Graphen*

Hans Jürgen Ohlbach

23. März 2018

Empfohlene Vorkenntnisse: Relationen

Inhaltsverzeichnis

*Dieser Text ist Teil einer Sammlung von Miniskripten zur Einführung in die Informatik. Er ist in erster Linien für Nichtinformatiker gedacht, kann aber natürlich auch als erste Einführung für Informatiker nützlich sein.

1 Einleitung

Graphen gehören zu den wichtigsten Datenstrukturen in der Informatik. Sie werden aber auch in der Mathematik thematisiert. Es gibt eine ganze Forschungsrichtung darüber, die *Graphentheorie*. In diesem Miniskript kann daher nur ein erster Eindruck gegeben werden. Insbesondere sollen die Begriffe eingeführt werden, die für die Informatik relevant sind.

Ein *Graph* besteht aus *Knoten* (engl. *node* oder *vertex*) und *Kanten* (engl. *edge* oder *link*). Jede Kante verbindet genau zwei Knoten[1].

Formell: Wenn K eine Menge von Knoten ist, dann ist $V \subseteq K \times K$ eine Menge von Kanten.
$G = (K, V)$ ist dann ein Graph.

Einige Beispiele aus der Realität sind:

- Das Straßennetz kann man als Graph modellieren. Die Knoten sind die Kreuzungen und die Straßen sind die Kanten.
- Das Telefonnetz kann man als Graph modellieren. Die Knoten sind die Verteilerstationen und die Telefone. Die Kanten sind die Kabel.
- Das Internet kann man auch als Graph modellieren. Die Knoten sind die Router und die angeschlossenen Computer. Die Kanten sind die Verbindungen (Kabel, Funk usw.) dazwischen.
- Auch eine Website kann man als Graph modellieren. Die Knoten sind die HTML-Seiten. Die Kanten sind die Hyperlinks zwischen den Seiten.
- Sogar Verwandtschaftsbeziehungen kann man als Graph modellieren. Die Knoten sind die Leute, und die Kanten modellieren die *verwandt-mit* Beziehung.

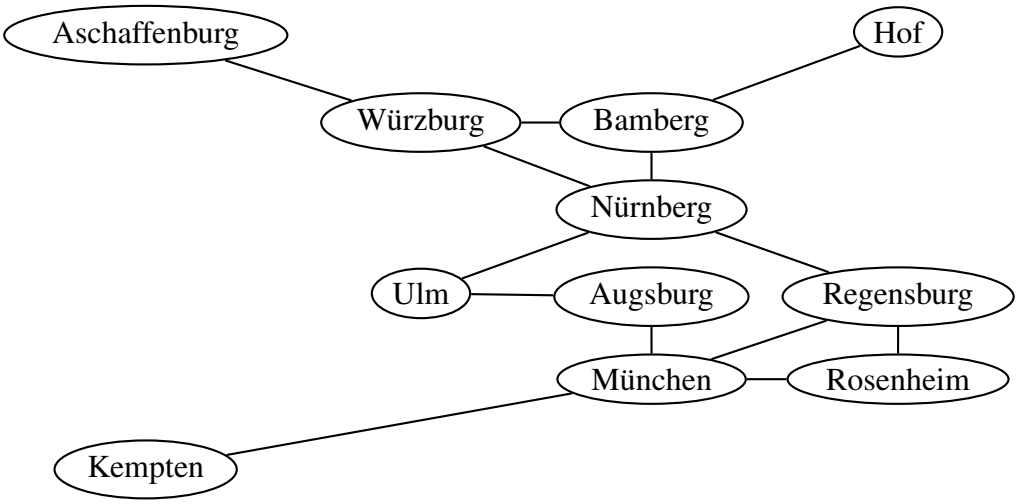
Das letzte Beispiel verdeutlicht eine schon etwas abstraktere Sichtweise auf Graphen: Jede *zweistellige* Relation r über einer Menge M entspricht einem Graphen: nämlich einfach $G = (M, r)$. Umgekehrt kann man jedem Graphen eine zweistellige Relation zwischen den Kanten zuordnen: die *Kantenrelation*. Zwei Knoten, die durch eine Kante verbunden sind, stehen in dieser Kantenrelation.

Z.B. ist die *teilt* Relation $\mid$ auf natürlichen Zahlen eine solche zweistellige Relation: $2 \mid 6, 3 \mid 6, 4 \mid 12$ usw. Sie entspricht dann einem *unendlichen* Graphen über den natürlichen Zahlen.

