationen ab.



b. Mit einem Biasfaktor von $\alpha = 1$ ergibt sich ein Oszillieren zwischen den beiden folgenden Mustern:




c. Eine Erhöhung des Biasfaktors würde hier keine weitere Verbesserung bewirken. Der Grund liegt in der Symmetrie des gegebenen Graphen. Abhilfe schaffen könnte hier die Einführung verschiedener "Grundlasten" $\alpha_k$ für die einzelnen Kanten $k$, um die Symmetrie auf diese Weise zu brechen.

**Lösung zu Aufgabe 25:**

Wir setzten o.B.d.A. voraus, dass jeder Knoten von A zuletzt eine Nachricht mit Folgenummer 1 erhalten (und diese abgespeichert) hat. Ferner sei zunächst vorausgesetzt, dass die Laufzeiten auf den Kanten sehr groß sind gegenüber den Verarbeitungenszeiten in den Knoten.
A sendet Meldungen (mit Folgenummer 2) an B und C. Daraufhin schickt

- B entsprechende Meldungen an C, D unde E,
- C Meldungen an B und E.

Schließlich sendet

- D eine Meldung an E,
- E ein Meldung an B oder C und an D
  (abhängig von der Reihenfolge der Verarbeitung der erhaltenen Meldungen von C. bzw. B).

Insgesamt ergeben sich so 10 Meldungen.

Ohne die obige Voraussetzung über Lauf- und Verarbeitungszeiten mag es z. B. möglich sein, dass B eine Meldung von C erhält, bevor die Meldung von A in B eintrifft. In diesem Falle würde B nur Meldungen an D und E schicken, jedoch keine an C.

**Lösung zu Aufgabe 26:**

Zwei beliebige Knoten $x$ und $k (x \neq k)$ seien gegeben. Zu zeigen ist: in $G(\rightarrow k)$ gilt entweder $x \rightarrow k$, oder aber es existieren Zwischenknoten $x_1, x_2, \ldots, x_t$ $(t \geq 1)$ mit

$$x \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots x_t \rightarrow k\,.$$

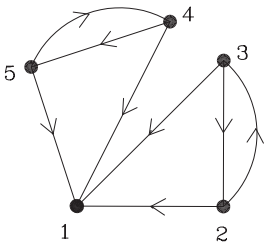
Angenommen, es ist nicht bereits $x \rightarrow k$, denn in diesem Fall muss nichts gezeigt werden. Da den Knoten $x$ in $G(\rightarrow k)$ mindestens zwei gerichtete Kanten verlassen, kann ein $x_1$ mit $x \rightarrow x_1$ gewählt werden. $x_2$ sei nun ein Nachfolger von $x_1$ in $G(\rightarrow k)$, der nicht gleich $x$ ist. Entsprechend kann eine Folge

$$x \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots x_{i-1} \rightarrow x_i \rightarrow x_{i+1} \ldots$$

konstruiert werden derart, dass stets (falls $x_i \neq k$ ist) $x_{i+1}$ als Nachfolger von $x_i$ mit $x_{i+1} \neq x_{i-1}$ gewählt wird. Da $G(\rightarrow k)$ nur endlich viele Knoten enthält, aber auch kein gerichteter Kreis entstehen kann, bleibt nur die Möglichkeit, dass die Folge abbricht, indem einmal $x_{i+1} = k$ wird. Damit ist der Beweis erbracht.

**Lösung zu Aufgabe 27:**

Zunächst ist es hilfreich, sich z. B. $G(\rightarrow 1)$ zu veranschaulichen:



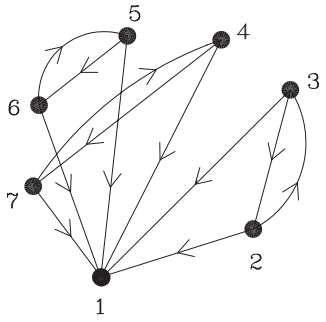

An dem Routing-Plan liest man ab, dass die Digraphen $G(\rightarrow 2)$, $G(\rightarrow 3)$ usw. durch "Rotation" aus obigem Bild entstehen. Daran sieht man sofort, dass in keinem der $G(\rightarrow k)$ ein gerichteter Kreis mit mehr als zwei Kanten existiert, mithin ist $C$ kreisfrei.

$C$ ist nicht bidirektional: fallen z. B. die Kanten 15 und 25 aus, so kann zwar Knoten 5 noch Nachrichten an 1 versenden (über $5 \rightarrow 4 \rightarrow 1$), jedoch 1 keine an 5.

Da bei diesem Routing-Plan stets der Erstweg aus einer direkten Kante besteht und es genau einen Zweitweg gibt, welcher zwei Kanten hat, wird immer unter den physikalisch noch möglichen Wegen ein möglichst kurzer ausgewählt.

**Lösung zu Aufgabe 28:**

$G(\rightarrow 1)$ sieht folgendermaßen aus:




Der Routing-Plan ist hier so aufgebaut, dass z. B. die aus den Knoten 1,5 und 6 bestehende "Zelle" auch in $G(\rightarrow 5)$ und $G(\rightarrow 6)$ entsprechend genutzt wird. Mathematisch verbirgt sich dahinter eine Überdeckung der Kantenmenge durch 7 Dreiecke ("Zellen"), die für das Routing verwendet werden.

Aus dieser Struktur ergibt sich sofort, dass $C$ sowohl kreisfrei als auch bidirektional ist und dass stets möglichst kurze Wege gewählt werden.

**Lösung zu Aufgabe 29:**

Gefragt ist nach dem Wert von $f_5$, wenn $p = \frac{1}{2}$ eingesetzt wird (s. Satz 9.2-1). Mit der in Satz 9.2-2 abgeleiteten rekursiven Formel hat man

$$f_5 = 1 - \binom{4}{0} f_1 \left(\frac{1}{2}\right)^4 - \binom{4}{1} f_2 \left(\frac{1}{2}\right)^6 - \binom{4}{2} f_3 \left(\frac{1}{2}\right)^6 - \binom{4}{3} f_4 \left(\frac{1}{4}\right)^4,$$

wobei $f_1, f_2, f_3$ und $f_4$ aus Beispiel 9.2-4 übernommen werden können.

Es ergibt sich:

| | | |
|---|---|---|
| $f_1$ | = | 1 |
| $f_2$ | = | 0,5 |
| $f_3$ | = | 0,5 |
| $f_4$ | = | 0,59375 |

Mit obiger Formel erhält man daraus $f_5 = 0,7109375$.

