5

Modellierung mit Graphen

Der Kalkül der Graphen eignet sich in vielen Aspekten außerordentlich gut zum Modellieren von Aufgaben und Systemen. Modelle beschreiben meist Objekte und Beziehungen zwischen ihnen. Genau das leistet ein Graph als Abstraktion aus Knoten und Kanten: Die Knoten repräsentieren eine Menge gleichartiger Objekte; jede der Kanten repräsentiert eine Beziehung zwischen je zwei Objekten, die sie verbindet. Solche Graphen sind also 2-stellige Relationen über der Knotenmenge.

Als Graphen formulierte Modelle sind leicht verständlich und haben anschauliche Darstellungen. Als Beispiel zeigt Abb. 5.1 einen Graphen, der Autobahnverbindungen in der Umgebung von Paderborn modelliert. Die Knoten repräsentieren Städte und jede Kante die Existenz einer Autobahn zwischen zwei Städten. Die Kanten dieses Graphen sind ungerichtet, um auszudrücken, dass die Autobahnen in beiden Richtungen befahren werden können. In dieser Hinsicht unterscheidet sich der Graph in Abb. 5.2. Er modelliert Abläufe von Telefongesprächen. Seine Knoten repräsentieren Zustände und die Kanten repräsentieren Aktionen, die von einem Zustand in einen anderen oder in denselben überführen. Hier müssen die Kanten natürlich eine Richtung haben.

Durch Graphen formulierte Modelle sind mathematisch präzise, da sie auf Relationen basieren. Es können daraus viele tief gehende Eigenschaften formal abgeleitet und zur Analyse des Modells genutzt werden: So kann man z. B. mit dem Begriff des Weges ausdrükken, dass es in Abb. 5.1 Autobahnverbindungen von Paderborn nach Unna über mehrere Städte hinweg gibt. Auch kann man leicht erkennen, dass es in diesem Graphen keinen Rundweg gibt, bei dem jedes Autobahnteilstück genau einmal befahren wird. Gäbe es einen solchen Rundweg, dann würde er jede Stadt genauso oft verlassen, wie er sie erreicht. Das ist aber nicht möglich, da in Bielefeld und Wünnenberg jeweils eine ungerade Anzahl von Autobahnverbindungen existieren. In dem Ablaufgraphen von Abb. 5.2 kann man z. B. erkennen, dass von jedem Knoten alle Knoten erreichbar sind. Dies ist eine typische und wichtige Eigenschaft von Graphen, die zyklische Abläufe modellieren.

Graphen können auch sehr systematisch als Datenstrukturen implementiert werden. Es gibt einen großen Fundus an Algorithmen, die Berechnungen auf Graphen durchführen.

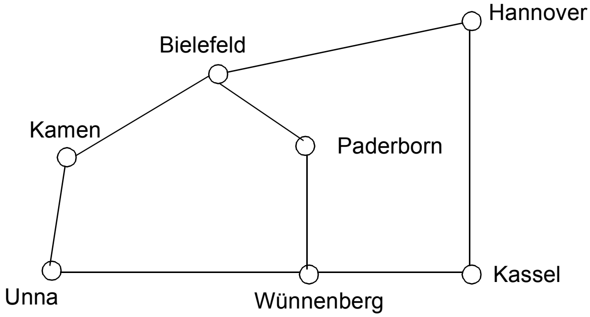


Abbildung 5.1: Ungerichteter Graph modelliert Autobahnverbindungen

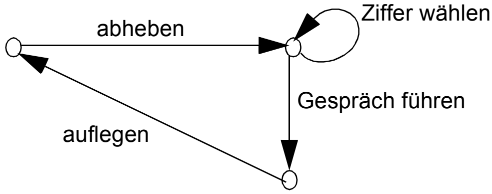


Abbildung 5.2: Gerichteter Graph modelliert Ablauf von Telefongesprächen

Wir führen zunächst die Grundbegriffe des Graphenkalküls in Abschnitt 5.1 ein. Dann zeigen wir, dass Graphen außerordentlich vielfältig zur Modellierung unterschiedlicher Themen eingesetzt werden können: In Abschnitt 5.2 zeigen wir die Modellierung von Wegeproblemen. Die Frage, welche Objekte des Modells miteinander direkt oder indirekt verbunden sind, untersuchen wir in Abschnitt 5.3. Bäume sind eine spezielle Art von Graphen, mit denen man z. B. geschachtelte Strukturen oder Folgen von Entscheidungen modellieren kann (Abschnitt 5.4). In Abschnitt 5.5 betrachten wir Zuordnungsprobleme. So kann man z. B. den Staaten auf einer Landkarte (Knoten des Graphen) Farben so zuordnen, dass zwei Staaten, die eine gemeinsame Grenze haben (repräsentiert durch eine Kante) verschieden gefärbt werden. Schließlich zeigen wir in Abschnitt 5.6, wie gerichtete Graphen eingesetzt werden, um Abhängigkeiten, Anordnungen und Abläufe zu modellieren. Aufbauend auf den Begriffen aus Abschnitt 4.1 führen wir weitere Begriffe des Graphenkalküls mit den Themen ein, für die sie benötigt werden.

5.1 Grundlegende Definitionen

In diesem Abschnitt führen wir grundlegende Begriffe des Graphenkalküls ein, die notwendig sind, um Graphen als Modelle anzugeben. Weitergehende Begriffe zur Formulierung von Eigenschaften von Graphen definieren wir in den nachfolgenden Abschnitten, dort wo das Modellierungsthema diese Eigenschaft benötigt. Außerdem stellen wir hier die wichtigsten Verfahren vor, um Graphen zu repräsentieren.

Definition 5.1: Gerichteter Graph

*Ein **gerichteter Graph** ist ein Paar $G = (V, E)$ mit einer endlichen Menge von Knoten V und einer Menge von Kanten $E \subseteq V \times V$.* ■

Die Adjektive *gerichtet* oder *ungerichtet* lassen wir häufig weg, wenn die Eigenschaft aus dem Kontext klar oder die Unterscheidung unwichtig ist. Die Kantenmenge E ist also eine 2-stellige Relation über V. Als Beispiel geben wir einen gerichteten Graphen $G_1 = (V_1, E_1)$ an:

V_1 = {a, b, c, d}

E_1 = {(a, b,), (a, c), (a, d), (b, b), (b, c), (d, b), (d, c)}

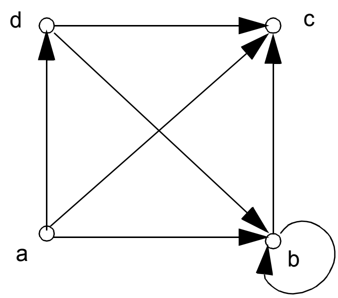


Abbildung 5.3: Gerichteter Graph G_1

Abb. 5.3 zeigt den Graphen G_1 in grafischer Darstellung: Knoten werden als benannte Punkte und Kanten als Pfeile dargestellt. Eine einzelne gerichtete Kante wird wie in der Relation durch *(u, v)* oder in Anlehnung an die Grafik durch $u \rightarrow v$ notiert. In der englischen Terminologie heißt ein Knoten *vertex* oder *node* und eine Kante *edge* oder *arc*, wenn betont werden soll, dass sie gerichtet ist.

Kanten *(v, v)*, die im selben Knoten münden, von dem sie ausgehen, heißen *Schleife* oder *Schlinge*. Der Graph G_1 enthält die Schleife *(b, b)*. Für manche Überlegungen muss gefordert werden, dass ein Graph schleifenfrei ist, also keine Schleife enthält.

Die Definition 5.1 fordert, dass die Menge der Knoten endlich ist – und dadurch auch die Menge der Kanten. Das schränkt den Graphenbegriff zwar ein, ist für unsere Zwecke aber ausreichend.

Da die Kanten eines Graphen als Menge definiert sind, kann es zwischen zwei Knoten *u* und *v* nur eine Kante *(u, v)* geben. Für manche Modellierungen ist das zu einschränkend, z. B. wenn in Abb. 5.2 eine Kante *„Daten übertragen“* ergänzt werden sollte, die dieselben Knoten wie die Kante *„Gespräch führen“* verbindet. In der Grafik wäre das einfach anzugeben, als Kantenmenge könnte es jedoch nicht formuliert werden. Wir zeigen später, wie man trotzdem die Anzahl der Kanten zwischen zwei Knoten angeben kann.

Schließlich sind die Graphen auch dadurch eingeschränkt, dass eine Kante nur zwei Knoten miteinander verbinden kann. Anwendungen wie die Modellierung von Leitungssystemen lassen sich so nicht unmittelbar angeben; man müsste zusätzliche Knoten einfügen, um komplexe Kanten zu gliedern. Solche Beschränkungen sind ein Preis für die Einfachheit des Kalküls.

Mit gerichteten Graphen können wir prinzipiell auch Beziehungen zwischen Objekten modellieren, die nicht in einer Richtung orientiert sind. So sind die Autobahnverbindungen in Abb. 5.1 in beiden Richtungen nutzbar. Wollten wir sie mit einem gerichteten Graphen modellieren, so müssten wir jede Verbindung durch zwei entgegengesetzt orientierte Kanten repräsentieren, z. B. *(Paderborn, Bielefeld)* und *(Bielefeld, Paderborn)*. Die Kantenmenge eines solchen gerichteten Graphen wäre eine symmetrische Relation: aus $(a, b) \in E$ folgt $(b, a) \in E$. Die Tatsache, dass die Richtung der Verbindung zweier Knoten nicht festgelegt ist, kann auch direkt durch ungerichtete Kanten ausgedrückt werden. Man fasst in der symmetrischen Kantenrelation alle Paare $(x, y), (y, x)$ zu einer ungerichteten Kante $\{x, y\}$ zusammen. $\{x, y\}$ ist die Menge der Knoten, die diese Kante verbindet, im Gegensatz zu den Elementen der Paare sind die Elemente der Menge nicht geordnet.

In der Graphentheorie und in der Modellierung haben ungerichtete und gerichtete Graphen jeweils eigenständige Bedeutung. Deshalb werden ungerichtete Graphen direkt definiert. Die obigen Überlegungen zur Herleitung aus gerichteten Graphen soll nur den Zusammenhang zwischen beiden Graphenarten verdeutlichen.

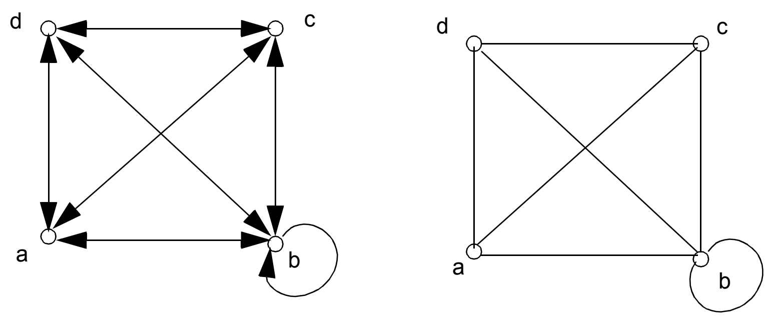


Abbildung 5.4: Gerichteter Graph mit symmetrischer Kantenrelation modelliert ungerichteten Graphen

Abb. 5.4 zeigt einen gerichteten Graphen mit symmetrischer Kantenrelation und den zugehörigen ungerichteten Graphen.

Definition 5.2: Ungerichteter Graph

*Ein **ungerichteter Graph** ist ein Paar $G = (V, E)$ mit einer endlichen Menge von Knoten V und einer Menge E von ungerichteten Kanten:*
$E \subseteq \{ \{x, y\} \mid x, y \in V\}$ ■

Der ungerichtete Graph aus Abb. 5.4 wird dann als Paar von Mengen wie folgt angegeben:

V = {a, b, c, d}

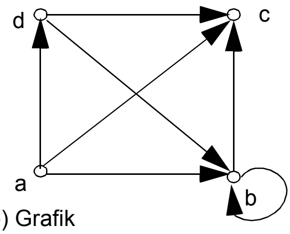
E = { {a, b,}, {a, c}, {a, d}, {b}, {b, c}, {d, b}, {d, c} }

Man beachte, dass eine Schleife im ungerichteten Graphen als 1-elementige Menge angegeben wird, z. B. *{b}*. Alle Kanten, die nicht Schleifen sind, werden durch 2-elementige Mengen angegeben. In der symmetrischen Kantenrelation des zugehörigen gerichteten Graphen wird eine Schleife durch eine einzige Kante, wie *(b, b)*, angegeben; alle anderen Kanten treten in Paaren *(a, b), (b, a)* auf.

Wir wollen nun die vier wichtigsten Arten der Darstellung von Graphen vorstellen.

V = {a, b, c, d}
E = { (a, b), (a, c), (a, d),
(b, b), (b, c),
(d, b), (d, c)}

a) Mengen V und E



b) Grafik

	a	b	c	d
a	f	w	w	w
b	f	w	w	f
c	f	f	f	f
d	f	w	w	f

c) Adjazenzmatrix AM

a	(b, c, d)
b	(b, c)
c	( )
d	(b, c)

d) Adjazenzlisten

Abbildung 5.5: Darstellung von gerichteten Graphen

In Abb. 5.5 haben wir sie für den Graphen G_1 zusammengefasst: Die *abstrakte Angabe der Mengen V und E* (Abb. 5.5a) ist in den Definitionen 5.1 und 5.2 eingeführt worden. Die *grafische Darstellung* mit Pfeilen für gerichtete und Linien für ungerichtete Kanten (Abb. 5.5b) wird wegen ihrer Anschaulichkeit für den menschlichen Leser am häufigsten verwendet. Hinzu kommen zwei Darstellungsarten, die insbesondere als Datenstrukturen für Algorithmen auf Graphen benutzt werden. Sie setzen voraus, dass die Knotenmenge *V* als Indexmenge verwendet und ihre Elemente linear geordnet werden können. Eine Adjazenzmatrix *AM* (Abb. 5.5 c) ist eine quadratische Matrix, deren Zeilen und Spalten jeweils mit Knoten indiziert werden. Die Matrixelemente sind Wahrheitswerte *w* und *f*. Es gilt

AM [i, j] = ((i, j) $\in$ E).

Jedes *w* in der Matrix steht für eine gerichtete Kante. Will man einen ungerichteter Graphen dargestellen, so muss er durch einen gerichteten mit symmetrischer Kantenrelation repräsentiert werden. Die Matrix-Darstellung erlaubt, durch direkten Zugriff auf *AM [i, j]* zu entscheiden, ob *E* die Kante *(i, j)* enthält. Außerdem kennzeichnen die Spaltenindizes der *w*-Elemente in der *i*-ten Zeile, zu welchen Knoten vom Knoten *i* eine Kante führt. Entsprechend geben die Zeilenindizes der *w*-Elemente in der *j-ten* Spalte an, von welchen Knoten eine Kante in den Knoten *j* mündet. Allerdings benötigt diese Darstellung immer Speicherplatz für n^2 Wahrheitswerte, wenn *n* die Anzahl der Knoten des Graphen ist.

Die *Adjazenzlisten* (Abb. 5.5d) repräsentieren Graphen kompakter als die Matrixdarstellung: Zu jedem Knoten *i* gibt es eine Folge (Liste) von Knoten, zu denen von *i* ausgehend eine Kante führt. Diese Darstellung benötigt gerade so viele Listenelemente, wie *E* Kanten enthält. Die Menge der Knoten, zu denen von Knoten *i* eine Kante führt, kann man leicht aufzählen, indem man die *i*-te Folge durchläuft. Allerdings muss man auch die *i*-te Folge durchlaufen, wenn man feststellen will, ob *E* die Kante *(i, j)* enthält. Das Aufzählen aller Knoten, von denen eine Kante in den Knoten *j* mündet, erfordert sogar, dass alle Folgen nach dem Element *j* durchsucht werden. Soll ein ungerichteter Graph dargestellt werden, so muss wie im Falle der Adjazenzmatrix eine symmetrische Relation von gerichteten Kanten angegeben werden.