Endliche Graphen können sehr einfach visualisiert werden. Die Knoten werden als Kreise oder so ähnlich gemalt, und die Kanten als Verbindungslinien dazwischen.

[1] Es gibt auch *Hypergraphen*, bei denen Kanten mehr als zwei Knoten verbinden.

Das folgende **Beispiel:** stellt die Eisenbahnverbindungen in Bayern extrem vereinfacht als Graph dar.



2 Arten von Graphen

Graphen gibt es in enorm vielen Variationen. Die wichtigsten davon werden in diesem Abschnitt vorgestellt.

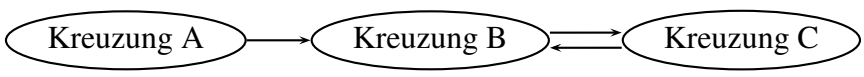
2.1 Ungerichtete Graphen

Dies sind Graphen, in denen die Kanten keine Richtung haben. Die Kantenrelation ist daher symmetrisch. Wenn man von Knoten A nach Knoten B kommt, dann kommt man auch von Knoten B nach Knoten A. Der obige „Eisenbahngraph“ ist ungerichtet. Es gibt zwischen allen Städten Verbindungen in beide Richtungen.

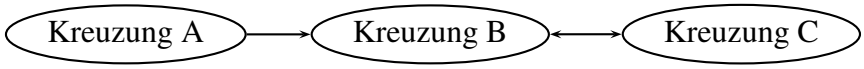
2.2 Gerichtete Graphen

Dies sind Graphen, in denen die Kanten eine Richtung haben. Die Kantenrelation ist dann nicht symmetrisch. Es gibt natürlich auch Situationen, wo einzelne Kanten gerichtet sind, und andere ungerichtet. Ein typisches Beispiel sind Straßennetze. Die gerichteten Kanten sind die Einbahnstraßen, und die ungerichteten Kanten sind die in beide Richtungen befahrbaren Straßen. Graphen mit gerichteten und ungerichteten Kanten kann man jedoch auf einfache Weise in einen vollständig gerichteten Graphen umwandeln: Für Straßennetze z.B. modelliert modelliert man jede einzelne Fahrbahn als Kante. Straßen, die in beiden Richtungen befahrbar sind, entsprechen dann zwei Kanten zwischen den Knoten. Graphisch malt man die gerichteten Graphen mit Pfeilchen zwischen den Knoten.

Im folgenden Bild wäre die linke Kante eine Einbahnstraße

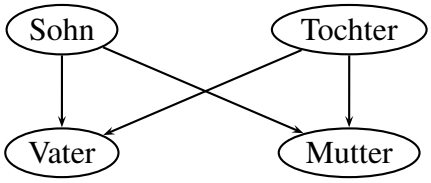


Vereinfacht kann man das auch so malen:



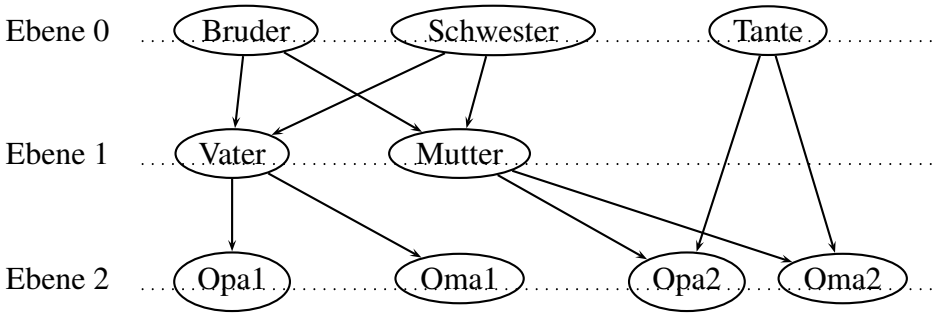
In gerichteten Graphen kann man zwischen **inneren Knoten** und **Blattknoten** unterscheiden. Blattknoten sind solche, die keine ausgehende Kante haben. Innere Knoten sind alle anderen Knoten. In Verkehrsnetzen z.B. wären die Enden von Sackgassen die Blattknoten. In Telefonnetzen wären die Telefone die Blattknoten.