**Lösung zu Aufgabe 30:**

Das gesuchte Polynom hat die Form

$$z(G) = F_3 p^3(1-p)^3 + F_4 p^4(1-p)^2 + F_5 p^5(1-p) + F_6 p^6,$$

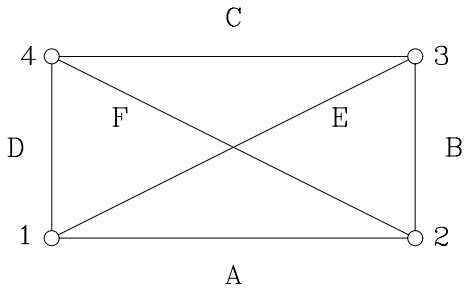
denn es ist $F_0 = F_1 = F_2 = 0$, da eine funktionsfähige Kantenmenge notwendigerweise einen zusammenhängenden Teilgraphen aufspannen muss.

Offensichtlich ist $F_6 = 1$. Auch $F_5$ ist leicht zu sehen, denn je 5 der 6 Kanten reichen (wegen der jeweils 2 Einträge und der Kreisfreiheit) für die Funktionsfähigkeit aus.

Zur Bestimmung von $F_4$ greifen wir auf die Beziehung

$$F_4 = 15 - C_2$$

zurück; es genügt also, die Anzahl $C_2$ der zweielementigen kritischen Kantenmengen zu bestimmen.




Wie man anhand des Routing-Plans nachprüft, sind die Mengen $\{A,B\}, \{A,D\}, \{A,E\}, \{A,F\}, \{B,E\}, \{C,D\}, \{D,E\}$ und $\{E,F\}$; jede der übrigen 7 zweielementigen Kantenmengen (z. B. $\{A,C\}$ oder $\{B,D\}$) ist nicht kritisch, d. h. ihr Ausfall stört nicht die Funktionsfähigkeit des Netzes. Es folgt also $F_4 = 7$.

Da es sich bei dem vorliegenden Routing-Plan um SP-Routing mit Präferenz handelt (vgl. Abschnitt 8.6), sieht man schließlich leicht, dass $\{A,D,E\}$ die einzige dreielementige funktionsfähige Kantenmenge ist, d. h. es gilt $F_3 = 2$.

Im Ergebnis lautet das gesuchte Polynom

$$\begin{aligned} \text{z(G)} &= \text{p}^3(1-p)^3 + 7p^4(1-p)^2 + 6p^5(1-p) + p^6 \\ &= \text{p}^3 + 4p^4 - 5p^5 + p^6. \end{aligned}$$

**Lösung zu Aufgabe 31:**

Wir bestimmen die Koeffizienten $F_4, F_5, F_6$ und $F_7$ für $G_1$ bzw. $F'_4, F'_5, F'_6$ und $F'_7$ für $G_2$.

Offenbar gilt $F_7 = F'_7 = 1$ und, da keiner der beiden Graphen eine Ecke mit nur einer inzidenten Kante enthält, $F_6 = F'_6 = 7$.

Die beiden Ecken von $G_1$ von Grad 2 indizieren 2 zweielementige kritische Kantenmengen. Da keine weiteren zweielementigen kritischen Kantenmengen existieren, gilt $C_2 = 2$ und damit

$$F_5 = \binom{7}{2} - 2 = 19$$

Mit analoger Argumentation folgt $F'_5 = 20$.

Auch bei der Bestimmung von $F_4$ bzw. $F'_4$ ist es etwas günstiger, die krtischen dreielementigen Kantenmengen zu betrachten. Man findet $C_3 = 14$ und $C'_3 = 11$, woraus sich (wegen $\binom{7}{4} = 35$)

$$\begin{array}{rrcl} & \mathrm{F}_4 & = & 21 \\ \text{und} & \mathrm{F}'_4 & = & 24 \end{array}$$

ergibt.

Insgesamt hat man somit $F'_i \geq F_i$ für alle i mit $F'_5 > F_5$ und $F'_4 > F_4$. Daraus folgt, dass (sogar jede Kantenwahrscheinlichkeit $p$) $G_2$ zuverlässiger ist als $G_1$:

$$z(G_2)(p) > z(G_1)(p)$$

**Lösung zu Aufgabe 32:**

Es wird zunächst die negative Adjazenzmatrix aufgestellt, in der Hauptdiagonalen werden zusätzlich die Eckengerade eingetragen. Streichen der letzten Zeile und Spalte führt schließlich zu der Matrix $M'$.

$$M' = \begin{pmatrix} 5 & -1 & -1 & -1 & -1 \\ -1 & 5 & -1 & -1 & -1 \\ -1 & -1 & 5 & -1 & -1 \\ -1 & -1 & -1 & 5 & -1 \\ -1 & -1 & -1 & -1 & 5 \end{pmatrix}.$$

Die gesuchte Anzahl der Gerüste ist gleich $det(M')$.

Zur Berechnung von $det(M')$ wird die Matrix auf Diagonalgestalt gebracht. Man erhält:

$$\begin{vmatrix} 5 & -1 & -1 & -1 & -1 \\ -1 & 5 & -1 & -1 & -1 \\ -1 & -1 & 5 & -1 & -1 \\ -1 & -1 & -1 & 5 & -1 \\ -1 & -1 & -1 & -1 & 5 \end{vmatrix}$$

$$= \frac{1}{5 \cdot 5 \cdot 5 \cdot 5 \cdot 5} \begin{vmatrix} 5 & -1 & -1 & -1 & -1 \\ 0 & 24 & -6 & -6 & -6 \\ 0 & -6 & 24 & -6 & -6 \\ 0 & -6 & -6 & 24 & -6 \\ 0 & -6 & -6 & -6 & 24 \end{vmatrix}$$

$$= \frac{1}{4^3 \cdot 5^4} \begin{vmatrix} 5 & -1 & -1 & -1 & -1 \\ 0 & 24 & -6 & -6 & -6 \\ 0 & 0 & 90 & -30 & -30 \\ 0 & 0 & -30 & 90 & -30 \\ 0 & 0 & -30 & -30 & 90 \end{vmatrix}$$

$$= \frac{1}{3^2 \cdot 4^3 \cdot 5^4} \begin{vmatrix} 5 & -1 & -1 & -1 & -1 \\ 0 & 24 & -6 & -6 & -6 \\ 0 & 0 & 90 & -30 & -30 \\ 0 & 0 & 0 & 240 & -120 \\ 0 & 0 & 0 & -120 & 240 \end{vmatrix}$$

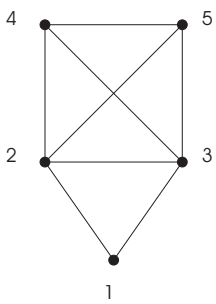
$$= \frac{1}{2 \cdot 3^2 \cdot 4^3 \cdot 5^4} \begin{vmatrix} 5 & -1 & -1 & -1 & -1 \\ 0 & 24 & -6 & -6 & -6 \\ 0 & 0 & 90 & -30 & -30 \\ 0 & 0 & 0 & 240 & -120 \\ 0 & 0 & 0 & 0 & 360 \end{vmatrix}$$

$$= \frac{1}{2 \cdot 3^2 \cdot 4^3 \cdot 5^4} \cdot 5 \cdot 24 \cdot 90 \cdot 240 \cdot 360$$
$$= 1296$$

Der Graph $K_6$ besitzt also 1296 Gerüste. Dies entspricht dem erwarteten Ergebnis, denn nach dem Satz von Cayley ist für $K_6$ die Anzahl der Gerüste gleich $6^4$.