Da wir häufig Graphen zerlegen und die Teile separat betrachten, führen wir den Begriff des Teilgraphen formal ein:

Definition 5.3: Teilgraph

*Der Graph G' = (V', E') ist ein **Teilgraph** des Graphen G = (V, E), wenn gilt V' $\subseteq$ V und E' $\subseteq$ E. G' heißt **durch V' induzierter Teilgraph von G**, wenn E' alle Kanten aus E enthält, deren beide Enden in V' liegen. G und G' sind entweder beide gerichtet oder beide ungerichtet.* ■

Sei *G* der Graph aus Abb. 5.5. Dann ist z. B.

G' = ({a, d, c}, { (a, d), (d, c) })

ein Teilgraph von *G*. Er wird aber nicht durch seine Knotenmenge induziert. Das gilt für den folgenden Teilgraphen, der zusätzlich die Kante *(a, c)* enthält:

G' = ({a, d, c}, { (a, d), (a, c), (d, c) }

Eine elementare und wirksam verwendbare Eigenschaft von Graphen ist ihr *Knotengrad*. Er macht Aussagen über die Zahl der Kanten, die mit einem Knoten verbunden sind.

Definition 5.4: Grad in ungerichteten Graphen

Sei G = (V, E) ein ungerichteter Graph. Dann ist der ***Grad eines Knotens*** *v die Anzahl der Kanten {x, v} ∈ E, die in v enden. Der* ***Grad des Graphen*** *G ist der maximale Grad seiner Knoten.* ■

In Abb. 5.6 ist ein ungerichteter Graph mit seinen Knotengraden angegeben. Insgesamt hat der Graph den Grad 4. Man beachte, dass die Schleife zwar mit ihren beiden Enden mit dem Knoten verbunden ist, aber nur einmal zur Zählung der Kanten des Knotens beiträgt.

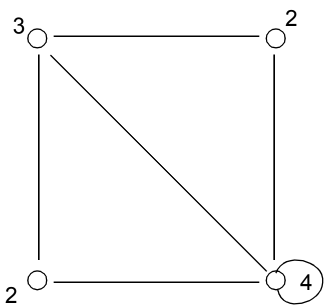


Abbildung 5.6: Knotengrade im ungerichteten Graphen

Definition 5.5: Grad in gerichteten Graphen

Sei G = (V, E) ein gerichteter Graph. Dann ist der ***Eingangsgrad eines Knotens*** *v die Anzahl der Kanten (x, v) ∈ E, die in v münden. Der* ***Ausgangsgrad eines Knotens*** *v ist die Anzahl der Kanten (v, x) ∈ E, die von v ausgehen. Der* ***Grad eines Knotens*** *v ist die Summe seines Eingangs- und Ausgangsgrades. Der* ***Eingangs-, Ausgangsgrad oder Grad des Graphen*** *G ist der entsprechende maximale Wert seiner Knoten.* ■

In Abb. 5.7 sind für ein Beispiel eines gerichteten Graphen Eingangs-, Ausgangsgrad und Grad der Knoten angegeben. Man beachte, dass die Schleife sowohl zur Zählung des Eingangsgrades als auch des Ausgangsgrades und deshalb zweimal zur Zählung des Grades ihres Knotens beiträgt.

Beim Einsatz eines Graphen *G = (V, E)* zur Modellierung beschreibt *V* die Existenz einer Menge verschiedener Objekte und *E* die Existenz bzw. Abwesenheit einer bestimmten Beziehung zwischen je zwei Objekten. Dies ist meist nur der Kern der modellierten Information, wie im Beispiel der Städteverbindungen von Abb. 5.1. Viele Modellierungsaufgaben erfordern jedoch, dass den Knoten oder den Kanten weitere Informationen zugeordnet werden. Dies leisten Markierungsfunktionen für Knoten oder für Kanten. In Abb. 5.8 haben wir die Städteverbindungen aus Abb. 5.1 ergänzt um Angaben der Einwohnerzahl in Tausend zu den Städten und der Entfernung in Kilometern zu den Autobahnverbindungen.

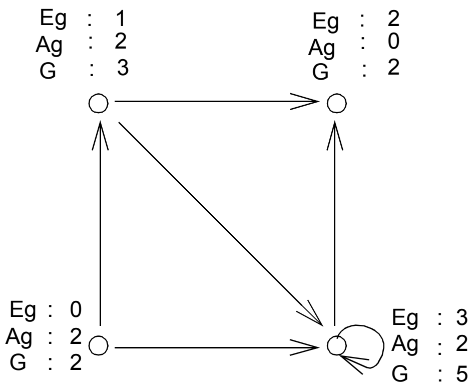


Abbildung 5.7: Eingangsgrad (Eg), Ausgangsgrad (Ag) und Grad (G) im gerichteten Graphen

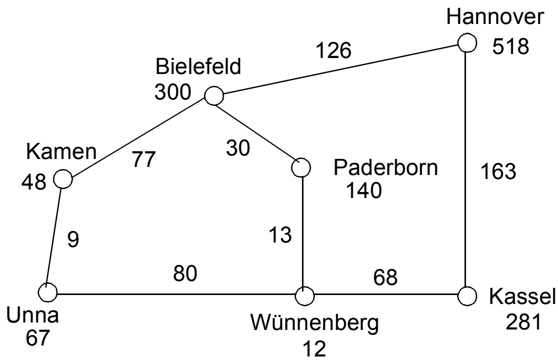


Abbildung 5.8: Graph mit Knoten- und Kantenmarkierungen

Eine *Knotenmarkierung MV* ist eine Funktion mit der Signatur $MV: V \to WV$, wobei WV ein geeigneter Wertebereich ist. In unserem Beispiel ist *Einwohnerzahl:* $V \to \mathbb{N}$ eine Knotenmarkierung. In der grafischen Darstellung annotieren wir einfach die Werte der Funktion an den Knoten. Wenn mehrere Knotenfunktionen eingesetzt werden, geben wir zusätzlich die Funktionsnamen an, wie in Abb. 5.7.

Entsprechend ist eine *Kantenmarkierung ME* eine Funktion mit der Signatur $ME: E \to WE$, wobei WE ein geeigneter Wertebereich ist. In dem Beispiel von Abb. 5.8 ist *Entfernung:* $E \to \mathbb{N}$ eine Kantenmarkierung. In der grafischen Darstellung annotieren wir die Werte der Funktion an den Kanten.

Abb. 5.9 zeigt weitere Anwendungen von Knoten- und Kantenmarkierungen in einem Kantorowitsch-Baum: Die Knoten sind mit Symbolen markiert, die die Operatoren und Variablen angeben. Die Markierung der Kanten legt eine Reihenfolge der Kanten fest.

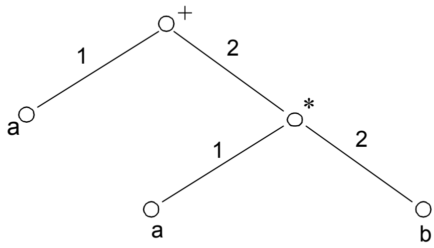


Abbildung 5.9: Beispiel für Knoten- und Kantenmarkierungen

Mit einer Kantenmarkierung kann man auch ausdrücken, dass es zwischen zwei Knoten mehr als eine Kante gibt. Die Markierungsfunktion gibt dann an, für wie viele Verbindungen die eine Kante des Graphen steht.

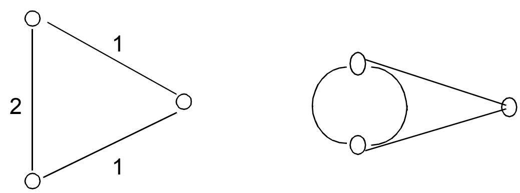


Abbildung 5.10: Multigraph mit Mehrfachkanten

Definition 5.6: Multigraph

Sei $G = (V, E)$ ein gerichteter oder ungerichteter Graph und $m: E \rightarrow$ IN eine Kantenmarkierung, die angibt, dass die Kante $(u, v) \in E$ ggf. mehrere $m(u, r) = n$ Kanten repräsentiert. Wir nennen G mit m dann einen ***Multigraph****. Die Definition der Knotengrade wird auf Multigraphen übertragen unter Berücksichtigung der durch m angegebenen Vielfachheit der Kanten.* ■

Abb. 5.10 zeigt einen Multigraphen, links mit Angabe der Kantenfunktion *m*. Rechts ist stattdessen die doppelte Kante zweimal gezeichnet. Da diese Graphik anschaulicher ist als die Angabe der Vielfachheit für die Kantenfunktion, verwenden wir sie meist zur Darstellung von Multigraphen.

5.2 Wegeprobleme

Die Kanten eines Graphen bilden eine Relation, die angibt, welche Knoten direkt verbunden sind. Setzt man die Relation transitiv fort, so erhält man eine, die angibt, ob zwei Knoten ggf. über mehrere Kanten hinweg verbunden sind. Viele Modellierungen geben den Kanten eines Graphen eine räumliche Bedeutung: Zwei Objekte sind z. B. durch eine Straße, Brücke oder Tür verbunden. Die transitive Fortsetzung der Kantenrelation gibt dann an, ob es einen Weg zwischen zwei Objekten gibt. Viele Fragestellungen bei der Modellierung mit Graphen lassen sich auf die Existenz von Wegen mit bestimmten Eigenschaften zurückführen. In diesem Abschnitt führen wir die Grundbegriffe zur Formulierung von Wegeproblemen ein und geben einige typische Modellierungen an, die diese Begriffe verwenden.

Als einführendes Beispiel für Wegeprobleme zeigen wir das so genannte Königsberger Brückenproblem: Der Schweizer Mathematiker Leonhard Euler formulierte und löste es 1736 und gilt damit als Begründer der Graphentheorie. Zu seiner Zeit gab es in der Stadt Königsberg sieben Brücken über den Fluss Pregel, die die Ufer und zwei Inseln miteinander verbanden. Abb. 5.11 skizziert ihren Verlauf.

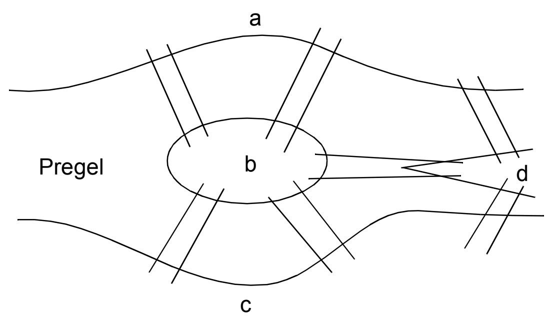


Abbildung 5.11: Skizze der Königsberger Brücken um 1736

Euler formulierte zwei Fragen dazu:

a) Gibt es einen Weg, der jede der sieben Brücken genau einmal überquert und zum Ausgangspunkt zurückkehrt?

b) Gibt es einen Weg, der jede der sieben Brücken genau einmal überquert?

Die Modellierung der Aufgabe liegt nahe: Jedes zusammenhängende Gebiet, das nicht durch einen Flussarm geteilt wird, wird durch einen Knoten repräsentiert. In Abb. 5.11 sind das die Ufer *a* und *c* und die Inseln *b* und *d.* Die Brücken werden durch Kanten modelliert. Da *a* und *b* sowie *b* und *c* durch jeweils zwei Kanten verbunden werden, liegt ein Multigraph vor (Abb. 5.12).

Man könnte nun geneigt sein, die obigen Fragen (a) und (b) zum Königsberger Brückenproblem zu beantworten, indem man Wege mit diesen Eigenschaften sucht. Findet man einen, lautet die Antwort „ja“, findet man keinen und hat alle Möglichkeiten geprüft, so lautet die Antwort „nein“. Euler hat jedoch ein deutlich einfacheres Verfahren zur Prüfung angegeben. Es kann die Entscheidungen an den Knotengeraden ermitteln: Bewegt sich jemand auf einem Rundweg durch einen Graphen, dann kommt er an jedem Knoten genauso oft an, wie er von dort weggeht. Da dieses in unserem Fall auf unterschiedlichen Kanten geschehen soll, muss der Grad jedes Knotens gerade sein, wenn es einen Rundweg wie in (a) gefordert geben soll. Bei der Berechnung des Knotengrades werden gemäß

Definition 5.8 die Mehrfachkanten des Multigraphen entsprechend ihrer Anzahl gezählt. Die Knoten des Multigraphen in Abb. 5.10 haben die Grade

a : 3, b : 5, c : 3, d : 3

Deshalb kann es den gesuchten Rundweg nicht geben.

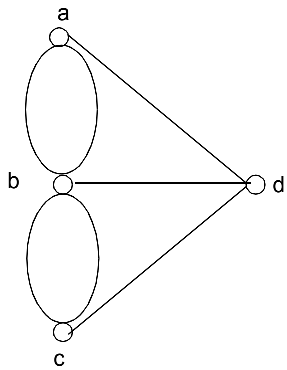


Abbildung 5.12: Multigraph modelliert die Königsberger Brücken

Für die Anwort auf Frage (b) brauchen wir nur noch zu untersuchen, ob es einen solchen Weg gibt, der nicht zum Ausgangspunkt zurückführt. Nach der gleichen Überlegung wie oben würde man den Ausgangspunkt einmal öfter verlassen, als man dort ankommt, und am Endpunkt einmal öfter ankommen, als man ihn verlassen hat. Solch einen Weg gibt es genau dann, wenn der Grad zweier Knoten ungerade und der aller übrigen Knoten gerade ist (siehe Satz 5.1 am Ende dieses Abschnittes). Das trifft auf das Königsberger Brückenmodell nicht zu. Deshalb gibt es dort auch solch einen Weg (b) nicht.

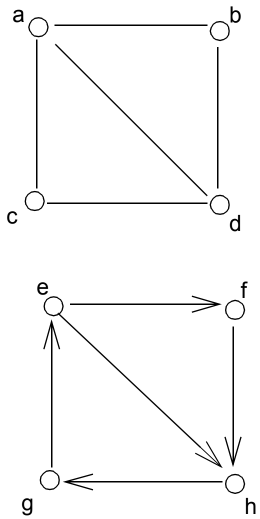
Dies ist wieder ein Beispiel, wo man sehr einfach die Existenz einer Lösung erkennen kann, ohne nach Lösungsinstanzen, d. h. Wegen durch den Graphen, suchen zu müssen.

Wir führen nun Grundbegriffe ein, um Wegeprobleme formal zu beschreiben und zu lösen.

Definition 5.7: Weg

Sei $G = (V, E)$ ein ungerichteter Graph. Eine Folge von Knoten $(v_0, v_1, ..., v_n)$ mit $\{v_i, v_{i+1}\} \in E$ für $0 \leq i \leq n -1$ und $n \geq 0$ heißt ein ***Weg von v_0 nach v_n****. Er hat die* ***Länge n****. Entsprechend sind Wege in gerichtete Graphen mit den Kanten $(v_i, v_{i+1}) \in E$ definiert.* ■

Die Länge eines Weges gibt also gerade an, wie viele Kanten auf dem Weg passiert werden. Dabei können einige Kanten auch mehrfach vorkommen. Definition 5.7 lässt auch Wege der *Länge 0* zu, damit beim Umgang mit dem Wege-Begriff unnötige Sonderfälle vermieden werden. Abb. 5.13 zeigt einige Wege in einem ungerichteten und einem gerichteten Graphen.