Von besonderem Interesse sind auch *gerichtete azyklische Graphen* (engl. Directed Acyclic Graph, oder DAG). Es sind gerichtete Graphen, in denen es keinen Zyklus (siehe Abschnitt 3.3) gibt. Ein Beispiel wäre der DAG für die *ist-Kind-Von* Relation. Schon die Beziehung zwischen Sohn, Tochter und Vater und Mutter ist ein DAG:



In DAGs ist es möglich, den Knoten Ebenen zuzuordnen. Ebene 0 sind die Knoten, die keinen Vorgängerknoten haben. Für alle anderen Knoten bestimmt sich die Ebene durch den längsten Pfad (siehe Abschnitt 3.3), der zu diesem Knoten führt.

Das folgende Beispiel illustriert das.



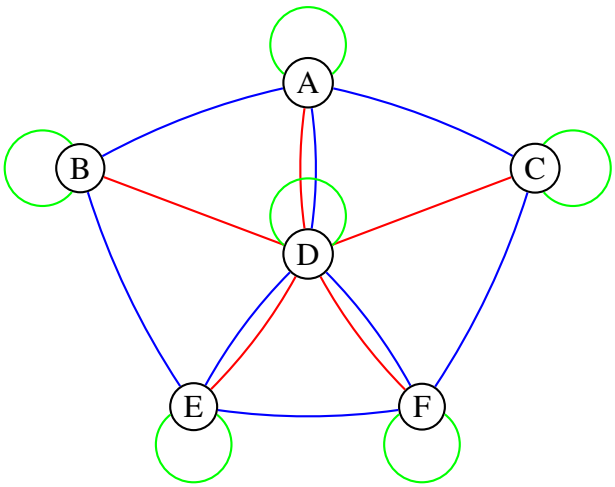
2.3 Orientierte Graphen

Ein orientierter Graph ist ein gerichteter Graph, bei dem zwischen zwei Knoten *höchstens eine* Kante existiert. Ein Verkehrsnetz, welches nur aus Einbahnstraßen besteht, könnte als orientierter Graph modelliert werden.

2.4 Multigraphen

Die bisherigen Graphen modellieren *eine* Relation zwischen den Knoten. Entsprechend gab es auch nur eine ungerichtete Kante bzw. maximal zwei gerichtete Kanten zwischen den Knoten. Will man mehr als eine Relation zwischen den Knoten modellieren, braucht man auch mehr Kanten zwischen den Knoten. Ein Beispiel mit zwei Relationen wäre die Straßen- und Eisenbahnverbindungen zwischen Städten.

Das folgende Bild zeigt einen ungerichteten Multigraphen für 3 symmetrische Relationen, eine symbolisiert durch die blauen Kanten, eine durch die roten Kanten und eine durch die grünen Kanten. Diese Relation ist zusätzlich noch *reflexiv*, d.h. $r(x, x)$ gilt für alle x. Reflexive Relationen werden durch Kanten, die den Knoten mit sich selbst verbinden symbolisiert.