**Lösung zu Aufgabe 33:**

Es ist $n = 5$ ( also $l = 4$) und $m = 8$. Weiter gilt offenbar $c = 2$, denn die beiden zu Ecke 1 inzidenten Kanten bilden eine kritische Menge.




Da keine weitere zweielementige kritische Menge existiert, ist $C_2 = 1$ und damit

$$R_2 = F_6 = \binom{8}{2} - 1 = 27$$

$R_{m-l} = R_4 (= F_4)$ ist die Anzahl der Gerüste und wird durch Aufstellen der Matrix $M'$ (wie in Anhang G beschrieben) ermittelt:

$$M' = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 \\ 0 & -1 & -1 & 3 \end{pmatrix}$$

Man errechnet $R_4 = det(M') = 40$, und damit ergibt sich

$$S = p^8 + 8p^7(1-p) + 27p^6(1-p)^2 + 40p^4(1-p)^4$$

Wegen $\binom{8}{3} = 56$ lauten die einfachen Schranken

$$S \leq z(G) \leq S + 56p^5(1-p)^3.$$

Die Sperner-Schranken ergeben

$$S + \frac{\binom{8}{3}}{\binom{8}{4}} 40p^5(1-p)^3 \leq z(G) \leq S + \frac{\binom{8}{3}}{\binom{8}{2}} 27p^5(1-p)^3$$

bzw.

$$S + 32p^5(1-p)^3 \leq z(G) \leq S + 54p^5(1-p)^3.$$

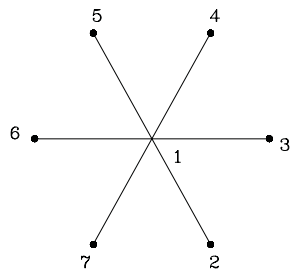
**Lösung zu Aufgabe 34:**

Ein beliebiger Graph $G = (E, K)$ sei gegeben. Wir bilden folgendes Mengensystem: Als Grundmenge $M$ wird die Knotenmenge $E$ gewählt. $\mathcal{M}$ besteht aus allen zweielementigen Teilmengen von $E$, die durch die Kanten des Graphen gegeben sind. Ist nun $T$ eine maximale Transversale dieses Mengensystems, so ist offenbar $M - T$ bzw. $E - T$ eine minimale Knotenüberdeckung von $G$. Mit einem polynomialen Algorithmus zur Lösung des Problems der maximalen Transversalen hätte man somit auch einen solchen für das Problem der minimalen Knotenüberdeckung.

**Lösung zu Aufgabe 35:**

a. In einem Gerüst $B$ von $K_7$ muss es offenbar Ecken geben, die nicht durch eine Kante verbunden sind, folglich gilt auf jeden Fall $diam(B) \geq 2$.

   Für das folgende Gerüst $B_1$ ist $diam(B_1) = 2$:



b. Wir behaupten, dass die Gerüste $B_1, B_2, \ldots, B_7$ die einzigen Gerüste von $K_7$ mit Durchmesser 2 sind; dabei ist mit $B_i$ das Gerüst bezeichnet, welches isomorph zu $B_1$ ist mit der Ecke $i$ als "Zentrum".

Es sei $B$ ein beliebiges Gerüst von $K_7$ mit $diam(B) = 2$. $e, f$ seien Ecken, die in $B$ Abstand 2 haben; $x$ sei eine Ecke, die mit $e$ und mit $f$ verbunden ist. Weder $e$ noch $f$ können außer $x$ weitere Nachbarn haben, denn dies würde entweder der Voraussetzung $diam(B) = 2$ oder der Tatsache widersprechen, dass $B$ keinen Kreis enthält. Also kann nur $x$ weitere Nachbarn haben. Aus dieser Überlegung folgt nun unmittelbar, dass $x$ das "Zentrum" dieses Gerüsts ist und alle anderen Ecken nur mit $x$ verbunden sind.

**Lösung zu Aufgabe 36:**

a. Sei $x$ einbe beliebige Ecke. Zu jeder zu $x$ inzidenten Kante $e = xy$ gibt es eine weitere Kante $f = xz$ derart, dass $\{e, f\}$ eine minimale kritische Kantenmenge ist: $f$ sein in $G(\rightarrow y)$ die zweite die Ecke x verlassende Kante. Also gibt es mindestens $\lceil \frac{\gamma(x)}{2} \rceil$ viele minimale kritische Mengen, deren zwei Kanten die Ecke $x$ gemeinsam haben.

Da für verschiedene Auswahlen von $x$ alle sich so ergebenden minimalen kritischen Mengen verschieden sind, folgt die Ungleichung.

b. Da in $K_4$ alle Ecken den Grad 3 haben, folgt aus a): jeder Routing-Plan für $K_4$ erfüllt

$$C_2 \geq 4 \cdot \lceil \frac{3}{2} \rceil = 8.$$

Da für SP-Routing mit Präferenz gilt $C_2 = 8$, kann es für $K_4$ keinen Routing-Plan mit weniger minimalen kritischen Mengen geben, mithin ist der Routing-Plan optimal bezüglich $\check{z}$.

**Lösung zu Aufgabe 37:**

Der Routing-Plan hat folgende Eigenschaften:

Die Ecken 1,2 und 3 sowie die Ecken 1,4 und 5 bilden "dreieckige Zellen" in dem Sinn, dass eine Ecke jeweils die Kanten zu den beiden anderen Ecken als Erst- und Zweitweg für Ziele innerhalb der Zelle nutzt.

Dies führt zu den minimalen kritischen Kantenmengen

$$\{12, 13\}, \{12, 23\}, \{13, 23\}$$

und

$$\{14, 15\}, \{14, 45\}, \{15, 45\}$$

Weiter gilt: Die Ecken 2 und 3 nutzen jeweils für Ziel 4 die Kante zu Ecke 5 als zweite Priorität und entsprechend für Ziel 5. In analoger Weise nutzen 4 und 5 ihre Kanten nach 2 und 3.

Dies führt zu den vier minimalen kritischen Kantenmengen

$$\{24, 25\}, \{34, 35\}$$

und

$$\{42, 43\}, \{52, 53\}$$

Weitere minimale kritsche Kantenmengen existieren nicht, es gilt also $C_2 = 10$. Da $K_5$ 10 Kanten besitzt und folglich für jeden Routing-Plan $C_2 \geq 10$ gelten muss, hat der gegebene Routing-Plan die kleinstmögliche Anzahl minimaler kritischer Kantenmengen.