	Länge
(c, a, b)	2
(a, b)	1
(a, b, d, a)	3
(a, b, d, a, b, d)	5
(a)	0
(e, h, g)	2
(e, h, g, e)	3
(e)	0

Abbildung 5.13: Wege in Graphen

Definition 5.8: Kreis und Zyklus

Ein Weg $(v_0, v_1, ..., v_n)$ mit $n \geq 1$ und $v_0 = v_n$,, dessen Kanten alle paarweise verschieden sind, heißt ***Kreis im ungerichteten Graphen*** *und* ***Zyklus im gerichteten Graphen****.* ■

Kreise und Zyklen sind also geschlossene Wege, deren Kanten nicht mehrfach vorkommen. Von den in Abb. 5.13 angegebenen Wegen ist *(a, b, d, a) ein Kreis* und *(e, h, g, e) ein Zyklus*. Die Wege *(a), (a, b, a)* und *(a, b, d, a, b, d, a)* erfüllen die Bedingung für Kreise jedoch nicht.

Einige Modellierungen unterlegen gerichteten Graphen eine Bedeutung, die die Existenz von Zyklen ausschließt, z. B. Abhängigkeiten von Aktionen. Diese Eigenschaft wird in folgender Definition benannt.

Definition 5.9: Azyklischer Graph

Ein gerichteter Graph, der keinen Zyklus enthält, heißt ***azyklischer Graph*** *(engl.* ***directed acyclic graph, DAG****).* ■

Manche Graphen bestehen aus Teilgraphen, die nicht miteinander durch Kanten verbunden sind oder mit Kanten, die nur in eine Richtung verlaufen. Abb.5.14 zeigt Beispiele für solche Graphen, in denen Knoten eines Teilgraphen aus einem anderen Teilgraphen nicht erreichbar sind. Dieser *Zusammenhang von Graphen* ist eine wichtige Eigenschaft für Wege- und Verbindungsprobleme, für die Zerlegung von Graphen sowie für viele Algorithmen auf Graphen.

Definition 5.10: Zusammenhang

Ein ungerichteter Graph G = (V, E) heißt ***zusammenhängend****, wenn es für beliebige Knoten v, w ∈ V einen Weg von v nach w gibt. Ein gerichteter Graph G = (V, E) heißt unter derselben Bedingung* ***stark zusammenhängend****.* ■

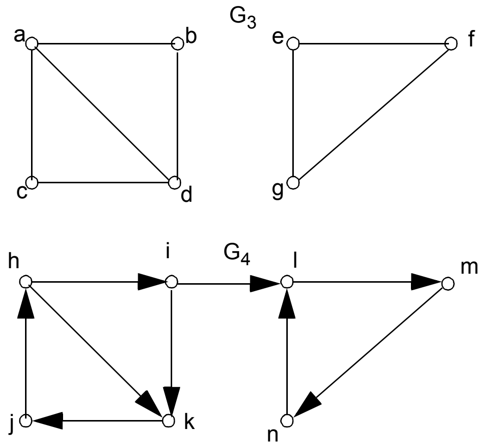


Abbildung 5.14: Zwei nicht-zusammenhängende Graphen

Die Graphen in Abb. 5.13 erfüllen beide die Bedingung aus Definition 5.10; beide Graphen in Abb. 5.14 erfüllen sie nicht. Man beachte, dass auch ein Graph *G = ({a}, ∅)* die Bedingung erfüllt, denn für *v = w = a gilt (a)* ist ein *Weg von a nach a*.

Insbesondere für systematische Zerlegungen von Graphen benötigt man folgenden Begriff.

Definition 5.11: Zusammenhangskomponente

Ein Teilgraph G' = (V', E') des ungerichteten bzw. gerichteten Graphen G = (V, E) heißt ***Zusammenhangskomponente*** *bzw.* ***starke Zusammenhangskomponente****, wenn folgende Bedingungen beide gelten:*

a) G' ist zusammenhängend bzw. stark zusammenhängend.
b) G hat keinen anderen Teilgraphen G'' , der zusammenhängend bzw. stark zusammenhängend ist und G' als Teilgraph enthält. ■

Man beachte, dass in Definition 5.11 die Bedingung (b) dafür sorgt, dass Zusammenhangskomponenten maximale (stark) zusammenhängende Teilgraphen sind. In Abb. 5.14 hat G_3 zwei Zusammenhangskomponenten. Sie werden durch *{a, b, c, d}* und *{e, f, g}* induziert. Der durch *{a, c, d}* induzierte Teilgraph ist zwar zusammenhängend, aber nicht Zusammenhangskomponente. G_4 hat auch zwei Zusammenhangskomponenten. Sie werden durch *{h, i, j, k}* und *{l, m, n}* induziert. Der durch *{h, j, k}* induzierte Teilgraph ist zwar stark zusammenhängend, aber nicht Zusammenhangskomponente.

Wir kommen nun auf das Königsberger Brückenproblem vom Anfang des Abschnittes zurück und definieren die dafür charakteristischen Eigenschaften präzise.

Definition 5.12: Euler-Weg, Euler-Kreis

Sei G = (V, E) ein ungerichteter, zusammenhängender Graph ohne Schleifen. Dann heißt ein Weg w ***Euler-Weg*** *bzw. ein Kreis k heißt* ***Euler-Kreis****, wenn w bzw. k jede Kante aus E genau einmal enthält.* ■

Beispiele für einen Euler-Weg und einen Euler-Kreis sind in Abb. 5.15 angegeben. Die Entscheidung über die Existenz solcher Wege könnten wir als Satz formulieren:

Satz 5.1: Existenz von Euler-Wegen und Euler-Kreisen

Sei G = (V, E) ein ungerichteter, zusammenhängender Graph ohne Schleifen.

a) G hat einen Euler-Kreis genau dann, wenn alle Knoten geraden Grad haben.
b) G hat einen Euler-Weg, der kein Kreis ist, genau dann, wenn genau zwei Knoten ungeraden Grad haben. ■

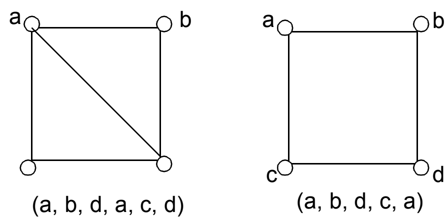


Abbildung 5.15: Graphen mit Euler-Weg und Euler-Kreis

Der Beweis des Satzes ist recht einfach und anschaulich: Wir beginnen mit (a):

Sei $w = (v_0, v_1, ..., v_n)$ ein Euler-Kreis. Bilden wir daraus eine Menge gerichteter Kanten $\{(v_0, v_1), ..., (v_{n-1}, v_n)\}$, dann kommt jeder von *w* berührte Knoten darin genauso oft an erster wie an zweiter Position der Paare vor. Da *w* alle Kanten aus *E* genau einmal berührt, haben alle Knoten aus *w* geraden Grad. Da *G* zusammenhängend ist, berührt *W* alle Knoten aus *V*. Für die Gegenrichtung des Beweises nehmen wir an, dass alle Knoten einen geraden Grad haben. Ausgehend von einem beliebigen Knoten konstruieren wir einen Euler-Kreis: Wir fügen eine noch nicht benutzte Kante in den bisher konstruierten Weg ein. Solange der Euler-Kreis noch nicht vollständig ist, gibt es eine solche Kante: Denn der bisher letzte Knoten auf dem Weg wurde auf dem Weg n-mal erreicht und *(n−1)*-mal verlassen. Da sein Grad gerade ist, gibt es noch eine Kante, mit der wir den Weg verlängern können. Da der Graph zusammenhängend ist, endet das Verfahren erst, wenn der Kreis geschlossen ist und alle Kanten auf dem Weg vorkommen. Den Beweis für Teil (b) führt man entsprechend.

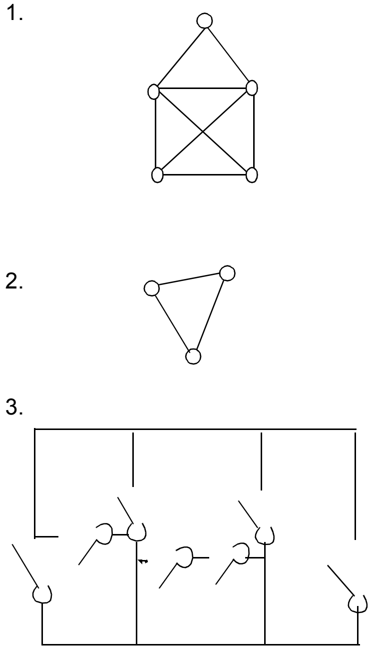


Abbildung 5.16: Wegeprobleme mit Euler-Wegen

Wir betrachten nun drei typische Aufgaben, deren Lösung auf dem Prinzip der Euler-Kreise und -Wege beruht.

1. Kann man die Figur in Abb. 5.16 (1) in einem Zuge nachzeichnen? Das ist möglich, wenn es einen Euler-Weg oder einen Euler-Kreis gibt. Wir ermitteln deshalb die Knotengrade: Die beiden unteren Knoten haben den Grad 3, der Knoten an der Spitze den Grad 2, die beiden dazwischen den Grad 4. Also können wir Euler-Wege finden, deren Endknoten jeweils die beiden unteren Knoten sind. Euler-Kreise gibt es aber nicht.
2. Für eine Inselgruppe, die aus $n \geq 3$ Inseln besteht, sollen Schiffsverbindungen organisiert werden. Jede Insel soll mit jeder anderen direkt verbunden sein. Es steht nur ein einziges Schiff zur Verfügung, um diese Verbindungen wahrzunehmen. Kann es auf einer Tour alle Verbindungen abfahren? Für welche n ist das möglich? Abb. 5.16 (2) zeigt als Beispiel einen Graphen, der 3 Inseln modelliert, von denen jede mit jeder anderen verbunden ist. Alle Knoten dieses Graphen haben den Grad 2. Deshalb gibt es einen Euler-Kreis, der hier ganz offensichtlich ist. Er kann als Plan für die Schiffsverbindung verwendet werden. Man kann leicht zeigen, dass in solchen Graphen mit beliebiger Knotenzahl $n \geq 3$ jeder Knoten den Grad $n - 1$ hat. Deshalb gibt es Euler-Kreise genau dann, wenn die Knotenzahl ungerade ist.

3. Ein Gruselkabinett für den Jahrmarkt soll geplant werden: Das ist ein Haus, das $n \geq 1$ Räume hat, eine Eingangstür und eine Ausgangstür sowie beliebig viele Türen jeweils in der Wand zwischen zwei Räumen. Die Türen werden so manipuliert, dass sie zum Entsetzen der Besucher endgültig verriegeln, sobald jemand durchgegangen ist. Jeder Besucher wird einzeln in das Haus geschickt. Die Türen sollen so angeordnet werden, dass sich der Besucher nicht einsperren kann. Erst wenn er erleichtert das Haus verlassen hat, werden alle Türen für den nächsten Besucher wieder freigegeben. Abb. 5.16 (3) zeigt ein Beispiel für solch ein Haus.
   Wir modellieren die Aufgabe mit einem ungerichteten Graphen. Jeder Raum sowie der Eingangsbereich (E) und der Ausgangsbereich (A) werden durch einen Knoten repräsentiert, jede Tür zwischen zwei Räumen durch eine Kante zwischen den entsprechenden Knoten. Im Allgemeinen entsteht auf diese Weise ein Multigraph. Wir untersuchen ihn, ob er einen Euler-Weg von E nach A hat. Das ist der Fall, wenn der Grad der Knoten E und A ungerade und der aller übrigen Knoten gerade ist. Dann kann eine Person auf dem Euler-Weg von E nach A gehen und dabei jede Tür genau einmal passieren. Immer, wenn sie in einen Raum gelangt, steht noch mindestens 1 Tür offen, durch die sie ihn wieder verlassen kann. Man kann also nicht eingesperrt werden. Eventuell könnte der Besucher den Weg abkürzen, indem er den Ausgang benutzt, bevor er alle Türen passiert hat.

Modellierungen von Räumen mit Verbindungstüren, wie im Beispiel 3, kommen häufig und in unterschiedlichen Kontexten vor. Es kann dasselbe Prinzip angewandt werden: Knoten repräsentieren Räume, Kanten repräsentieren Türen. Wenn zwei Räume durch mehr als eine Tür direkt verbunden sind, ist das Modell ein Multigraph.

Während die Euler-Kreise und -Wege durch einmaliges Vorkommen von Kanten charakterisiert sind, definieren wir nun *Hamilton-Kreise* so, dass die Knoten des Graphen genau einmal vorkommen.

Definition 5.13: Hamilton-Kreis und -Weg

Ein Weg $w = (v_0, v_1, ..., v_n)$ in einem Graphen $G = (V, E)$ heißt ***Hamilton-Weg****, wenn jeder Knoten aus V in w genau einmal vorkommt. w heißt* ***Hamilton-Kreis****, wenn w ein Kreis ist und jeder Knoten aus V in $v_0, v_1, ..., v_{n-1}$ genau einmal vorkommt.* ■

Zu entscheiden, ob ein Graph einen Hamilton-Kreis enthält, ist um Größenordnungen schwieriger, als die Frage für Euler-Kreise zu entscheiden: Für Hamilton-Kreise ist das Entscheidungsproblem NP-vollständig, d. h. ein effizientes Verfahren ist nicht bekannt; für Euler-Kreise kann man die Entscheidung durch Untersuchen der Knotengrade mit linearem Aufwand fällen.

Der Graph G_1 in Abb. 5.17 hat Hamilton-Kreise, z. B. *(a, b, e, d, c, a)*.

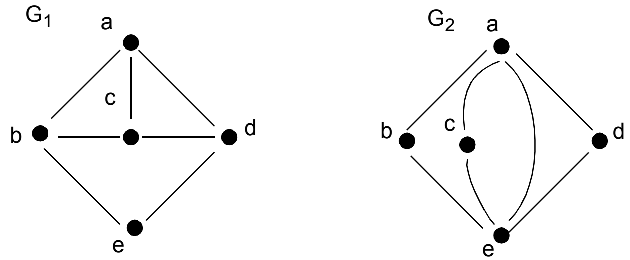


Abbildung 5.17: Hamilton-Kreise in Graphen

Der Graph G_2 hat keine Hamilton-Kreise, aber einen Hamilton-Weg. Wir können dies für diesen speziellen Graphen nachweisen: Nehmen wir an, G_2 hätte einen Hamilton-Kreis *w*. Weil *b, c, d* auf *w* liegen müssen und alle den Grad 2 haben, müssen beide Kanten, die jeweils zu *b, e* und *d* führen, auf w liegen. Also liegen *{a, b}, {a, c}* und *{a, d}* auf *w*. Es ist aber nicht möglich, dass ein Hamilton-Kreis mehr als zwei Kanten enthält, die mit demselben Knoten verbunden sind. Also ist *w* kein Hamilton-Kreis.

Auch Hamilton-Kreise werden zur Modellierung vieler Aufgaben verwendet. Die bekannteste Aufgabe, die mit Hamilton-Kreisen modelliert wird, ist die des Handlungsreisenden (engl.: Travelling Salesman's Problem): *n* Städte sind mit Straßen bestimmter Länge verbunden. Gesucht ist eine kürzeste Rundreise durch alle Städte. Für diese Aufgabe wird eine Kantenmarkierung eingeführt, die die Länge der Straßenverbindungen modelliert. Diese Angaben können auch verallgemeinert werden zu einem Maß für die Kosten, die entstehen, wenn man diese Kante in einen Weg aufnimmt. Dadurch wird die Aufgabe zu einer Optimierungsaufgabe: Es wird ein Hamilton-Kreis gesucht, der bezüglich dieser Kosten minimal ist. In Abb. 5.18 ist ein Graph mit einer Kantenmarkierung angegeben. Ein minimaler Hamilton-Kreis darin ist (a, c, d, b, a) mit den Kosten 157.