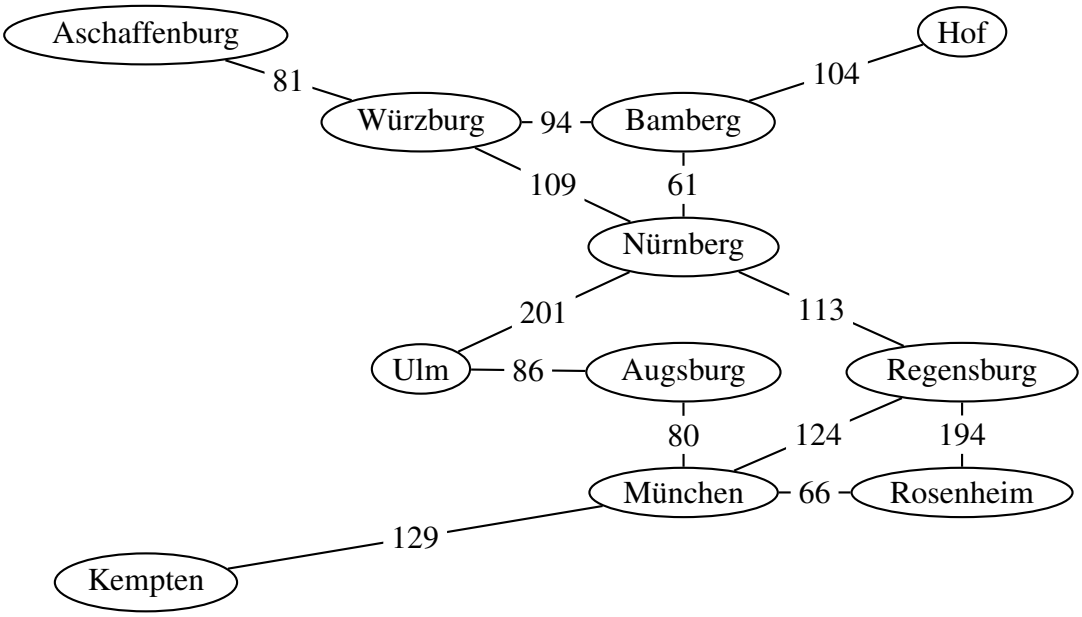
2.5 Gelabelte Graphen

In gelabelten Graphen tragen die Kanten zusätzliche Informationen außer den beiden Endknoten. Der obige Multigraph war genau genommen schon ein gelabelter Graph. Die Zusatzinformation war die Farbe. Sie stand für die zugrunde liegende Relation.

2.6 Gewichtete Graphen

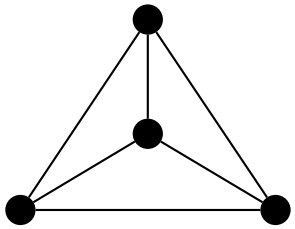
Eine besondere Klasse von Kantenlabels sind Zahlen. Sie geben den Kanten ein *Gewicht*. Das Gewicht kann aber ganz unterschiedliche Bedeutung haben. In Verkehrsnetzen könnte das Gewicht die Straßenlänge bedeuten. Im Internet könnte das Gewicht die Übertragungskapazität in Bit pro Sekunde bedeuten. Im Telefonnetz könnte das Gewicht die Anzahl von Gesprächen sein, die gleichzeitig übertragen werden können.

Im folgenden Graphen sind die Kantengewichte die ungefähren Straßenkilometer zwischen den Städten.

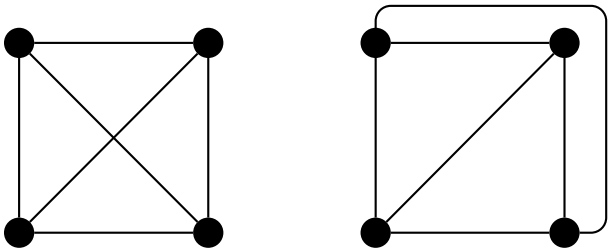


2.7 Planare Graphen

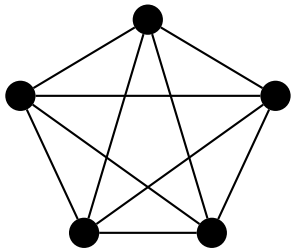
Ein Graph ist *planar* falls er zweidimensional gezeichnet werden kann, ohne dass sich Kanten überschneiden, wie in folgendem Bild:



Der linke Graph im folgenden Bild ist nicht planar gezeichnet. Die beiden diagonalen Kanten überschneiden sich. Er könnte aber planar gemacht werden, so wie im rechten Bild dargestellt.



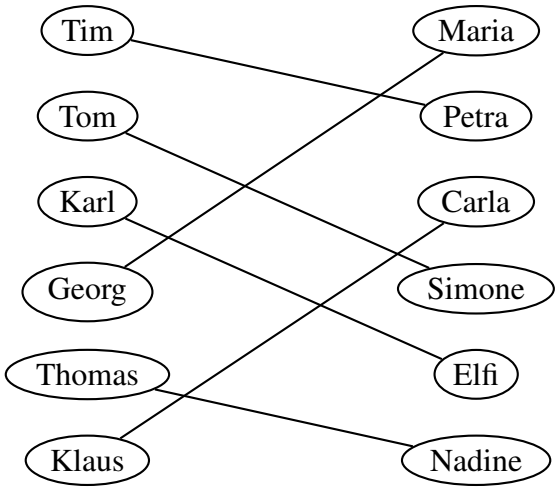
Es gibt aber auch Graphen, die auf keine Weise planar gezeichnet werden können, z.B. der folgende:



2.8 Bipartite Graphen

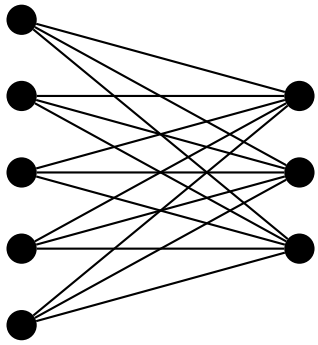
Bei diesen Graphen kann man die Knoten in zwei disjunkte Mengen F und M aufteilen, so dass es nur Kanten zwischen diesen Mengen gibt.