**Lösung zu Aufgabe 38:**

$C$ sei ein kürzester Kreis in $G$. Angenommen, es sei $g(G) \geq 2 \cdot diam(G) + 2$. $C$ muss dann zwei Ecken $x$ und $y$ enthalten, die in $C$ einen Abstand von mindestens $diam(G) + 1$ haben. Da $x$ und $y$ in $G$ geringeren Abstand haben, gibt es zwischen ihnen einen kürzesten Weg $W$, der nicht in $C$ enthalten ist. Zusammen mit dem kürzeren der beiden $x - y-$Wege in $C$ ergibt $W$ nun einen kürzeren Kreis als $C$, und dies ist ein Widerspruch.

**Lösung zu Aufgabe 39:**

Da in den betrachteten Zufallsgraphen jede der $\binom{n}{2}$ vielen möglichen Kanten unabhängig voneinander mit Wahrscheinlichkeit $p$ vorhanden ist, ist der Erwartungswert der Kantenzahl der Graphen $G \in \mathfrak{G}(n, p)$ gleich $p \cdot \binom{n}{2}$.

**Lösung zu Aufgabe 40:**

Nach Satz 10.3-1 gilt für den Cluster-Koeffizienten $C = \frac{3(10^3-2)}{4(10^3-1)} \approx 0,75$.

Zur Bestimmung des mittleren Abstands reicht es aus Symmetriegründen, den durchschnittlichen Abstand einer fest gewählten Ecke $e$ zu allen anderen Ecken zu ermitteln. Aufgrund der Konstruktion von $K_{n,z}$ hat $e$ zu $10^3$ vielen Ecken den Abstand 1, zu $10^3$ weiteren den Abstand 2 usw. Für den mittleren Abstand erhält man so:

$l \approx \frac{10^3 \cdot 1 + 10^3 \cdot 2 + ... + 10^3 \cdot 10^6}{10^9 - 1}$

Dies ergibt

$l \approx \frac{10^3 \cdot \frac{10^6}{2} \cdot (10^6+1)}{10^9} \approx \frac{10^6}{2} = 500.000.$

Die präzise Formel lautet

$l = \frac{n(n+z-2)}{2k(n-1)}$, was offenbar in der Nähe von $\frac{n}{2z}$ liegt; als genauerer Wert ergäbe sich hier ca. $500.000,5$.

**Lösung zu Aufgabe 41:**

In dem skalenfreien Graphen gilt offenbar $p(100) \approx 0,01$ und $p(200) \approx 0,0025$. In

dem Zufallsgraphen $G$ hat man allgemein $p_i = \binom{n-1}{i} p^i (1-p)^{n-1-i}$ (siehe Lemma 10.5-1), für die hier gegebenen Werte erhält man $p_{100} \approx 0,04$ und $p_{200} \approx 10^{-21}$.

**Lösung zu Aufgabe 42:**

Wir behaupten, dass $\lambda_c = \frac{1}{n-1}$ die epidemische Schranke ist.

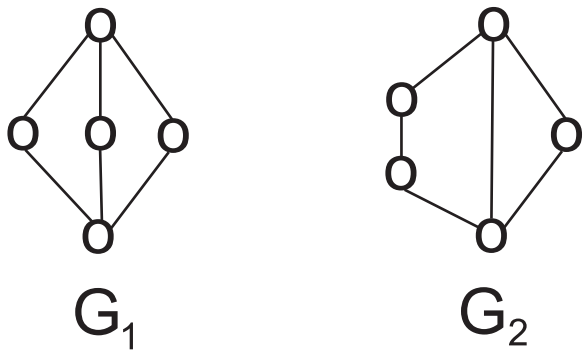
Gilt nämlich für die Ausbreitungsrate $\lambda \geq \frac{1}{n-1}$, so sind erwartungsgemäß im zweiten Zeittakt $\lambda \cdot (n-1) > 1$ viele Knoten infiziert (da der zu Anfang infizierte Knoten $n-1$ viele Nachbarn besitzt), im dritten Zeittakt $\lambda^2 \cdot (n-1)^2 > 1$ viele Knoten usw. Da der Graph endlich ist, kann langfristig ein stetiges "Flackern" erwartet werden, bei dem jeweils eine Hälfte der Knoten infiziert ist.
Gilt für die Ausbreitungsrate $\lambda < \frac{1}{n-1}$, so stirbt die Infektion wegen

$$\lambda^i \cdot (n-1)^i \to \infty$$

erwartungsgemäß langfristig ab.

**Lösung zu Aufgabe 43:**

Im untenstehenden Bild sind zwei Graphen mit dieser Gradfolge dargestellt – dabei gilt $s(G_1) = 36$ und $s(G_2) = 37$.



# Literaturverzeichnis

## Literaturverzeichnis zu Kapitel 1 - 7


[BUS] Busacker, R.G.; Saaty, T.L. *Endliche Graphen und Netzwerke* Oldenbourg,1968

[CHA] Chan, S. P. *Introductory topological analysis of el. networks Holt* Rinehart and Winston, 1969

[CHE] Chen, W.K. *Applied graph theory* North Holland, 1971

[DEO] Deo, N. *Graph theory with applications to engineering and computer science* Prentice Hall, 1974

[FOR] Ford, L.R.; Fulkerson, D.R. *Flows in networks* Princeton University Press, 1962

[GAR] Garey, M.R.; Johnson, D.S. *Computers and Intractability* W.H. Freeman, 1979

[GOU] Gould, R. *Graph theory* Benjamin/Cummings 1988

[HAR] Harary, F. *Graph theory* Adison-Wesley, 1969

[JUN] Jungnickel, D. *Graphen, Netzwerke und Algorithmen* B.I., 1987

[KOE] König, D. *Theorie der endlichen und unendlichen Graphen* Chelsea, 1950

[MAY] Mayeda, W. *Graph theory* Wiley, 1972

[ORE 1] Ore, O. *Theory of graphs* Am. Math. Soc., Coll. Pub., 1962

[ORE 2] Ore, O. *Graphs and their uses* Random House, 1963

[PAP] Papadimitriou, C.H.; Steiglitz, K. *Combinatorial Optimization* Prentice-Hall, 1982

[REA] Read, E. *Graph theory and computing* Academic Press, 1972

[SES] Seshu, S.; Reed, M. *Linear graphs and electrical networks* Addison Wesley, 1961

[SWA] Swamy, M.N.S.; Thulariseman, K. *Graphs, networks and algorithms* Wiley Interscience, 1981