Als zweites Beispiel betrachten wir folgende Aufgabe: In einem Parallelrechner seien die Prozessoren als $n * n$-Gitter verbunden. Abb. 5.19 zeigt dies für $n = 4$. Eine Botschaft soll von einem Prozessor zu einem anderen weitergegeben werden, jeden Prozessor erreichen und schließlich zum Initiator zurückkehren. Für welche *n* ist das möglich? Es wird also nach der Existenz eines Hamilton-Kreises gefragt.

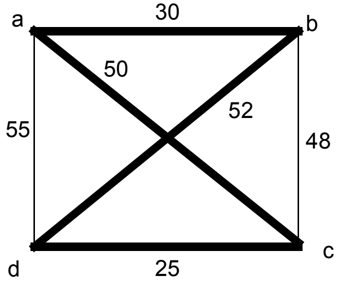


Abbildung 5.18: Minimaler Hamilton-Kreis

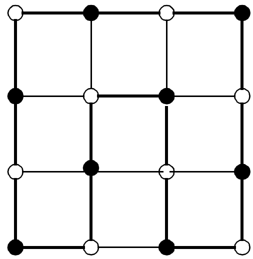


Abbildung 5.19: Hamilton-Kreis im Gitter

Für diese spezielle Klasse von Graphen (Gitter), können wir die Frage mit einfachen Überlegungen beantworten. Wir stellen uns vor, die Knoten des Gitters seien abwechselnd schwarz und weiß gefärbt, sodass zwei Knoten, die verbunden sind, unterschiedliche Farben haben. Dann werden wir auf jedem Weg durch den Graphen die Farben alternieren. Ein Hamilton-Kreis muss daher ebenso viele schwarze wie weiße Knoten berühren. Das ist genau dann möglich, wenn die Anzahl der Knoten $n * n$ und damit auch die Kantenlänge n des Gitters gerade ist.

5.3 Verbindungsprobleme

Wir wenden uns nun einer weiteren Klasse von Aufgaben zu, die auch mit ungerichteten Graphen modelliert werden. Während wir zur Beschreibung von Wegeproblemen Wege mit bestimmten Eigenschaften gesucht haben, interessieren uns bei Verbindungsproblemen die Existenz von Verbindungen (Wegen) zwischen Knoten und die Erreichbarkeit von Knoten. Das bedeutet, dass der Zusammenhang von Graphen auch für diese Aufgabenklasse eine zentrale Eigenschaft ist. Häufig möchte man auch die Verbindungen in einem Modell so minimieren, dass die Anzahl der Kanten oder ihre Kosten möglichst klein sind.

Für die Modellierung von Verbindungen sind die folgenden Begriffe grundlegend:

Definition 5.14: Ungerichteter Baum

Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph. Wenn G keine Kreise enthält, ist es ein ***ungerichteter Baum***. *Alle Knoten in V, die den Grad 1 haben, nennen wir dann Blätter des Baumes.* ■

Abb. 5.20 gibt drei Beispiele für ungerichtete Bäume an.

Satz 5.2: Anzahl von Knoten und Kanten

Für die Anzahl von Knoten und Kanten in einem ungerichteten Baum $G = (V, E)$ gilt $|E| = |V| - 1$. ■

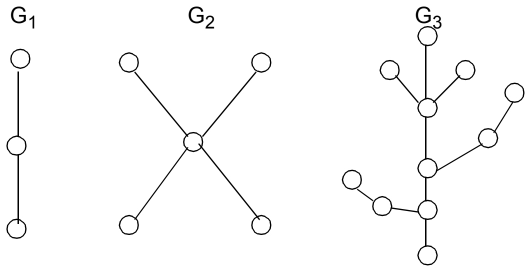


Abbildung 5.20: Ungerichtete Bäume

Diesen Satz beweist man induktiv:

Induktionsanfang: Ein ungerichteter Graph, der nur einen Knoten und keinen Kreis hat, kann nach Definition 5.8 keine Kante haben, d. h. $|V| = 1$ und $|E| = 0$. Also gilt $|E| = |V|-1$.

Induktionsschritt: Sei $G = (V, E)$ ein ungerichteter Baum mit $|V| = n \geq 1$. Wir bilden $V' = V \cup \{x\}$ durch Zufügen eines neuen Knotens x. Wir verbinden x mit einem beliebigen Knoten y aus V über die Kante $\{x, y\}$. Dann ist $E' = E \cup \{\{x, y\}\}$ und $|E'| = |E|+1$, weil x ein neuer Knoten ist. Weil G ein Baum ist, ist auch $G' = (V', E')$ zusammenhängend und kreisfrei, also ein Baum. Wegen $|E| = |V|-1$ gilt $|E|+1 = |V|+1-1$, also $|E'| = |V'|-1$ für den ungerichteten Baum G'.

Induktionsschluss: Die Formel gilt also für jeden ungerichteten Baum mit $|V| \geq 1$.

Da ein ungerichteter Baum zusammenhängend ist, gibt es einen Weg zwischen je zwei beliebigen Knoten; da er auch kreislos ist, sind je zwei beliebige Knoten durch genau einen Weg verbunden. Für eine gegebene Knotenmenge erzeugt deshalb der ungerichtete Baum Zusammenhang unter Aufwendung einer kleinstmöglichen Anzahl von Kanten. Diese Eigenschaft führt zum folgenden Begriff:

Definition 5.15: Spannbaum

*Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph und $G' = (V', E')$ ein ungerichteter Baum, der Teilgraph von G ist, und $V = V'$. Dann ist G' ein **Spannbaum** von G.*

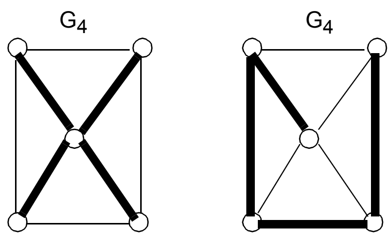


Abbildung 5.21: Spannbäume zu G_4

Geht man vom Graphen G zu einem seiner Spannbäume über, heißt das auch, dass man die Kantenmenge von G auf |V| – 1 Kanten verkleinert, ohne den Zusammenhang der Knoten aufzugeben. In Abb. 5.21 werden zwei verschiedene Spannbäume zu demselben Graphen G_4 angegeben.

Mit dem Begriff des Spannbaums wird also ein bezüglich der Kanten kostengünstiger Zusammenhang modelliert. Manche Modellierungen beziehen die Kosten auf die Anzahl der Kanten; dann ist jeder Spannbaum gleich günstig. Andere streben ein Minimum bezüglich der Kantenmarkierungen an; dann werden unter den Spannbäumen günstigste gesucht. Für jede der beiden Aufgabenklassen geben wir ein Beispiel an:

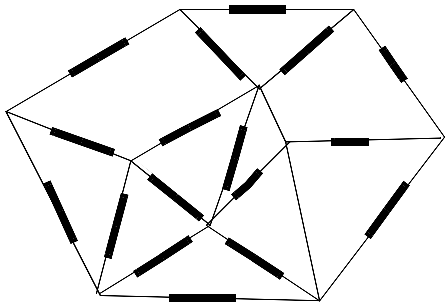


Abbildung 5.22: Gefängnis mit Türen – Wege in die Freiheit

Abb. 5.22 zeigt den Grundriss eines mittelalterlichen Gefängnisses. Die Gefangenen werden in verwinkelten Räumen gehalten, deren Verbindungstüren (breite Linien) verschlossen sind. Sie haben einen Gebäudeplan ergattert und planen einen Massenausbruch. Dazu wollen sie gerade so viele Türen sprengen, dass aus jedem Raum die Gefangenen entkommen können. Wir modellieren die Aufgabe nach dem Schema „Räume mit Türen“ (siehe Ende des vorigen Abschnittes). Dabei wird die Umgebung des Gefängnisses als ein Knoten repräsentiert. Wir benötigen einen Graph, der alle Knoten (Räume und Umgebung) enthält. Seine Kanten repräsentieren gesprengte Türen. Sie sollen gerade von jedem Raum einen einzigen Weg in die Freiheit der Umgebung liefern. Das heißt, wir suchen einen Spannbaum zu dem Graphen, der die verschlossenen Türen repräsentiert.

Die zweite Aufgabe ist der Organisation von Nachrichtendiensten nachempfunden. Abb. 5.23 beschreibt den ursprünglich hoch-geheimen Plan eines Agentenringes. Die Knoten A bis H repräsentieren Agenten. Sie sollen alle direkt oder indirekt miteinander kommunizieren. Dafür stehen die als Kanten angegebenen direkten Verbindungen zur Verfügung. Jede davon ist mit einem Wert markiert, der das Risiko charakterisieren soll, dass die Verbindung aufgedeckt wird. Die Planer suchen für das Kommunikationssystem ein Netz mit geringstmöglichem Risiko. Sie benötigen also einen Spannbaum des Graphen, so dass die Summe der Kantenwerte minimal ist.

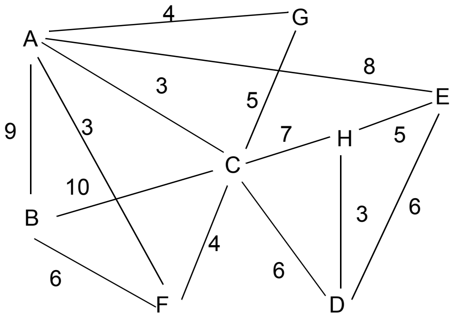


Abbildung 5.23: Agentenverbindungen mit Risikofaktoren

Da der Zusammenhang in Graphen für Verbindungsprobleme entscheidend ist, wollen wir nun drei weitere Begriffe einführen, die die Bedeutung bestimmter Knoten und Kanten für den Zusammenhang hervorheben.

Definition 5.16: Schnittknoten

Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph. Dann ist $v \in V$ ein ***Schnittknoten*** *in G, wenn der Teilgraph, der durch Entfernen von v aus G entsteht, nicht zusammenhängend ist.* ■

Definition 5.17: Brückenkante

Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph. Dann ist $e \in E$ eine ***Brückenkante****, wenn der Teilgraph, der durch Entfernen von e aus G entsteht, nicht mehr zusammenhängend ist.* ■

Definition 5.18: Orientierbar

Ein ungerichteter, zusammenhängender Graph $G = (V, E)$ heißt ***orientierbar****, wenn man für jede Kante $e \in E$ eine Richtung so festlegen kann, dass der entstehende gerichtete Gaph stark zusammenhängend ist.* ■

Abb. 5.24 zeigt einen Graphen mit einer Brückenkante und drei Schnittknoten. Entfernt man die Brückenkante oder einen der Schnittknoten, so zerfällt der Graph in zwei Zusammenhangskomponenten. Es ist auch sofort klar, dass dieser Graph nicht orientierbar ist: Legt man die Orientierung der Brückenkante in der einen oder der anderen Richtung fest, so wird ein Teilgraph vom anderen unerreichbar. Diese Konsequenz gilt sogar in beiden Richtungen:

Satz 5.3: Orientierbarkeit

Ein ungerichteter, zusammenhängender Graph ist genau dann orientierbar, wenn er keine Brückenkante hat. ■

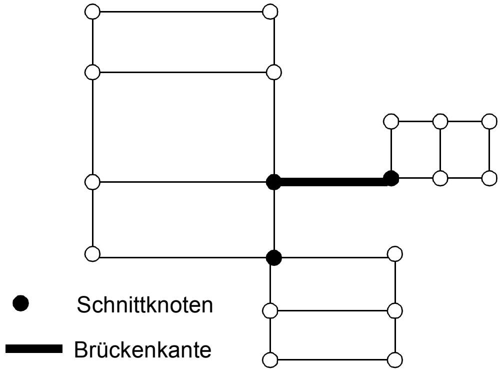


Abbildung 5.24: Schnittknoten und Brückenkante

Die Begriffe *Brückenkante*, *Schnittknoten* und *Orientierbarkeit* können durch folgende Modellierung illustriert werden:

In einer Stadt sollen einzelne Straßen zur Reparatur gesperrt werden. Bleiben nach einer Sperrung noch alle Plätze der Stadt erreichbar? Wir modellieren die Plätze als Knoten und die Straßen als Kanten eines ungerichteten Graphen, der zusammenhängend sein muss. Ein Beispiel ist der Graph in Abb. 5.24. Die Frage übersetzt man dann in das Modell: Wird eine Straße gesperrt, die durch eine Brückenkante repräsentiert wird, dann gibt es Teile des Graphen, die von anderen nicht mehr erreichbar sind.

Wir variieren nun die Aufgabe etwas: Im Zentrum einer Großstadt sollen zur Hauptverkehrszeit alle Straßen zu Einbahnstraßen erklärt werden. Bleiben dann alle Plätze von überall her erreichbar? Die Frage übersetzt man dann in das Modell: Ist der Graph orientierbar? Sie wird beantwortet, indem man feststellt, ob der Graph Brückenkanten hat.

Eine letzte Variante: In einer Stadt soll ein Platz für eine Demonstration gesperrt werden. Das Ordnungsamt will den Antrag nur genehmigen, wenn während der Demonstration alle anderen Plätze von überall her erreichbar bleiben. Im Modell muss man prüfen, ob der Platz durch einen Schnittknoten repräsentiert wird.

5.4 Modellierung mit Bäumen

In diesem Abschnitt führen wir den Begriff des *gerichteten Baumes* ein. Solche Bäume spielen eine besondere und wichtige Rolle in der Modellierung: Sie können vielfältige Arten von schrittweiser oder rekursiver Verfeinerung repräsentieren, z. B. die hierarchische Organisationsstruktur einer Firma, die Gliederung eines komplexen Gerätes in seine Komponenten und deren Einzelteile oder die Alternativen, die sich aus einer Folge von Entscheidungen ergeben.

Definition 5.19: Gerichteter Baum

Ein gerichteter, azyklischer Graph G = (V, E) ist ein ***gerichteter Baum****, wenn alle Knoten einen Eingangsgrad von 1 oder 0 haben und es genau einen Knoten w mit Eingangsgrad 0 gibt. w ist die* ***Wurzel*** *von G. G heißt auch gewurzelter Baum.* ■

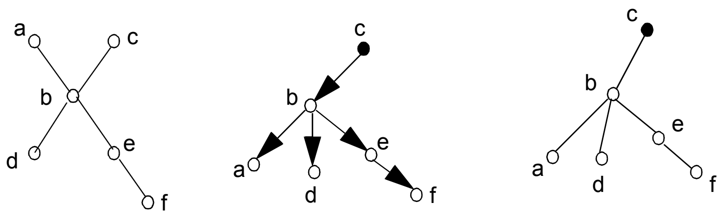


Abbildung 5.25: Ungerichteter und gerichteter Baum

Abb. 5.25 zeigt in der Mitte einen gerichteten Baum. Seine Wurzel ist der Knoten c. In c mündet keine Kante, in alle anderen Knoten mündet eine Kante.

Wir blicken noch einmal zurück auf ungerichtete Bäume, die im vorigen Abschnitt definiert wurden. Darin gibt es zwischen je zwei beliebigen Knoten genau einen Weg. In einem gerichteten Baum gibt es genau einen Weg von der Wurzel zu jedem anderen Knoten. Man kann aus einem ungerichteten Baum einen gerichteten Baum machen. Dazu bestimmt man einen beliebigen Knoten als Wurzel und orientiert dann die Kanten so, dass die Wege von der Wurzel zu den übrigen Knoten führen. Wenn man dieses Verfahren auf den linken Graphen in Abb. 5.25 anwendet und c als Wurzel bestimmt, erhält man den gerichteten Baum in der Mitte der Abbildung. Hiermit ist klar, dass auch für gerichtete Bäume gilt $|E| = |V| - 1$. Häufig wird beim Zeichnen eines gerichteten Baumes die Kantenrichtung nicht angegeben. Wenn die Wurzel bekannt ist, kann sie eindeutig konstruiert werden. Der Wurzelknoten wird meist oben oder oben-links gezeichnet. Wir werden gerichtete Bäume nun meist so, ohne Angabe der Kantenrichtung, zeichnen.