[WIL] Wilson, R.J. *Introduction to graph theory* Academic Press, 1972

# Literaturverzeichnis zu Kapitel 8.1 - 8.4

[BER] Bertsekas, D.; Gallager, R. *Data networks* Prentice-Hall, 1987

[BEY] Berry, L. *Graph Theoretic Models for Multicast Communications* Computer Networks and ISDN Systems 20 (1990),95-99

[GAF] Gafni, E.M. ; Bertsekas, D.P. *Asymptotic optimality of shortest path routing* IEEE Trans. Inf. Theory, (1983)

[LIE] Liebl, F. *Routing-Algorithmen für paketvermittelnde Datennetze - Ein Beitrag zur Systematik* ntz Archiv Bd.9 (1987), S.317-325

[SCH] Schwartz, M. *Telecommunication networks* Addison-Wesley, 1987

# Literaturverzeichnis zu Kapitel 8.5 - 8.7

[BER] Bertsekas, D. P.; Gallager, R. *Data networks* Prentice-Hall, 1987

[CCI] *Signalling network functions and messages.* CCITT Recommendation Q. 704 Melbourne, 1988

[COM] Comer, D. E. *TCP/IP* mitp-Verlag, 2003

[DAV1] Davies, D. W.; Barber, D.L.A.; Price, W. L.; *Solomonides, C. M. Computer networks and their protocols* John Wiley & Sons, 1979

[DAV2] Davies, D. W.; Barber, D.L.A. *Communication networks for computers* John Wiley & Sons, 1973

[KLE] Klein, W. *Design and validation of routing plans for a large-scale common-channel signalling network* Proc. Int. Conf. on Communication Systems (Singapore 1988), 397 - 401

[POG1] Poguntke, W. *Graph problems related to common channel signalling* In: Proceedings on Twente Workshop on Graphs and Combinatorial Optimization, 1989, Universiteit Twente

[POG2] Poguntke, W. *On the design of communication networks using restricted message routing* Proceedings of the IEEE Int. Conf. on Communications, Genf 1993, 681 - 685

[SCH] Schwartz, M. *Telecommunication networks* Addison-Wesley, 1987

# Literaturverzeichnis zu Kapitel 9.1 - 9.3


[BOL] Bollobas, B. *Random graphs* Academic Press, 1985

[COL] Colboun, C.J. *The combinatorics of network reliability* Oxford University Press, 1987

[FEL] Feller, W. *An introduction to probability theory and its applications* Wiley, 1987

[FRA] Frank, H.;Frisch, I.T. *Communication transmission, and transportation networks* Addision-Wesley, 1971

[GAR] Garey, M.R.;Johnson, D.S. *Computers and intractability* W.H. Freeman, 1979

[GUM] Gumm, H.-P.;Poguntke, W. *Boolsche Algebra* BI-HTB, 1981

[PAP] Papoulis,A. *Probability, random variables, and stochastic processes* McGraw-Hill, 1984

[SCH1] Schneeweis, W.G. *Boolean functions with engineering applications and computer programs* Springer, 1989

[SCH2] Schneeweis, *W.G. Fehlertolerierende Rechensysteme* FernUniversität Hagen, Kurs Nr. 1750

[WEG] Wegener, I. *The complexity of Boolean functions* Wiley, 1987


# Literaturverzeichnis zu Kapitel 9.4 - 9.5


[BAL] Ball, M. O.; Provan, J. S. *Bounds on the reliability polynomial for shellable independence systems* SIAM J. on Alg. and Disc. Methods 3 (1982), 166-181

[BRO] Brooks, R. L.; Smith, C. A. B.; Stone, H. G.; Tutte, W. T. *The dissection of rectangles into squares* Duke Math. Journal 7 (1940), 312 - 340

[COL] Colbourn, C. J. *The combinatorics of network reliability* Oxford University Press, 1987

[DÖR] Dörfler, W.; Mühlbacher, J. *Graphentheorie für Informatiker* de Gruyter, 1973

[GAR] Garey, M.R.;Johnson, D.S. *Computers and intractability* W.H. Freeman, 1979

[KEL] Kel'mans, A. K. *Some problems of network reliability analysis* Automation and Remote Control 26 (1965), 564 - 573

[KIR] Kirchhoff, G. *Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der Verteilung galvanischer Ströme geführt wird* Ann. Phys. Chem. 72 (1847), 497 - 508

[PIE] Piekarski, M. *Listing of all possible trees of a linear graph* IEEE Trans. Circuit Theory 12 (1965), 124 - 125

[PRO] Provan, J. S.; Ball, M. O. *The complexity of counting cuts and of computing the probability that a graph is connected* SIAM J. on Computing 12 (1983), 777-788

[SPE] Sperner, E.; *Ein Satz über Untermengen einer endlichen Menge* Mathematische Zeitschrift 27 (1928), 544 - 548

[VAN] Van Slyke, R. M.; Frank, H. *Network reliability analysis: part I* Networks 1 (1972), 279 - 290

## Literaturverzeichnis zu Kapitel 9.6 - 9.7

[AWE] Awerbuch, B; Even, S. *Reliable broadcast protocols in unreliable networks* Networks 16 (1986), 381-396

[BAU] Bauer, D.; Boesch, F.; Suffel, C.; Tindell, R. *Combinatorial optimization problems in the analysis and design of probabilistic networks* Networks 15 (1985), 257 - 271

[BOE1] Boesch, F. T. *On unreliability polynomials and graph connectivity in reliable network synthesis* Journal of Graph Theory 10 (1986), 339 - 352

[BOE2] Boesch, F. T.; Satyanarayana, A.; Suffel, C. L. *Least reliable networks and the reliability domination* IEEE Trans. on Comm. 38 (19301, 2004 - 2009

[BOF] Boffey, T. B. *Computer network design problems* Preprint, University of Liverpool, 1991

[COL] Colbourn, C. J. *The combinatorics of network reliability* Oxford University Press, 1987

[GAF] Gafni, E. M.; Bertsekas, D. P. *Distributed algorithms for generating loop-free routes in networks with frequently changing topology* IEEE Trans. on Comm. 29 (1981), 11 - 18

[HU] Hu, T. C. *Optimum communication spanning trees SIAM* J. Comput. 3 (1974), 188 - 195

[LOM] Lomonosov, M. V.; Poleskii, *V. P. Lower bound of network reliability Problems of Information* Transmission 8 (1972), 118 - 123

[MAN] Manoussakis, J.; *Tuza, Z. Optimal routings in communication networks with linearly bounded capacity* Rapport de Recherche No. 597, Universite de Paris-Sud, 1990

[MIN] Minoux, M. *Network synethesis and optimum network design problems* Networks 19 (1989), 313 - 360

[POG1] Poguntke, W. *Static routing tables with multiple entries: a graph-theoretic study* Manuskript, FernUniversität Hagen, 1990

[POG2] Poguntke, W. *On the design of communication networks using restricted message routing* Proceedings of the IEEE Int. Conf. on Communications, Genf 1993, 681 - 685