In einem gerichteten Baum unterscheiden wir drei Arten von Knoten: die *Wurzel* mit Eingangsgrad 0, die *Blätter* mit Ausgangsgrad 0 und *innere Knoten* mit Eingangsgrad 1 und Ausgangsgrad > 0. Alternativ zu Definition 5.19 können wir gerichtete Bäume auch induktiv definieren:

Definition 5.20: Gerichteter Baum, Höhe

Der Graph $G_0 = (\{a\}, \emptyset)$ ist ein ***gerichteter Baum der Höhe*** *0. Seien $G_1 = (V_1, E_1), ..., G_n = (V_n, E_n)$ gerichtete Bäume, deren Knotennamen alle paarweise verschieden sind. Die Knoten $v_i \in V_i$ seien die Wurzeln von G_i. Der Baum G_i habe die Höhe k_i. w sei ein neuer Knoten. Dann ist G = (V, E) ein gerichteter Baum mit der Wurzel w, $V = \{w\} \cup V_1 \cup V_2 \cup ... \cup V_n$ und $E = E_1 \cup E_2 \cup ... \cup E_n \cup \{(w, v_1), (w, v_2), ..., (w, v_n)\}$. Der Baum G hat die Höhe h = maximum $(h_1,..., h_n)+1$. Wir nennen die G_i auch* ***Teilbäume*** *von G.* ■

Die Höhe eines Baumes ist auch gleich der Länge des längsten Weges von der Wurzel zu einem Blatt.

Eine besondere Rolle in der Modellierung spielen Bäume, die auf allen Ebenen höchstens zwei Teilbäume haben. Mit ihnen beschreibt man z. B. Kaskaden von Ja-nein-Entscheidungen oder Binär-Codierungen.

Definition 5.21: Binärbaum

*Ein gerichteter Baum heißt **Binärbaum**, wenn seine Knoten einen Ausgangsgrad von höchstens 2 haben. Ein Binärbaum heißt **vollständig**, wenn kein Knoten den Ausgangsgrad 1 hat und die Wege von der Wurzel zu jedem Blatt gleich lang sind.* ■

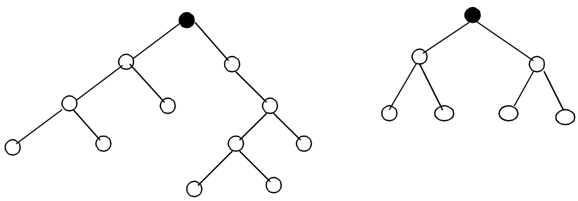


Abbildung 5.26: Unvollständiger und vollständiger Binärbaum

Abb. 5.26 zeigt einen unvollständigen und einen vollständigen Binärbaum.Es gibt einen wichtigen Zusammenhang zwischen der Höhe, der Anzahl der Knoten und der Blätter eines vollständigen Binärbaumes:

Satz 5.4: Höhe vollständiger Binärbäume

Ein vollständiger Binärbaum der Höhe h hat 2^h Blätter und $2^{h+1} - 1$ Knoten. ■

Der Satz kann leicht induktiv bewiesen werden:

Induktionsanfang: Ein vollständiger Binärbaum der Höhe 0 hat nach Definition 5.20 einen Knoten, der ein Blatt ist. D.h. er hat 2^h Blätter und 2^{h+1}-1 Knoten.

Induktionsschritt: Seien $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ vollständige Binärbäume der Höhe h mit 2^h Blättern und $2^{h+1}-1$ Knoten mit den Wurzeln w_1 und w_2. Dann ist $G = (V_1 \cup V_2 \cup \{w\}, E_1 \cup E_2 \cup \{(w, w_1), (w, w_1)\})$ ein vollständiger Binärbaum. Er hat die Höhe h+1 und hat $2 \cdot 2^h = 2^{h+1}$ Blätter und $2 \cdot (2^{h+1}-1)+1 = 2^{h+2}-1$ Knoten.

Induktionsschluss: Also gelten die Formeln für alle $h \geq 0$.

Wir betrachten nun einige Modellierungen, die typisch sind für die Anwendung von gerichteten Bäumen:

In vielen Zusammenhängen können Folgen von Entscheidungen durch einen gerichteten Baum modelliert werden, wir sprechen dann von einem *Entscheidungsbaum*. Abb. 5.27 zeigt einen Entscheidungsbaum zum Morse-Alphabet.

Man kann ihn verwenden, um Meldungen im Morse-Code zu entschlüsseln. In diesem Code wird jeder Buchstabe durch ein Folge von kurzen und langen Signalen verschlüsselt. Man entschlüsselt eine eingehende Meldung, indem man an der Wurzel des Baumes beginnt und bei einem kurzen Signal nach links, bei einem langen nach rechts verzweigt. Eine längere Pause zeigt an, dass ein Buchstabe vollständig übermittelt ist.

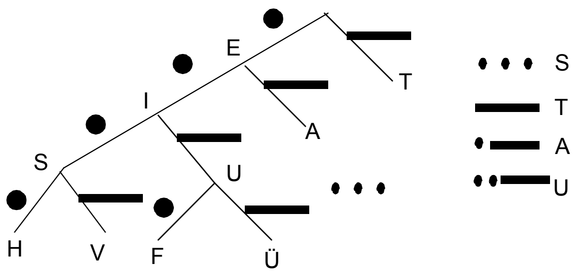


Abbildung 5.27: Entscheidungsbaum zum Morsealphabet

In jedem Entscheidungsbaum modellieren die Knoten einen Zwischenstand bei der Entscheidungsfindung. Sie können entsprechend markiert sein, z. B. mit dem codierten Buchstaben im Baum zum Morse-Code. Die Kanten, die von einem Knoten ausgehen, modellieren die Alternativen, aus denen in diesem Zustand eine ausgewählt werden kann. Das ist im Morse-Code ein kurzes oder langes Signal, das als Kantenmarke angegeben wird.

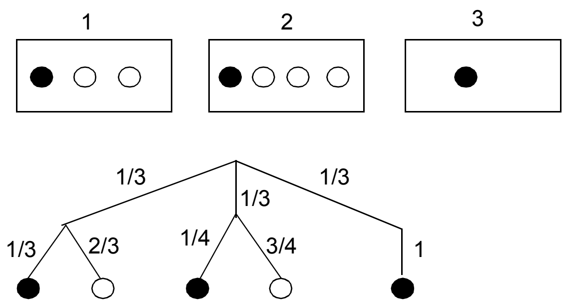


Abbildung 5.28: Wahrscheinlichkeiten im Entscheidungsbaum

Wir können mit einem Entscheidungsbaum auch sehr anschaulich zusammengesetzte Wahrscheinlichkeiten modellieren: Abb. 5.28 zeigt drei Behälter mit einigen schwarzen und weißen Kugeln. Es wurde nun erst ein Behälter ausgewählt und dann daraus eine Kugel gezogen. Die Kantenmarken des Baumes geben an, mit welcher Wahrscheinlichkeit die zugehörige Entscheidung getroffen wird. Multipliziert man die Wahrscheinlichkeiten auf dem Wege von der Wurzel zu einem Blatt, dann erhält man den Wahrscheinlichkeitswert für die Entscheidungsfolge, die das Blatt repräsentiert. So zeigt der Baum, dass mit der Wahrscheinlichkeit 2/9 eine weiße Kugel aus Behälter 1 gezogen wird. Addieren wir die Wahrscheinlichkeiten aller Wege zu weißen Blättern, erhalten wir die Wahrscheinlichkeit, dass eine weiße Kugel aus irgendeinem Behälter gezogen wird:

2/9 + 3/12 = 17/36.

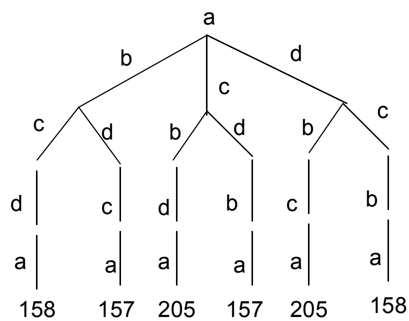


Abbildung 5.29: Lösungsraum zum Handlungsreisenden-Problem von Abb. 5.18

Das dritte Beispiel (Abb. 5.29) modelliert den Lösungsraum eines kombinatorischen Problems. Ein Baum spannt alle potenziellen Lösungen einer Aufgabe auf; sie werden durch die Blätter charakterisiert. Der Weg zu einem Blatt beschreibt die Folge der Entscheidungen, die getroffen werden, um diese Lösung zu erhalten. Diese Abbildung stellt den Lösungsraum zum Problem des Handlungsreisenden mit dem Graphen aus Abb. 5.18 dar. Jeder Weg von der Wurzel des Baumes zu einem Blatt repräsentiert einen Rundweg im Graphen der verbundenen Städte, der bei dem Knoten a beginnt, z. B. *(a, b, c, d, a)*. Die Marken an den Kanten, die von einem Baumknoten ausgehen, geben an, wohin der Rundweg fortgesetzt werden kann, ohne ein Ziel ein zweites Mal zu besuchen. Die Blätter sind mit den Kosten des Rundweges markiert. Daran können wir ablesen, dass es im Lösungsraum zwei Rundwege mit minimalen Kosten 157 gibt: *(a, b, d, c, a)* und *(a, c, d, b, a)*. Natürlich kommen die Rundwege immer in symmetrischen Paaren vor.

Nach dem gleichen Schema werden auch Zugfolgen in Spielen modelliert: Jeder Knoten des Entscheidungsbaumes modelliert einen Spielzustand. Die von dort ausgehenden Kanten geben an, welche Möglichkeiten für den nächsten Zug bestehen. Solche Darstellungen werden z. B. in Schachprogrammen verwendet, um die Folgen der ausstehenden Entscheidung zu analysieren und zu bewerten.

Gerade bei der Modellierung von Spielabläufen können manche Spielzustände, die auf unterschiedlichen Wegen erreicht werden, im Sinne des Spieles denselben Zustand beschreiben. Dann könnte man auch im Entscheidungsbaum die zugehörigen Knoten identifizieren. Damit geht allerdings die Baum-Eigenschaft verloren. Es entsteht dann ein allgemeiner, gerichteter Graph, der sogar Kreise enthalten kann. Sie modellieren, dass eine

Folge von Spielzügen in einen Zustand des Spieles zurückführt, der früher schon einmal durchlaufen wurde.

Gerichtete Bäume werden auch zur *Modellierung von Strukturen* aus ganz unterschiedlichen Anwendungsgebieten mit unterschiedlichen Bedeutungen eingesetzt, z. B. Objektbäume, Typ- und Klassenhierarchien, Ausdrucksbäume und Strukturbäume. Gemeinsam ist diesen Modellen, dass ein Knoten einen abstrakten oder konkreten Gegenstand modelliert und mit den Kanten Beziehungen wie *„besteht aus*“, „*enthält*“ oder *„wird spezialisiert zu“* dargestellt werden.

Schon in Kapitel 3 haben wir die Struktur von Termen durch Bäume dargestellt, wie z. B. in Abb. 5.30. Die Knoten repräsentieren Variable, Konstanten und Operatoren, die Kanten verbinden sie mit ihren Operanden. Formeln bzw. Ausdrücke können ebenso dargestellt werden. Wir nennen die Darstellungsform auch Kantorowitsch-Baum. Er beschreibt, wie ein Term aus Untertermen bzw. ein Ausdruck aus Teilausdrücken zusammengesetzt ist.

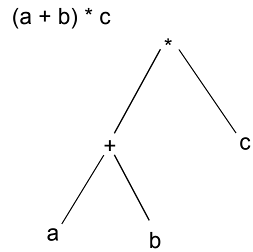


Abbildung 5.30: Kantorowitsch-Baum für einen Term bzw. Ausdruck

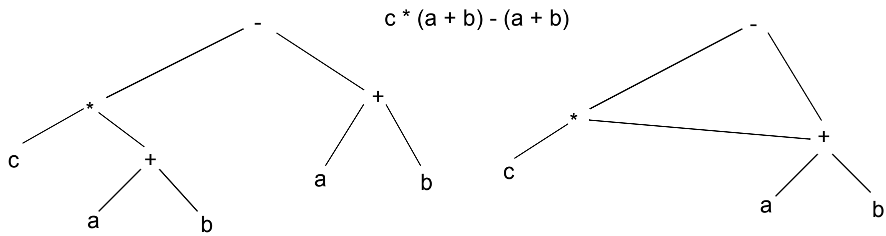


Abbildung 5.31: Identifikation gleicher Teilbäume

In Abb. 5.31 ist ein Ausdruck dargestellt, in dem der Teilausdruck (a + b) zweimal auftritt. In dem rechten Graphen ist diese Eigenschaft dadurch repräsentiert, dass die entsprechenden Knoten identifiziert wurden: Die rechten Operanden des Subtraktions- und des Multiplikationsoperators führen auf denselben Teilgraphen. Hierdurch wird der gerichtete Baum zu einem gerichteten, azyklischen Graphen. (Natürlich müssen wir die im Baum implizit angenommene Kantenrichtung beibehalten.) Solch eine Transformation wird

z. B. von Übersetzern vorgenommen, um Code zu erzeugen, der gleiche Teilausdrücke nur einmal auswertet.

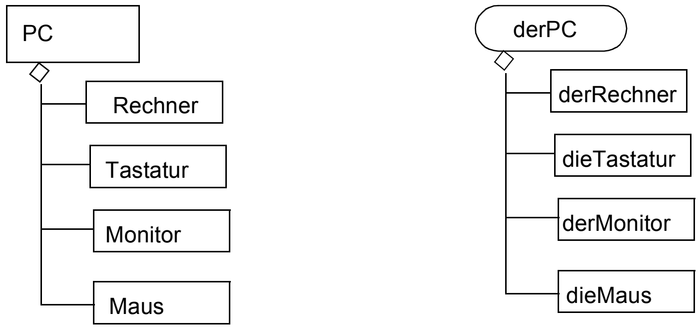


Abbildung 5.32: Klassen- und Objektdiagramm

Auch Klassen- und Objekthierarchien werden häufig durch gerichtete Bäume modelliert. So werden z. B. in Klassendiagrammen der UML-Notation (Abschnitt 6.4) unter anderem Kompositionsbeziehungen modelliert. Die Knoten in einem Klassendiagramm, wie im linken Teil von Abb. 5.32, beschreiben Klassen, die Kompositionskanten geben an, aus Objekten welcher Klassen ein Objekt dieser Klasse bestehen kann. Hier wird also ausgesagt, dass ein PC-Objekt aus einem Rechner-Objekt, einem Tastatur-Objekt, einem Monitor-Objekt und einem Maus-Objekt bestehen kann. Der rechte Graph der Abb. 5.32 ist ein Objekt-Baum. Er modelliert ein bestimmtes PC-Objekt, das aus vier bestimmten Objekten besteht, die jeweils Klassen angehören, wie es das Klassendiagramm vorschreibt.

Solche Objektdiagramme müssen konzeptionell Bäume sein. Denn ein bestimmtes Objekt kann nicht gleichzeitig Teilobjekt mehrerer verschiedener Objekte im engen Sinne einer *„besteht-aus“*-Beziehung sein. Für die Modellierungsebene der Klassen gilt das natürlich nicht: Ein Klassendiagramm kann durchaus modellieren, dass Objekte einer Klasse mehrfach und an verschiedenen Stellen in Objektbäumen vorkommen können. Klassendiagramme können auch Zyklen haben und dadurch Objektstrukturen rekursiv beschreiben, wie z. B. in Abb. 5.33 die Binärbäume, die aus einem Wert und bis zu zwei Unterbäumen bestehen.