# Literaturverzeichnis zu Kapitel 10.1 - 10.4

[Ada99] L. A. Adamic: *The Small World web. LNCS*, vol. 1696, pp. 443-454, Springer, New York (1999)

[Ama00] L. A. N. Amaral, A. Scala, M. Barthelemy, *H. E. Stanley: Classes of small-world networks*. Proc. Natl. Acad. Sci. USA 97, pp. 11149-11152 (2000)

[Bar00] A. Barrat, M. Weigt: *On the properties of small world network models*. Eur. Phys. J. B 13, pp. 547-560 (2000)

[Bara99] A.-L. Barabasi, R. Albert, H. Jeong: I. Nature Vol. 401, pp. 130-131 (1999)

[Bara99-2] A.-L. Barabasi, R. Albert: *Emergence of scaling in random networks*. SCIENCE, Vol. 286, pp. 509-512 (1999)

[Bara00] A.-L. Barabasi, R. Albert, H. Jeong: *Scale-free characteristics of random networks: The topology of the World Wide Web*. Physica A 281, pp. 69-77 (2000)

[Barr01] L. Barriere, P. Fraigniaud, E. Kranakis, D. Krizanc: *Efficient routing in networks with long range contacts*. Proc. 15th Int. Symp. on Dist. Computing, LNCS 2180, pp. 270-284 (2001)

[Ber00] N. Berger, C. Borgs, J. T. Chayes, A. Saberi: *On the spread of viruses on the internet*. Proc. 14th ACM-SIAM Symp. on Discrete Algorithms (SODA), 301-310 (2005)

[Ber03] N. Berger, B. Bollobas, C. Borgs, J. Chayes, O. Riordan: *Degree distribution of the FKP network model*. Proc. 30th Int. Coll. on Automata, Languages and Programming, LNCS 2719, pp. 725-738 (2003)

[Bol85] B. Bollobas: *Random graphs*. Academic Press (1985)

[Bol02] B. Bollobas, O. Riordan, J. Spencer, G. Tusnady: *The degree sequence of a scale-free random graph process. Random Structures and Algorithms*, vol.18(3), pp.279-290 (2001)

[Buc02] M. Buchanan: *Small Worlds - Das Universum ist zu klein für Zufälle*. Campus-Verlag (2002)

[Coh03] R. Cohen, S. Havlin: *Scale-free networks are ultrasmall*. arXiv:cond-mat/0205476 v2 (2003)

[Dog01] S. N. Dogorovtsev, J. F. F. Mendes: *Evolution of networks*. arXiv:cond-mat/0106144 v2 (2001)

[Fal99] M. Faloutsos, P. Faloutsos, C. Faloutsos: *On power-law relationships of the internet technology. Computer Communication Review*, p. 251 (1999)

[Fro04] A. Fronczak, P. Fronczak, J. A. *Holyst: Average path length in random networks*. arXiv:cond-mat/0212230 v3 (2004)

[Hay00] B. Hayes: *Graph Theory in Practice: Part II. American Scientist*, Vol. 88, Number 2, pp. 104-109, (2000)

[Kum00] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, E.Upfal: *The web as a graph. Proc. 19th ACM SIGACT-SIGMOD-AIGART Symp.* Principles of Database Systems (2000)

[New99] M. E. Newman, D. Watts: *Renormalization group analysis of the small-world network model*. Phys. Lett. A 263, pp. 341-346 (1999)

[New03] M. E. Newman: *The structure and function of complex networks*. arXiv:cond-mat/0303516 v1 (2003)

[Kle00] J. Kleinberg: *The small-world phenomenon: an algorithmic perspective. In: Proc. of the 32nd Annual ACM* Symposium on Theory of Computing, pp. 163-170 (2000)

[Wat99] D. J. Watts: *Small worlds*. Princeton University Press (1999)

[Wat98] D. J. Watts - S. H. Strogatz: *Collective dynamics of small world networks*. Nature 393, pp. 440-442 (1998)

## Literaturverzeichnis zu Kapitel 10.5 - 10.7

[Ada99] L. A. Adamic: *The Small World web. LNCS*, vol. 1696, pp. 443-454, Springer, New York (1999)

[Aie02] W. Aiello, F. Chung, L. Lu: *Random evolution in massive graphs*. In: J. Abello et al.: Handbook of Massive Data Sets, pp. 97-122, Kluwer (2002

[Ama00] L. A. N. Amaral, A. Scala, M. Barthelemy, *H. E. Stanley: Classes of small-world networks*. Proc. Natl. Acad. Sci. USA 97, pp. 11149-11152 (2000)

[Bai75] N. T. J. Bailey: *The mathematical theory of infectious diseases*. Griffin (1975)

[Bara99] A.-L. Barabasi, R. Albert, H. Jeong: I. Nature Vol. 401, pp. 130-131 (1999)

[Bara99-2] A.-L. Barabasi, R. Albert: *Emergence of scaling in random networks*. SCIENCE, Vol. 286, pp. 509-512 (1999)

[Bara00] A.-L. Barabasi, R. Albert, H. Jeong: *Scale-free characteristics of random networks: The topology of the World Wide Web*. Physica A 281, pp. 69-77 (2000)

[Bar00] A. Barrat, M. Weigt: *On the properties of small world network models*. Eur. Phys. J. B 13, pp. 547-560 (2000)

[Barr01] L. Barriere, P. Fraigniaud, E. Kranakis, D. Krizanc: *Efficient routing in networks with long range contacts*. Proc. 15th Int. Symp. on Dist. Computing, LNCS 2180, pp. 270-284 (2001)

[Ber05] N. Berger, C. Borgs, J. T. Chayes, A. Saberi: *On the spread of viruses on the internet*. Proc. 14th ACM-SIAM Symp. on Discrete Algorithms (SODA), 301-310 (2005)

[Bol85] B. Bollobas: *Random graphs*. Academic Press (1985)

[Bol01] B. Bollobas, O. Riordan, J. Spencer, G. Tusnady: *The degree sequence of a scale-free random graph process*. Random Structures and Algorithms 18, pp.279-290 (2001)

[Bol04] B. Bollobas, O. Riordan: *The diameter of a scale-free random graph*. Combinatorica 24(1), pp. 5-34 (2004)

[Bro00] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, J. Wiener: *Graph structure in the web*. Computer Networks 33, 309-320 (2000)

[Buc02] M. Buchanan: *Small Worlds - Das Universum ist zu klein für Zufälle*. Campus-Verlag (2002)

[Coh03] R. Cohen, S. Havlin: *Scale-free networks are ultrasmall*. arXiv:cond-mat/0205476 v2 (2003)

[Dog01] S. N. Dogorovtsev, J. F. F. Mendes: *Evolution of networks*. arXiv:cond-mat/0106144 v2 (2001)