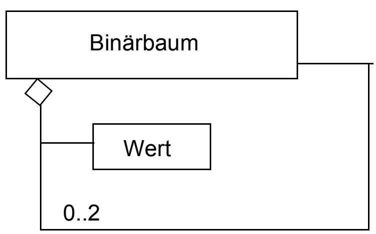


Abbildung 5.33: Rekursive Definition im Klassendiagramm

Als letztes Beispiel für die Darstellung von Strukturen durch gerichtete Bäume betrachten wir Programme und strukturierte Texte oder Daten. Ihre Struktur kann durch eine kontextfreie Grammatik formal definiert werden. In Kapitel 7 führen wir kontextfreie Grammatiken als formalen Kalkül ein. Hier zeigen wir im Vorgriff darauf nur, wie dort Bäume eingesetzt werden.

Die Regeln einer kontextfreien Grammatik beschreiben z. B. für eine Programmiersprache, aus welchen Teilen ein Programmkonstrukt besteht.

WhileStatement ::= Expression Statement
Assignment ::= Variable Expression
Statement ::= Assignment

Von den obigen Regeln bedeutet z. B. die erste

Ein WhileStatement besteht aus einem Expression, gefolgt von einem Statement.

Die *„besteht aus"*-Relation einer einzelnen Regel kann man auch als Baum darstellen. Abb. 5.34 zeigt dies für die drei angegebenen Regeln.

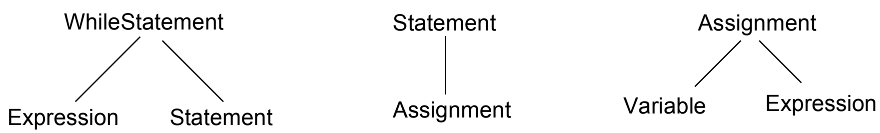


Abbildung 5.34: Bäume repräsentieren Regeln einer kontextfreien Grammatik

Die Regeln einer kontextfreien Grammatik definieren die Struktur jedes syntaktisch korrekten Programmes. Sie wird durch einen Strukturbaum dargestellt. In Abb. 5.35 haben wir einen Strukturbaum für eine while-Schleife angegeben, die eine Zuweisung als Rumpf hat.

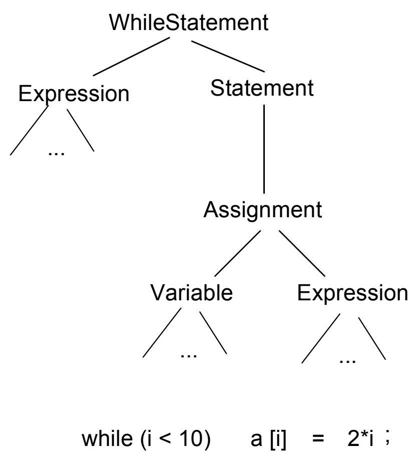


Abbildung 5.35: Strukturbaum zu einer while-Schleife

Die Knoten des Strukturbaumes sind Symbole der Grammatik und repräsentieren hier Programmkonstrukte. Die Kanten, die jeweils von einem Knoten ausgehen und zu Unterbäumen führen, beschreiben die Anwendung einer Regel, wie die Beispiele in Abb. 5.34. Der Baum stellt dar, wie Programmkonstrukte aus kleineren Konstrukten zusammengesetzt sind. In den Grammatikregeln und im Strukturbaum sind die Symbole weggelassen, die für die Struktur nicht wichtig sind, z. B. *while* und *;*. Deshalb wird die Grammatik auch *abstrakte Syntax* genannt, und der zugehörige Baum stellt die abstrakte Struktur des Programms dar.

5.5 Zuordnungsprobleme

Zuordnungsprobleme erfordern, dass Objekte einander so zugeordnet werden, dass bestimmte Randbedingungen erfüllt sind. Solche Aufgaben werden durch ungerichtete Graphen modelliert. Die folgenden Beispiele charakterisieren unterschiedliche Varianten dieses Aufgabentyps:

a) In einem Tennisverein sollen die Vereinsmitglieder für ein Tunier zu Doppelpaarungen zusammengestellt werden. Dabei möchte man nur befreundete Personen miteinander spielen lassen.

b) Eine Gruppe unterschiedlich ausgebildeter Piloten soll so auf Flugzeuge verteilt werden, dass jeder sein Flugzeug fliegen kann.

c) Die Gäste einer Party sollen so an Tischen platziert werden, dass Personen, die sich nicht ausstehen können, an verschiedenen Tischen sitzen.

In jedem der drei Beispiele werden die Randbedingungen durch eine zweistellige Relation über den Objekten repräsentiert, die einander zugeordnet werden sollen:

a) Zwei Personen sind miteinander befreundet.

b) Ein Pilot kann ein bestimmtes Flugzeug fliegen.

c) Zwei Personen können sich nicht ausstehen.

Diese Relation wird durch einen ungerichteten Graphen modelliert. In den Beispielen (a) und (b) werden paarweise Zuordnungen gesucht, während in der Aufgabe (c) die Anzahl der Personen, die an demselben Tisch sitzen, zunächst nicht vorgegeben ist. Das Beispiel (b) unterscheidet sich von (a) dadurch, dass es zwei unterschiedliche Arten von Objekten gibt, Piloten und Flugzeuge, und die Paare aus jeweils einem jeder Art gebildet werden. In allen drei Varianten sucht man nach möglichst günstigen Lösungen: viele befreundete Doppelpaarungen, viele passend bemannte Flugzeuge, wenige Tische, auf die die Gäste verteilt werden müssen.

Als Erstes betrachten wir die Aufgabenvariante der *paarweisen Zuordnungen* (engl.: *matching*). Die Beispiele (a) und (b) gehören dieser Variante an. Ein ungerichteter Graph $G = (V, E)$ modelliert die Randbedingungen. Dabei gibt V die Menge der zuzuordnenden

Objekte an, d. h. in Beispiel (a) die Mitglieder des Tennisvereins. Jede Kante *{a, b}* $\in$ *E* modelliert „*a und b passen zueinander*“, d. h. hier: a und b sind befreundet.

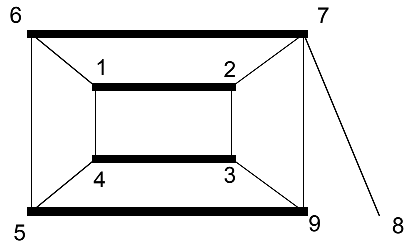


Abbildung 5.36: Matching im ungerichteten Graphen

In Abb. 5.36 gibt ein solcher Graph die Freundschaftsbeziehungen zwischen neun Vereinsmitgliedern an. Als Lösung dieser Zuordnungsaufgabe suchen wir eine möglichst große Zahl von Kanten aus *E*, die keine Knoten gemeinsam haben. Das sind dann die Doppelpaarungen. Wir definieren dafür folgenden Begriff:

Definition 5.22: Maximale Menge unabhängiger Kanten

Sei G = (V, E) ein ungerichteter Graph und M = (V, E') ein Teilgraph von G, dessen Kantenmenge E' möglichst groß ist und dessen Knoten alle höchstens den Grad 1 haben. M ist dann eine ***maximale Menge unabhängiger Kanten*** *oder kurz ein* ***Matching****.* ■

Die stark gezeichneten Kanten in Abb. 5.36 sind ein solches Matching im gesamten Graph. In diesem Beispiel kann man sehr leicht zeigen, dass der Graph aus diesen Kanten die Bedingungen aus Definition 5.22 erfüllt: Es ist ein Teilgraph mit allen Knoten; Knoten 8 hat den Grad 0, alle anderen den Grad 1. Bei insgesamt 9 Knoten kann es höchstens 4 unabhängige Kanten geben.

Die Aufgabenklasse zum Beispiel (b) wird dadurch charakterisiert, dass zwei verschiedene Arten von Objekten, Piloten und Flugzeuge, jeweils einander paarweise zugeordnet werden. Die zugehörigen Graphen nennt man *bipartit*.

Definition 5.23: Bipartiter Graph

Ein Graph G = (V, E) heißt ***bipartit****, wenn V in zwei disjunkte Teilmengen* $V = V_1 \cup V_2$ *zerlegt werden kann, sodass jede Kante zwei Knoten aus verschiedenen Teilmengen verbindet.* ■

Abb. 5.37 zeigt einen bipartiten Graphen mit $V = \{1, 3, 7, 5\} \cup \{2, 4, 6, 8, 9\}$ und einem Matching darin. In Aufgabe (b) würde dann die eine der beiden Mengen die Piloten, die andere die Flugzeuge modellieren. Bei dieser Aufgabenvariante geht man schon von bipartiten Graphen in der Aufgabenstellung aus; man nennt sie auch Heiratsprobleme. Wir geben weitere Beispiele für Paare von Objektarten an, die in solchen Zuordnungsproblemen vorkommen:

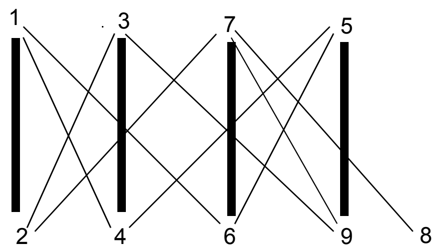


Abbildung 5.37: Bipartiter Graph mit einem Matching

- Mann – Frau
- Aufgabe – Bearbeiter
- Verbraucher – Produkt.

Gemäß Definition 5.23 wird ein Graph schon bipartit genannt, wenn seine Knotenmenge V wie angegeben zerlegt werden *kann*. Sie braucht nicht von vornherein durch unterschiedliche Objektklassen vorgegeben zu sein. Dies wird durch die Abb. 5.36 und 5.37 illustriert: sie zeigen denselben Graphen mit unterschiedlicher Anordnung der Kanten und Knoten. Mit der angegebenen Aufteilung der Knotenmenge ist gezeigt, dass der Graph bipartit ist. Würde man den Graphen z. B. um die Kante *{1, 3}* erweitern, dann gäbe es eine solche Zerlegung der Knotenmenge nicht; der so erweiterte Graph wäre nicht bipartit.

Das dritte Beispiel (c), die Platzierung von Partygästen an Tischen, führt zu einer weiteren wichtigen Variante von Zuordnungsproblemen. Der Graph modelliert wieder eine zweistellige Relation zwischen den Objekten, die einander zugeordnet werden sollen. Anders als in den vorigen Beispielen drückt eine Kante *{a, b}* hier meistens aus, dass *a* und *b nicht einander zugeordnet* werden dürfen. Außerdem werden in dieser Klasse die Objekte zu möglichst wenigen Gruppen zusammengefasst, deren Größe nicht vorgegeben ist. Wir modellieren deshalb das Ergebnis der Zuordnung durch eine Markierung jedes Knotens mit der Nummer der Gruppe, der er zugeordnet wird. In unserem Beispiel (c) ist das dann die Nummer des Tisches, an dem der Partygast platziert wird.

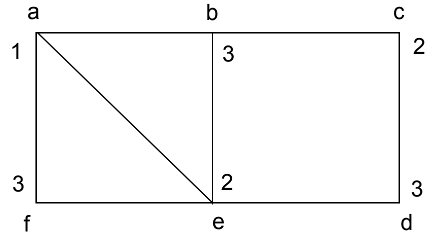


Abbildung 5.38: Konfliktfreie Knotenmarkierung

Abb. 5.38 zeigt einen Graphen, dessen Kanten angeben, welche Partygäste *a, ... , f* sich jeweils nicht ausstehen können. Die Knotenmarkierung gibt an, wie man sie an den Tischen 1, 2, 3 platzieren kann, um Streit zu vermeiden.

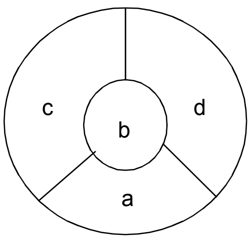
Definition 5.24: Konfliktfreie Knotenmarkierung

Sei G = (V, E) ein ungerichteter Graph, der eine Unverträglichkeitsrelation modelliert: {a, b} ∈ E bedeutet „a und b sind miteinander unverträglich". Eine totale Funktion $M : V \rightarrow \mathbb{N}$ *heißt* ***konfliktfreie Knotenmarkierung,*** *wenn für jede Kante {a, b} ∈ E gilt M (a) ≠ M (b).* ■

Man überzeugt sich leicht, dass die Markierung in Abb. 5.38 konfliktfrei ist. Man kann auch zeigen, dass man eine konfliktfreie Markierung mit weniger als 3 Zahlen nicht finden kann. Denn schon in dem Dreieck *a, b, e* dürfen gleiche Marken nicht vorkommen.

Die berühmteste Aufgabe dieser Klasse ist das so genannte Vier-Farben-Problem. Es bezeichnet die Hypothese, dass vier verschiedene Farben ausreichen, um eine beliebige Staatenkarte so einzufärben, dass zwei Staaten, die ein Stück gemeinsamer Grenze haben, durch unterschiedliche Farben dargestellt werden. Erst 1976 wurde diese Hypothese bewiesen, und zwar durch eine Fallunterscheidung mit mehr als 1000 Fällen, die von einem Computerprogramm entwickelt wurde.

Abb. 5.39 zeigt eine kleine abstrakte Karte mit den Staaten *a, b, c, d* und einen Graphen dazu, dessen Kanten *{x, y}* die Eigenschaft modellieren; *x* und *y* haben eine gemeinsame Grenze.



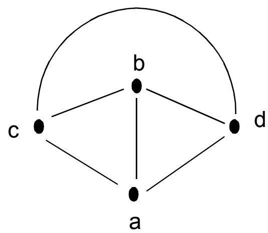


Abbildung 5.39: Staatenkarte mit Unverträglichkeitsgraph

Man kann sich leicht davon überzeugen, dass den vier Staaten (Knoten) vier unterschiedliche Farben zugeordnet werden müssen, da jeder Knoten mit jedem der drei übrigen Knoten verbunden ist.

Das Vierfarbenproblem ist zwar mit dem Einfärben solcher Staatenkarten motiviert, es wird jedoch für die Klasse der *planaren Graphen* formuliert.

Definition 5.25: Planarer Graph

Graph G heißt ***planar****, wenn er in der Ebene so gezeichnet werden kann, dass seine Kanten sich nicht kreuzen.* ■

Modelliert man eine Staatenkarte, sodass eine Kante *{a, b}* die Eigenschaft „*Die Staaten a und b haben eine gemeinsame Grenze*“ beschreibt, so ist der resultierende Graph immer planar.

Das Vierfarbenproblem hat die Untersuchung dieser Grapheigenschaften stark geprägt. Deshalb bezeichnet man die konfliktfreie Markierung häufig auch als *Färbung*, auch wenn sie etwas anderes modelliert, wie z. B. die Tische der Partygäste. Die Anzahl verschiedener Farben, die mindestens nötig sind, um die Knoten eines Graphen konfliktfrei zu markieren, nennt man auch *chromatische Zahl* des Graphen.

Mit der folgenden Tabelle wollen wir zeigen, dass das Prinzip der Färbung von Unverträglichkeitsgraphen ein breites Spektrum an Anwendungen hat:

Knoten	**Kante zwischen a und b**	**Farbe bzw. Marke**
Staat auf der Karte	haben gemeinsame Grenze	Farbe
Partygast	sind unverträglich	Tisch
Kurs	haben gemeinsame Teilnehmer	Termin
Prozess	benötigen dieselben Ressourcen	Ausführungstermin
Variable im Programm	ihre Werte werden gleichzeitig benötigt	Registerspeicher

5.6 Abhängigkeiten

Mit gerichteten Graphen können Abhängigkeiten zwischen Operationen modelliert werden. Jeder Knoten modelliert eine Operation. Er ist häufig mit der Ausführungsdauer markiert. Eine Kante *(a, b)* gibt an, dass die Ausführung von *a* Vorbedingung ist für die Ausführung von *b*. Dafür kann es unterschiedliche Gründe geben:

- *b* benutzt ein Ergebnis von *a*;
- *a* schreibt in eine Speicherstelle, bevor *b* den Wert an der derselben Stelle überschreibt;
- *a* liest von einer Speicherstelle, bevor *b* den Wert an der Stelle überschreibt.