[Fal99] M. Faloutsos, P. Faloutsos, C. Faloutsos: *On power-law relationships of the internet technology. Computer Communication Review*, p. 251 (1999)

[Fro04] A. Fronczak, P. Fronczak, J. A. *Holyst: Average path length in random networks*. arXiv:cond-mat/0212230 v3 (2004)

[Hay00] B. Hayes: *Graph Theory in Practice: Part II. American Scientist*, Vol. 88, Number 2, pp. 104-109, (2000)

[Kle99] J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins: *The web as a graph: measurements, models, and methods*. In: Proc. of the Int. Conf. on Combinatorics and Computing, no. 1627 LNCS, pp. 1-18 (1999)

[Kle00] J. Kleinberg: *The small-world phenomenon: an algorithmic perspective*. In: Proc. of the 32nd Annual ACM Symposium on Theory of Computing, pp. 163-170 (2000)

[Kum00] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, E.Upfal: *The web as a graph. Proc. 19th ACM SIGACT-SIGMOD-AIGART Symp.* Principles of Database Systems (2000)

[Li05] L. Li, D. Alderson, R. Tanaka, J. C. Doyle, W. Willinger: Towards a theory of scale-free graphs. \newline arXiv:cond-mat/0501169 v2 (2005)

[Mar99] J. Marro, R. Dickman: *Nonequilibrium phase transitions in lattice models*. Cambridge University Press (1999)

[Mur93] J. D. Murray: I. Springer (1993)

[New99] M. E. Newman, D. Watts: *Renormalization group analysis of the small-world network model*. Phys. Lett. A 263, pp. 341-346 (1999)

[New03] M. E.Newman: *The structure and function of complex networks*. arXiv:cond-mat/0303516 v1 (2003)

[Pas01] R. Pastor-Satorras, A. Vespignani: *Epidemic spreading in scale-free networks*. Phys. Review Letters, Vol. 86, No. 14, pp.3200-3203 (2001)

[Wat99] D. J. Watts: *Small worlds*. Princeton University Press (1999)

[Wat98] D. J. Watts - S. H. Strogatz: *Collective dynamics of small world networks*. Nature 393, pp. 440-442 (1998)

# Literaturverzeichnis zu Kapitel 11

[Bol98] B. Bollobas: *Modern Graph Theory.* Springer (1998)

[Bur05] R. Burioni, D. Cassi: *Random walks on graphs: ideas, techniques and results.* J. Phys. A: Mathe. Gen. 38 (2005), R45-R78

[Cha89] A. K. Chandra, P. Raghavan, W. L. Ruzzo, R. Smolensky, P. Tiwari: *The electrical resistance of a graph captures its commute and cover times.* In: Proc. 21st Ann. ACM Symp. Theor. Comput., Seattle (1989), pp. 574-586

[Doy84] P. G. Doyle, J. L. Snell: *Random walks and electrical networks*. The Mathematical Association of America (1984)

[Pol21] G. Polya: Über eine Aufgabe der Wahrscheinlichkeitsrechnung betreffend die Irrfahrt im Straßennetz. Math. Annalen 84 (1921), pp. 149-160

# Index

**Symbole**

$(n, s) - Ring$ 289

0-1-Fluss 125

$c - C_c$-optimal 288

**A**

Abstand 19, 97

Adjazenzmatrix 39, 74

äquivalent 254

äquivalente Boolesche Ausdrücke 385

Äquivalenz 254

Äquivalenzklassen 360

Äquivalenzrelation 360

äußerer Graph 11

Algorithmus 48, 51

Algorithmus ”Gerüst mit Breadth first search” 115

Algorithmus von Dijkstra 102

Algorithmus von Floyd-Warshall 105

Algorithmus "Azyklisch" 72

Algorithmus "Baumtest" 49

Algorithmus "Bipartit" 58

Algorithmus ”Breadth first search” 97

Algorithmus ”Gerüst mit Depth first search” 115

Algorithmus von Bellman-Ford 173

Algorithmus von Kruskal 119

Algorithmus von Prim 118

Algorithmus von Dijkstra 172

Algorithmus von Edmonds und Karp 137

Algorithmus von Floyd-Warshall 170

Algorithmus "Knotenüberdeckung" Nr. 1 60

Algorithmus "Knotenüberdeckung" Nr. 2 61

Algorithmus "Konstruktion eines Gerüsts" 50

Algorithmus "Planar" 59

Anfangsecke, Endecke 66

Anfangsecke, Endecke einer Kantenfolge 12

Anzahl aller zusammenhängenden Graphen 239

approximativer Algorithmus 61

Assoziativgesetz 365

Außengrad 67

Axiome der Booleschen Algebra 385

azyklisch 70

**B**

Bögen 65

Bahn 69

Ball-Provan-Schranken 270

Basis 363

Baum kürzester Wege 101

Baum kürzester Wege 168

Bellmansche Gleichungen 101, 178

benachbarte Ecken 3

Bewertung 84

Bezugsecke 42, 77

Biasfaktor 189

bibipartite unabhängige Menge 256

bidirektional 223

Bidirektionalität 213, 223

bijektiv 360

Bild 364

binäres Suchen 16

bipartiter (paarer) Pseudograph 21
Blätter zu einer Brücke 22
Blockschaltbild Funktionsbaum Fehlerbaum 388
Bogenbewertung 86
Bogenfolge 68
Bogenzug 68
Boolesche Variable 384
Boolesche Methode 247
Brücke 22
branch-and-bound-Methode 62
Broadcasting 194
BUM-Problem 256

**C**

Cluster-Koeffizient 297

**D**

Darstellung durch Listen 39
Datagramm-Modus 165
Datenstrukturen 53
DDNF 386
de Morgansche Regeln 386
Determinante 366
dezentraler Algorithmus 304
Digraph 66
Dimension 363
disjunkte Mengen 359
disjunkte Vereinigung 359
Disjunktive Normalform DNF 386
Durchmesser 294
Durchschnitt 359
Durchschnitt und Vereinigung von Pseudographen 9

**E**

echte Teilmenge 359
Ecken 65
Eckenbewertung 86
effektive Widerstand 346
effektiver Leitwert 346
effizienter Algorithmus 56
Effizienz 53
Eindeutigkeits-Prinzip 337
Einschränkung einer Abbildung 360
einseitig verbindbare Ecke 69
Einzelfall 51
elektrisches Netzwerk 344
Element 359
Endecke 19
Endecken einer Kante 1
Endkante 19
Entscheidungsproblem 256
erbliches Mengensystem 264
Erkennungsalgorithmus 252
erwartete Zustellzeit 305
Eulersche Polyederformel 36
Eulerscher Kantenzug 33