Wenn die Kanten Abhängigkeiten in solch engem Sinne modellieren, müssen die gerichteten Graphen natürlich azyklisch sein. Abb. 5.40 zeigt oben ein Beispiel für einen Abhängigkeitsgraphen. Er modelliert die Operationen von *a* bis *e* und einige Abhängigkeiten zwischen ihnen.

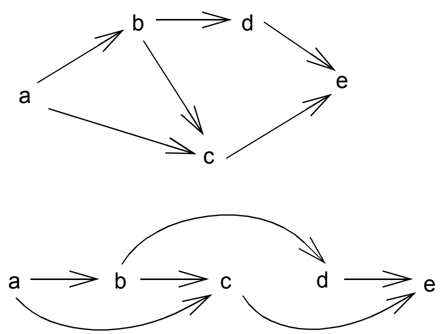


Abbildung 5.40: Abhängigkeitsgraph und lineare Anordnung

Abhängigkeiten zwischen Operationen müssen beachtet werden, wenn man eine Reihenfolge für deren Ausführung bestimmt. Eine solche Ausführungsreihenfolge kann man modellieren, indem man die Knoten des Abhängigkeitsgraphen so anordnet, dass alle Kanten vorwärts, in Ausführungsrichtung zeigen. Die lineare Anordnung der Knoten des Graphen im unteren Teil von Abb. 5.40 modelliert eine sequentielle Ausführung der Operationen.

Im Allgemeinen hat man dabei Freiheiten der Entscheidung, die zur Optimierung zusätzlicher Kriterien genutzt werden können. Hier hätten die Knoten *c* und *d* in der Anordnung vertauscht werden können. Es gibt ein breites Spektrum solcher *Anordnungsaufgaben* (engl. *scheduling*), die sich durch Randbedingungen und Ziefunktionen für die Optimierung unterscheiden. Auch Algorithmen für die Lösung von Scheduling-Aufgaben sind tiefgehend untersucht. Solche Aufgaben kommen in unterschiedlichen Anwendungskontexten vor, z. B.

- Projektplanung mit abhängigen Teilaufgaben;
- abhängige Transaktionen in Datenbanken;
- Anordnung von Code für die parallele Ausweitung von Ausdrücken.

An einem abstrakten Beispiel wollen wir einige wichtige Eigenschaften solcher Abhängigkeitsgraphen und der damit modellierten Aufgaben zeigen.

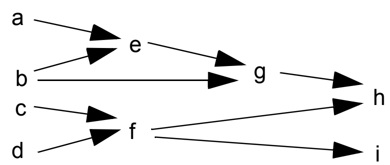


Abbildung 5.41: Abhängigkeitsgraph mit kritischen Pfaden

Abb. 5.41 zeigt einen Abhängigkeitsgraphen mit den Operationen *a* bis *i*. Er könnte z. B. die Abhängigkeiten bei der Auswertung zweier arithmetischer Ausdrücke modellieren:

(a + b) * b - (c / d) und (c / d)^2

Die Operationen *a, b, c, d* stehen für das Laden der Werte von vier Variablen aus dem Speicher. *e* repräsentiert die Addition, *g* die Multiplikation, *f* die Division, *h* die Subtraktion und *i* das Quadrieren. Die Teilausdrücke *b* und *(c / d)* kommen jeweils zweimal vor; deshalb werden ihre Ergebnisse von zwei anderen Operationen benutzt. Die Operationen verknüpfen die Werte ihrer Operanden und liefern ein Ergebnis, das von anderen Operationen benutzt wird. Diese Abhängigkeiten modelliert der Graph.

Es ist eine elementare Aufgabe des Scheduling, herauszufinden, in wie vielen Schritten die Operationen solch eines Abhängigkeitsgraphen abgearbeitet werden können. Dabei nehmen wir vereinfachend an, dass eine Operation in einem Zeitschritt erledigt werden kann und dass beliebig viele Bearbeiter (hier Prozessoren) eingesetzt werden können, damit mehrere Operationen gleichzeitig ausgeführt werden. Dann können die Operationen dieses Graphen trotzdem nicht in weniger als vier Schritten bearbeitet werden. Der Grund dafür ist die Existenz von Pfaden der Länge *4 : (a, e, g, h)* und *(b, e, g, h)*. Sie werden durch folgenden Begriff charakterisiert:

Definition 5.26: Kritischer Pfad

Sei G ein gerichteter, azyklischer Graph. Dann heißen Wege maximaler Länge ***kritische Pfade*** *von G.* ■

Kritische Pfade führen immer von einem Anfangsknoten, mit Eingangsgrad 0, zu einem Ausgangsknoten mit Ausgangsgrad 0.

Die Abarbeitung der Operationen des Graphen durch Bearbeiter kann man grafisch modellieren, indem man die Knoten in einem zweidimensionalen Raster auslegt: die Zeitachse horizontal, die Bearbeiter vertikal. In Abb. 5.42 ist unser Beispielgraph für die Ausführung durch drei Bearbeiter ausgelegt.

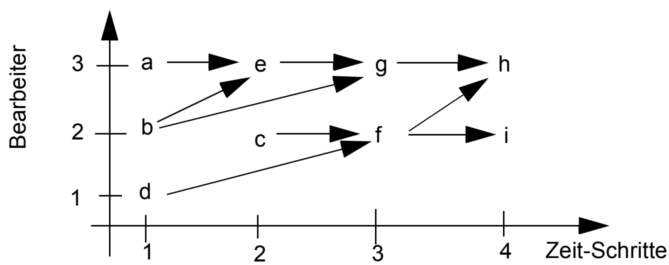


Abbildung 5.42: Anordnung für maximal 3 Bearbeiter

Die Kanten weisen alle in die Richtung der Zeitachse. Knoten, die übereinander am gleichen Zeitpunkt angeordnet sind, werden gleichzeitig von verschiedenen Bearbeitern ausgeführt. Knoten, die nebeneinander in gleicher Höhe angeordnet sind, werden nacheinander von demselben Bearbeiter ausgeführt. Wir stellen fest, dass diese Anordnung insgesamt vier Zeitschritte erfordert. Sie lässt sich nicht verkürzen, da die kritischen Pfade die Länge vier haben. Es gibt auch andere Anordnungen desselben Graphen für 3 Bearbeiter, die 4 Schritte erfordern: z. B. könnte man die Operation *d* auch in den zweiten Schritt verschieben. Wir können Abb. 5.42 als einen Plan verstehen, den ein Übersetzer aufgestellt hat, um Code für die beiden Ausdrücke so zu erzeugen, dass er von einem Prozessor mit 3 Funktionseinheiten in vier Schritten ausgeführt wird.

Abb. 5.43 zeigt zwei verschiedene Anordnungen unseres Beispielgraphen für sequentielle Ausführung durch einen Bearbeiter. Sie benötigen natürlich genauso viele Schritte, wie der Graph Knoten hat.

Häufig produzieren die Operationen, wie in unserem Beispiel der Ausdrucksauswertung, Ergebnisse, die von anderen Operationen benutzt werden. Dann muss jedes Zwischenergebnis so lange gespeichert werden, bis es nicht mehr gebraucht wird. In unserem Beispiel werden solche Werte in Registern des Prozessors gespeichert. Die Anordnung des Graphen modelliert auch, wie viele Register für diesen Zweck benötigt werden: Legt man einen Schnitt durch den Graphen zwischen zwei Operationen, dann gibt die Zahl der geschnittenen Kanten an, wie viele Werte zur weiteren Verwendung in Registern gespeichert werden müssen. In Abb. 5.43 sind nur die Schnitte mit dem maximalen Speicherbedarf angegeben. Wir erkennen, dass man für die obere Anordnung des Graphen 4 Register benötigt, während man bei der unteren Anordnung mit 3 Registern auskommt.

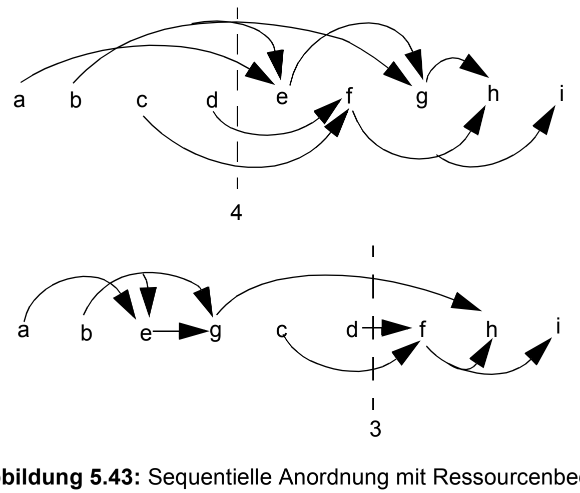


Abbildung 5.43: Sequentielle Anordnung mit Ressourcenbedarf

Es gibt Scheduling-Verfahren, die solche Ressourcenbedarfe minimieren. Solche Modelle können auch für andere Anwendungen eingesetzt werden, z. B. für Produktionsanlagen, wo eine oder mehrere Maschinen Zwischenprodukte in unterschiedlichen Arbeitsgängen weiterverarbeiten.

In vielen praktischen Anwendungen ist die Annahme, dass die Ausführung aller Operationen gleich lange dauert, nicht haltbar: Die Produktionsschritte können je nach Komplexität unterschiedlich lange dauern; auch die Ausführung von Additionen und Multiplikationen benötigt unterschiedlich viele Taktzyklen des Prozessors. Wir können unser Modell verfeinern, indem wir zwei Knotenmarkierungen für die Operation ergänzen:

- Dauer der Operation und
- frühester Abschlusstermin der Operation, berechnet als Dauer der Operation plus spätester Abschlusstermin aller Vorgängerknoten.

Wir haben unseren Beispielgraphen in Abb. 5.44 entsprechend erweitert. Für Graphen mit unterschiedlicher Operationsdauer müssen wir den Begriff des kritischen Pfades neu formulieren:

> *Ein Pfad in einem gerichteten Graphen ist ein **kritischer Pfad**, wenn kein anderer Pfad eine größere Summe der Dauer seiner Operationen hat.*

In Abb. 5.44 ist (c, f, i) der einzige kritische Pfad; seine Operationen dauern insgesamt 10 Einheiten.

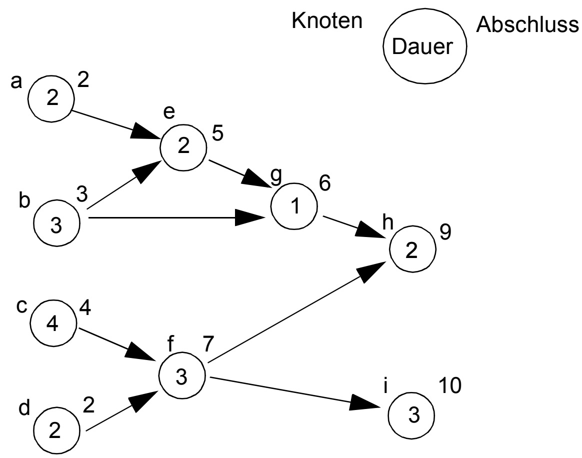


Abbildung 5.44: Operationen mit unterschiedlicher Dauer

Schließlich wollen wir noch darauf hinweisen, dass man bei der Modellierung von Abhängigkeiten auch die Rollen von Knoten und Kanten vertauschen kann. In solch einer *dualen Modellierung* beschreibt ein Knoten das Ereignis, das anzeigt, dass alle Operationen abgeschlossen sind, die Voraussetzung für das Ereignis sind. Wir markieren jeden Knoten mit dem frühest möglichen Abschlusstermin. Eine Kante beschreibt eine Operation mit einer bestimmten Ausführungsdauer. Sie wird als Kantenmarkierung zugeordnet. In Abb. 5.45 haben wir unsere Beispielgraphen in diesem Sinne modelliert.

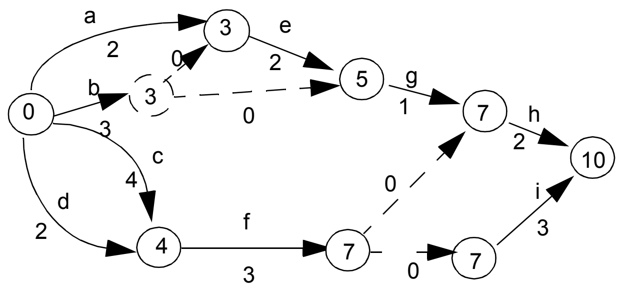


Abbildung 5.45: Duale Modellierung

Für die Transformation eines Abhängigkeitsgraphen in diese duale Modellierung reicht es nicht aus, aus jedem Knoten (jeder Kante) des ursprünglichen Modells eine Kante (einen Knoten) im dualen Modell zu machen: Außerdem müssen im dualen Modell ein Anfangs- und ein Endknoten eingeführt werden. Wenn es im ursprünglichen Modell einen Knoten v gibt, von dem mehrere Kanten zu Knoten w_i ausgehen, dann führen wir im dualen Modell einen neuen Knoten ein. In ihn mündet nun die Kante v und geht je eine neue Kante zu jeder Vorbedingung von w_i aus. Diese neuen Kanten haben die Dauer 0. In Abb. 5.45 sind die beiden neuen Knoten und die vier neuen Kanten gestrichelt gezeichnet. Sie drücken aus, dass die Operation *b* Voraussetzung der beiden Operationen *e* und *g* ist sowie die Operation *f* Voraussetzung für *h* und *i*. Bei dieser Art der dualen Modellierung entstehen im Allgemeinen Multigraphen.

Zum Schluss wollen wir einen Typ von Anwendungen der Modellierung mit Graphen vorstellen, bei denen Abläufe beschrieben werden. Da Abläufe insgesamt oder in Teilen zyklisch sein können, kommen hier allgemeine, gerichtete Graphen zum Einsatz. Ein Knoten modelliert einen Zustand, von dem aus der Ablauf zu unterschiedlichen Nachfolgezuständen fortgesetzt werden kann. Eine Kante *(a, b)* gibt an, dass ein Ablauf vom Zustand *a* in den Nachfolgezustand *b* übergehen kann. Die Entscheidung, zu welchem von mehreren Nachfolgezuständen übergegangen wird, kann im Modell offen bleiben oder z. B. durch Kantenmarkierungen bestimmt werden. Jeder Weg durch solch einen Graphen beschreibt einen potenziellen Ablauf des modellierten Systems. Mit Hilfe des Modells kann man dann Eigenschaften und Aussagen herleiten, die für alle Abläufe gelten. Im Folgenden geben wir Beispiele zu diesem Aufgabentyp an.

Die Abläufe eines Programms oder einer Funktion modelliert man mit so genannten *Programmablaufgraphen*. Sie werden von Übersetzern und Werkzeugen der Software-Technik zur Analyse von Programmeigenschaften benutzt. Abb. 5.46 zeigt ein Programmstück und Abb. 5.47 den zugehörigen Programmablaufgraphen. Jeder Knoten des Graphen repräsentiert einen Grundblock. Das ist eine maximal lange Anweisungsfolge, die eine Sprungmarke nur am Anfang und eine Verzweigung (Sprung) nur am Ende enthalten darf. Die Kanten geben die potenziellen Nachfolger-Blöcke im Ablauf an. Das Modell enthält keine Informationen darüber, wie beim Ablauf über Verzweigungen entschieden wird.

```
ug = 0 ;
og = obereGrenze;                          A

while (ug < = og)                          B
{  mitte = (ug + og) / 2;                  C
   if (a[mitte] == x)
        return Mitte;                      H
   else if (a[mitte] < x)                  D
              ug = mitte + 1;              E
   else       og = mitte - 1;              F
}
return nichtGefunden;                      G
```

Abbildung 5.46: Programmstück mit Angabe der Grundblöcke

In Abb. 5.47 repräsentiert der Knoten *A* den Grundblock, der aus den ersten beiden Zuweisungen besteht. Der Grundblock *B* ist die Schleifenbedingung. Je nach Ergebnis ihrer Auswertung kann auf *C* in den Schleifenrumpf oder *G* hinter die Schleife verzweigt werden. Ein zusätzlicher Knoten *Exit* modelliert den Ausgang aus dem Programmstück, der durch Ausführung der Grundblöcke *H* oder *G* mit den return-Anweisungen erreicht wird.

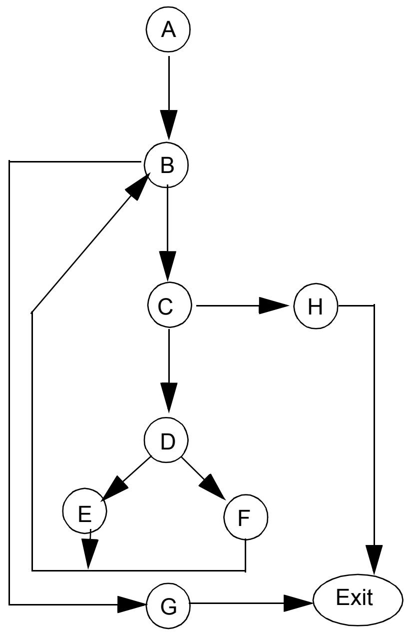


Abbildung 5.47: Programmablaufgraph zu Abb. 5.46

Aus diesem Graphen kann man z. B. die Aussage herleiten: „Jeder Ausführung eines der Grundblöcke *C, D, E, F* geht eine Ausführung von *B* voran.“ Dies ist eine wichtige und grundlegende Eigenschaft von Programmschleifen. Mit Verfahren der Datenflussanalyse werden z. B. Fragen wie „Gibt es einen Weg vom Knoten *k* zum Knoten *Exit*, auf dem die Variable *x* benutzt wird?“ beantwortet. Lautet die Antwort „nein“, dann wäre z. B. eine Zuweisung an *x* am Ende von *k* überflüssig und könnte gestrichen werden. Programmablaufgraphen werden auch zum systematischen Testen von Software eingesetzt: Man versucht eine Menge von Testfällen so zu konstruieren, dass alle Kanten oder alle Knoten beim Ausführen der Tests überdeckt werden.

Ein anderes Modell für Programme sind *Aufrufgraphen*. Sie modellieren Aufrufbeziehungen zwischen Funktionen in Programmen und werden zu ähnlichen Zwecken eingesetzt wie die Programmablaufgraphen. Abb. 5.48 zeigt einen Aufrufgraphen für die Funktionen *a* bis *e*. Die Knoten modellieren die Funktionen des Programms.

Eine Kante *(a, b)* bedeutet, dass die Funktion *a* einen Aufruf der Funktion *b* enthält. *a* könnte also *b* aufrufen. Unter welchen Bedingungen das geschieht, wird nicht modelliert. Wieder kann man aus dem Modell Aussagen ableiten, z. B.

- *e* und *c* sind Funktionen, die sich rekursiv aufrufen können; *a, b, d* sind nicht rekursiv.
- Alle Funktionen sind von *a* aus erreichbar.
- Nehmen wir an, nur in *e* würde eine globale Variable *x* verändert. Dann lassen Aufrufe von *b* und *d* die Variable garantiert unverändert; für *a* und *c* gilt das nicht.

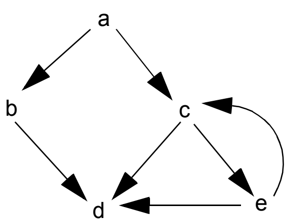


Abbildung 5.48: Aufrufgraph

Solche Aussagen können wichtig sein zum Verstehen, Warten und Optimieren von Programmen.

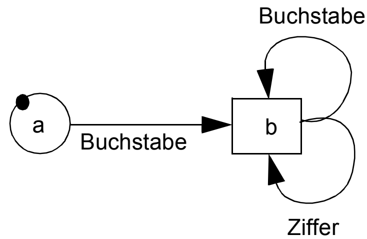


Abbildung 5.49: Endlicher Automat für Bezeichner

Der Graph in Abb. 5.49 stellt einen endlichen Automaten dar. Der Kalkül der endlichen Automaten dient zur Modellierung von Abläufen und zur Definition von Sprachen. Er wird in Kapitel 7 besprochen. Wir betrachten hier nur einen speziellen endlichen Automaten als Beispiel zur Modellierung von Abläufen mit gerichteten Graphen. Jeder Knoten des Graphen modelliert einen Zustand des Automaten. Eine Kante *(a, b)* modelliert einen Übergang von *a* in den Zustand *b*. Die Kanten sind mit Zeichen markiert. Der Automat liest beim Ausführen von Übergängen Zeichen aus der Eingabe: Wenn sich der Automat im Zustand *a* befindet, in der Eingabe ein Zeichen *x* ansteht und eine Kante *(a, b)* existiert, die mit *x* markiert ist, entnimmt der Automat das *x* aus der Eingabe und geht in den Zustand *b* über. Der Automat beginnt seine Abläufe in dem speziell gekennzeichneten Anfangszustand *a*. Sie müssen in einem Endzustand enden, hier der Zustand *b*. Solche endlichen Automaten akzeptieren Folgen der Zeichen, mit denen die Kanten markiert sind. Dieser Automat akzeptiert Folgen aus Buchstaben und Ziffern, die mit einem Buchstaben beginnen. Das ist eine einfache Form von Bezeichnern, wie sie z. B. in der Programmiersprache Pascal verwendet werden.

Übungen

5.1 Knobeleien mit Graphen

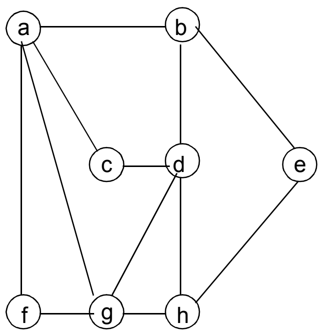


Abbildung 5.50: Ungerichteter Graph

Betrachten Sie den ungerichteten Graphen G in Abb. 5.50.

a) Geben Sie einen Euler-Weg an.

b) Geben Sie einen Hamilton-Kreis an.

c) Zeigen Sie, dass der Graph *G* orientierbar ist.

d) Geben Sie einen Spannbaum von *G* an, den man so wurzeln kann, dass der gewurzelte Baum die Höhe 2 hat. Kennzeichnen Sie die Wurzel in Ihrer Lösung.

e) Geben Sie einen Spannbaum von *G* an, der den Grad 2 besitzt.

f) Geben Sie den Kanten von G Richtungen, sodass der entstehende gerichtete Graph genau zwei Zusammenhangskomponenten besitzt. Geben Sie die Knotenmengen der Zusammenhangskomponenten an.

Geben Sie die Wege zu (a) und (b) als Folge von Knoten an. Zeichnen Sie zu den Teilaufgaben (d) bis (f) jeweils einen eigenen Graphen und verwenden Sie die gleiche Knoten-Anordnung wie in der Abbildung.

5.2 Modellierung von Wegen

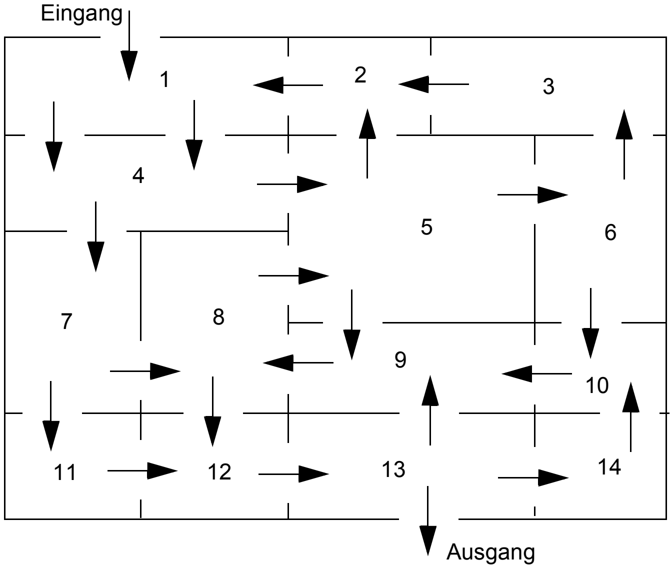


Abbildung 5.51: Ein Irrgarten

Abb. 5.51 stellt den Grundriss eines Irrgartens auf einem Rummelplatz dar. Die Türen in diesem Irrgarten schwingen nur zu einer Seite auf und haben keine Klinken o. ä. Nachdem also ein Besucher die Eingangstür oder eine nachfolgende Tür durchschritten hat und die Tür hinter ihm zugefallen ist, kann der Besucher nicht mehr durch diese Tür zurück. Im Gegensatz zum Beispiel in Abb. 5.16 bleibt die Tür für weitere Durchgänge in der ursprünglichen Richtung benutzbar. Die allgemeinen Sicherheitsbestimmungen für Irrgärten schreiben vor, dass jeder Besucher, der den Irrgarten betritt, wieder den Ausgang erreichen kann.

a) Modellieren Sie den Irrgarten als einen Graphen (Zeichnung des Graphen).
b) Formulieren Sie die allgemeinen Sicherheitsbestimmungen für Irrgärten mit Begriffen der Graphentheorie.
c) Überprüfen Sie anhand der Formulierungen aus (b), ob der angegebene Irrgarten den allgemeinen Sicherheitsbestimmungen entspricht.

5.3 Modellierung von Rechnernetzen

Sie bekommen die Aufgabe, *n* Rechner zu vernetzen. Ihr Auftraggeber verlangt folgende Eigenschaften des Netzwerkes:

E0: Von jedem Rechner aus muss jeder andere Rechner über einen Leitungsweg erreichbar sein.

E1: Auch wenn genau eine Leitung zwischen zwei Rechnern ausfällt, muss jeder Rechner über einen Weg mit jedem anderen Rechner verbunden sein.

E2: An jedem Rechner können maximal 4 Leitungen angeschlossen werden.

Ein Netzwerk lässt sich leicht als Graph darstellen: ein Knoten repräsentiert einen Rechner, eine Kante eine Leitung.

a) Formulieren Sie die Eigenschaften E0, E1 und E2 mit Begriffen der Graphentheorie.
b) Untersuchen Sie die Graphen G_1, G_2 und G_3 auf ihre Tauglichkeit bezüglich der Eigenschaften E0, E1 und E2.

(1) $G_1 = (V, E_1)$ mit $V = 0, ..., n-1$ und $E_1 = \{\{0, i\} \mid 1 \le i \le n-1\}$

(2) $G_2 = (V, E_2)$ mit $E_2 = \{\{i, i+1\} \mid 0 \le i \le n-2\}$

(3) $G_3 = (V, E_3)$ mit $E_3 = \{\{i, (i+1) \bmod n\} \mid 0 \le i \le n-1\}$

5.4 Entscheidungsbaum

Christian und Heike spielen folgendes Spiel: Heike wählt eine Zahl aus der Menge $\{1, 2\}$. Danach wählen beide Spieler abwechselnd eine Zahl aus der Menge $\{1, 2, 3\}$ mit der Einschränkung, dass die vom Gegner zuvor gewählte Zahl nicht wählbar ist. Ein Spieler gewinnt, wenn durch seine Wahl die Summe aller gewählten Zahlen den Wert 6 annimmt. Übersteigt die Summe 6, verliert der Spieler.

a) Beschreiben Sie das Spiel durch einen Entscheidungsbaum.
b) Wer gewinnt, wenn beide Spieler optimal spielen?

5.5 Modellierung mit Graphen, Graphfärbung

Beschaffen Sie sich eine Karte mit den Staaten Europas.

a) Modellieren Sie zu dieser Karte einen Graphen, welcher die direkten Autoverbindungen zwischen den einzelnen Ländern darstellt. Zusätzlich zu den normalen Landverbindungen existiert noch
   – eine Autobrücke zwischen Dänemark und Schweden,
   – ein Autotunnel zwischen Großbritanien und Frankreich,
   – eine Autofähre zwischen Großbritanien und Irland
b) Kennzeichnen Sie vorkommende Schnittknoten und Brückenkanten.
c) Wie viele unterschiedliche Farben benötigt man mindestens, um diese Karte so einzufärben, dass so verbundene Länder nicht die gleiche Farbe haben?

5.6 Zuordnungsprobleme

Claudia veranstaltet eine Cocktailparty und hat als Begrüßung unter anderem 7 verschiedene Cocktails vorbereitet: Bloody Mary, Daiquiri, Green Spider, Harvey Wallbanger, Ladykiller, Tequila Sunrise und Whiskey Sour Madison. Zu ihrer Party lädt sie ihre 6 Freunde Andrea, Bernd, Christiane, Dennis, Elizabeth und Frank ein. Diese haben jedoch in puncto Cocktails spezielle Vorlieben:

- Andrea mag Bloody Mary und Daiquiri.
- Bernd mag Green Spider, Bloody Mary und Harvey Wallbanger.
- Christiane mag Daiquiri, Harvey Wallbanger und Green Spider.
- Dennis mag Ladykiller, Bloody Mary, Tequila Sunrise und Daiquiri.
- Elizabeth mag Harvey Wallbanger, Daiquiri, Bloody Mary und Green Spider.
- Frank mag Daiquiri, Green Spider und Bloody Mary.

a) Stellen Sie die Vorlieben der einzelnen Gäste mit Hilfe eines Graphen dar.
b) Können alle Gäste eines ihrer Lieblingsgetränke bekommen? Modellieren und lösen Sie die Aufgabe mit einem bipartiten Graphen.

5.7 Abhängigkeitsgraph für eine Kaffeemaschine

Von einem Knopfdruck zur Bestellung bis zur Ausgabe von Kaffee in einen Becher geschieht einiges intern bei einer Kaffeemaschine. Für diesen Vorgang werden einige Aktionen in der Kaffeemaschine vordefiniert. Die Aktionen werden in einer bestimmten Reihenfolge ausgeführt, bis der Vorgang abgeschlossen ist.

Die für eine Luxus-Kaffeemaschine benötigten Aktionen seien wie folgt definiert.

- A_1: Kaffebohnen mahlen, Dauer: 20 Sekunden
- A_2: Wasser kochen, Dauer: 30 Sekunden
- A_3: Kaffeepulver in Filterfach füllen, Dauer: 2 Sekunden
- A_4: Kochwasser eintropfen, Dauer: 60 Sekunden
- A_5: Becher aufstellen, Dauer: 2 Sekunden
- A_6: Kaffee ausgeben, Dauer: 2 Sekunden
- A_7: gebrauchtes Pulver wegwerfen, Dauer: 1 Sekunde.

a) Modellieren Sie die Abhängigkeiten zwischen den Aktionen und ihre Ausführungsreihenfolgen durch einen Graphen. Benutzen Sie dazu die zwei Knotenmarkierungen. Dauer der Aktion und frühester Abschlusstermin wie in Abb. 5.44.
b) Geben Sie zu dem Graphen aus Teil (a) einen kritischen Pfad als Folge von Knoten an.
c) Wie sieht der Graph aus, wenn die Knoten sequentiell angeordnet werden, sodass alle Kanten vorwärts zeigen?