**F**

Fernkontakt-Graphen 307
festes Routing 213
flooding 169
Fluchtwahrscheinlichkeit 346
Fluss 122
Fluss-Funktion 157
Fluss-Netz 122
Flussanforderungen 158

Flussdiagramm 48
Folgenummern 196
fundamentaler Schnitt 26
funktionsfähiger Zustand 242

**G**

Gebiete 35
geordnetes n-Tupel 360
Gerüst 17
gerichteter Baum 73
geschlossene Bogenfolge 68
geschlossene, offene Kantenfolge 13
Größenordnung 55
Grad 67
Grad einer Ecke 3
Graphenzerlegungsmethode 248

**H**

Hamiltonscher Pseudograph 33
harmonische Funktionen 335
hartes Problem 57
Heiratssatz 147
Heuristik 61
heuristisches Verfahren 61
homöomorph 37

**I**

Implementierung 49
Indizierung 365
induzierter Teilgraph 8
induzierter Untergraph 7
Innengrad 67
innerer Pseudograph 11
Inzidenz 66
Inzidenz 1
Inzidenzliste 39
Inzidenzmatrix 40, 75
Iteration 52

**K**

Körper $\mathbb{F}_2$ 362
kürzester Weg 96
kanonische DNF 386
Kantenbewertung 86
Kantenliste 39
Kantenzusammenhangszahl 289
Kapazität eines Schnittes 127
Kern 364
Kirchhoffsche Gesetze 91
Komplement 359
Komplexität 55
Komponenten 70
Korrektheit 51
Korrespondenz 143
Kosten 156
Kostenfunktion 156
Kreis-Bogen-Matrix 80
Kreis-Kanten-Matrix 44
kreisloser Pseudograph 15
kritische Kantenmenge 249
kritischer Weg 111
Kruskal-Katona-Schranken 270

**L**

Länge einer Bogenfolge 68
Länge einer Kante 96
Länge einer Kantenfolge 13, 96
längster Weg 110
Lastteilung 167, 215
leichtes Problem 57
leitungsvermittelte, paketvermittelte Kommunikation 165

Leitwert 344
linear unabhängig 363
lineare Abbildung 364
Linearkombination 362

**M**

Markierungsalgorithmus von Ford und Fulkerson 132
Markov-Kette 341
massive Graphen 322
Matrix über einem Körper 365
maximale Korrespondenz 143
maximale unabhängige Menge 254
maximaler Fluss 124
Maximum-Prinzip 337
Meldung 194
Mengendifferenz 359
Methode der Relaxationen 342
mimimales Gerüst 112
minimale Knotenüberdeckung 59, 254
minimale kritische Menge 277
minimaler Schnitt 127
minimales Netz 158
mittleren Abstand 294
Monte-Carlo-Methode 343
Multidigraph 66
Multiplikation von Matrizen 365

**N**

n-faches kartesisches Produkt 360
n-regulärer Pseudograph 4
Nährungslösung 61
Nachfolger 66
Nachricht 194
Netz 96, 170, 277
NP-vollständig 57, 255

**O**

offene Bogenfolge 68
optimal 287
orientierter Kreis 78
orientierter Schnitt 79
Orientierung 78
orthogonal 364
OSI-Schichtenmodell 164
Oszillation 185

**P**

Paarung 143
planare Darstellung 34
planarer Graph 34
polynomialer Algorithmus 56
Potentialdifferenz 90
Potenz 365
Potenzmenge 359
Primärmeldung 164
Prinzip der Inklusion – Exklusion 388
Problem 51
Problemgröße 55
Produkt 365
Programm 49
Protokoll 164
Pseudodigraph 65

**Q**

quadratische Matrix 365
Quasi-Hochsprache 49

**R**

Rückkopplungseffekt 189
random routing 169
Rang 366
Rangabfall 24

Rayleigh's Monotoniegesetz 351
reduzierbar 254
Reduzierbarkeit 254
reduzierte Inzidenzmatrix 42, 77
Regel für die doppelte Verneinung 386
Regelungs- und Steuerungstheorie 189
reguläre Matrix 367
Relaxationen 342
Routing-Informationen 193
Routing-Plan 216
Routing-Problem 110, 164
Routing-Tabellen 211, 216
Rundspruch 194

**S**

s,t-Zuverlässigkeitsproblem 252
Satz von Ford und Fulkerson 133
Satz von Hall 146
Satz von König und Egerváry 144
Satz von Kuratowski 37
Schicht 164
Schleife 52
Schnitt eines Fluss-Netzes 127
Schnitt-Bogen-Matrix 80
Schnitt-Kanten-Matrix 45
schwieriges Problem 256
Siebformel 388
singuläre Matrix 367
skalare Multiplikation 362
Skalarprodukt 363
Small-World-Graphen 298
Small-World-Phänomen 292
SP-Routing 221
SP-Routing mit Präferenz 222
Spaltenrang 366
spannender Baum 17
Sperner-Schranken 266
Sprungbefehl 52
stark verbindbare Ecken 69
stark zusammenhängend 70
starke Komponenten 70
statisches, adaptives Routing 168
Summe von Pseudographen über einem Graphen 11
Summe von zwei Graphen 10
symmetrische Differenz 362
Synchronität 179

**T**

Teilgraph-Zählmethode 247
Teilmenge 359
topologische Anordnung 71
Träger 384
transfer prohibited procedure 215
Transponierte 366
trennende Eckenmenge 141
trennende Kantenmenge 140, 256
Turing-Maschine 57

**U**

Unterteilung 37
Untervektorraum 362
unzugängliches Problem 57

**V**

Variablenbelegungen 384
Vektorraum der Bogenbewertungen 86
Vektorraum der Eckenbewertungen 86
Vektorraum 362
verbindbare Ecken 14, 69

verbindungslose, verbindungsorientierte Datenübermittlung 165
Vereinigung 359
Verifikation 52
verteilter asynchroner Bellman-Ford Algorithmus 179
Verzweigung 52
virtuelle Verbindung 165
volle Zuverlässigkeit 243
volles Zuverlässigkeitsproblem 252
vollständige Korrespondenz 145
vollständiger Graph 2
Vorgänger 66

**W**

Wachstumsrate 55
Wahrheitstafel 384
Wegeauswahl 164
Wert eines Flusses 124
Widerstand 344
Wurzel 73

**Z**

Zählproblem 256
Zeichengabesystem Nr. 7 211
Zeilenrang 366
zentrale Ecke 107
zentrales, verteiltes Routing 168
Zirkulation 87
Zufallsgraph 234
Zufallskreis 301
zulässige Zirkulation 125, 150
zulässiger Fluss 149
zulässiges Netz 158
zunehmender Weg 128
Zusammenhang 14
Zusammenhangskomponenten 15
Zustand 242
Zuverlässigkeit 242
Zuverlässigkeitspolynom 246
Zuweisungen 52
Zyklus 69