Das einprägsamste Beispiel dazu ist eine Tanzstunde, bei der nur Männer mit Frauen tanzen, und es keine gleichgeschlechtlichen Tanzpaare gibt. Im folgenden Graphen kann man sich vorstellen, wie die Teilnehmer bei Damenwahl aufeinander zugehen.



Ein bipartiter Graph

Es können in den beiden Knotenmengen natürlich auch unterschiedlich viele Knoten sein. Jeder Knoten kann dabei auch mehr als eine Kante in die andere Menge haben, wie in folgendem Beispiel.



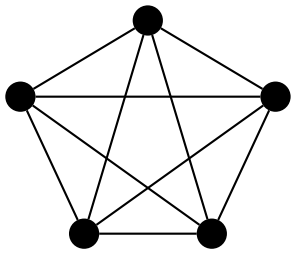
Auch ein bipartiter Graph

3 Eigenschaften von Graphen

3.1 Vollständigkeit

Ein Graph ist *vollständig* falls jeder Knoten mit jedem Knoten verbunden ist.

Dieser Graph ist vollständig.



Fehlt eine Kante, dann ist er unvollständig.

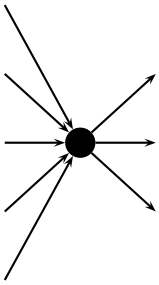
Anzahl Kanten in einem vollständigen Graphen:
Angenommen, der Graph hat die Knoten $K_1, \ldots, K_n$. Wenn er vollständig ist, dann gibt es von
K_1 Kanten zu $K_2, \ldots, K_n$, das sind $n-1$ Kanten
K_2 Kanten zu $K_3, \ldots, K_n$, das sind $n-2$ Kanten
$\vdots$
K_{n-1} eine Kante zu K_n, das ist 1 Kante.

Insgesamt sind es $1 + 2 + \ldots + n - 1 = ((n-1) \cdot n)/2 = (n^2 - n)/2$ Kanten.

Bei vielen Algorithmen, die mit Graphen arbeiten, hängt die Zeitkomplexität von der Anzahl von Kanten ab. Wenn man nicht weiß, wie viele Kanten der Graph hat, muss man vom schlimmsten Fall ausgehen. Das ist dann der vollständige Graph. Man bekommt daher einen Faktor $\mathcal{O}(n^2)$ für die Zeitkomplexität.

3.2 Grad eines Knoten

Die Anzahl Kanten, die mit einem Knoten verbunden sind, ist der *Grad des Knotens*. Bei gerichteten Graphen kann man noch unterscheiden zwischen *Eingangsgrad* und *Ausgangsgrad*. Der Eingangsgrad ist die Anzahl gerichteter Kanten, die auf den Knoten zeigen. Der Ausgangsgrad ist die Anzahl gerichteter Kanten, die von einem Knoten weg zeigen.



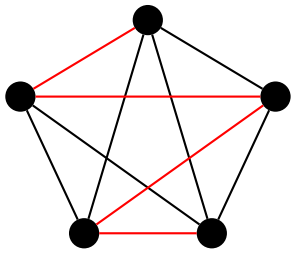
Eingangsgrad: 5 Ausgangsgrad: 3

Reguläre Graphen: Ein Graph ist *regulär* falls alle Knoten den gleichen Grad haben. Ist der Grad k, dann spricht man auch von einem k-regulären Graph.

3.3 Pfad in einem Graphen

Ein *Pfad* ist eine Folge von Kanten, die von Knoten zu Knoten gehen.

Die roten Kanten in folgendem Graph bilden einen Pfad.

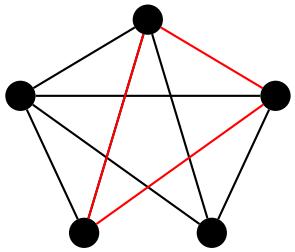


Man kann einen Pfad entweder als Folge von Kanten $K_1, \ldots, K_n$ angeben, wobei der Endknoten von K_i mit dem Anfangsknoten von K_{i+1} übereinstimmt.

Man kann den Pfad auch als Folge von Knoten $N_1, \ldots, N_n$ angeben, wobei es Kanten zwischen Knoten N_i und N_{i+1} gibt.

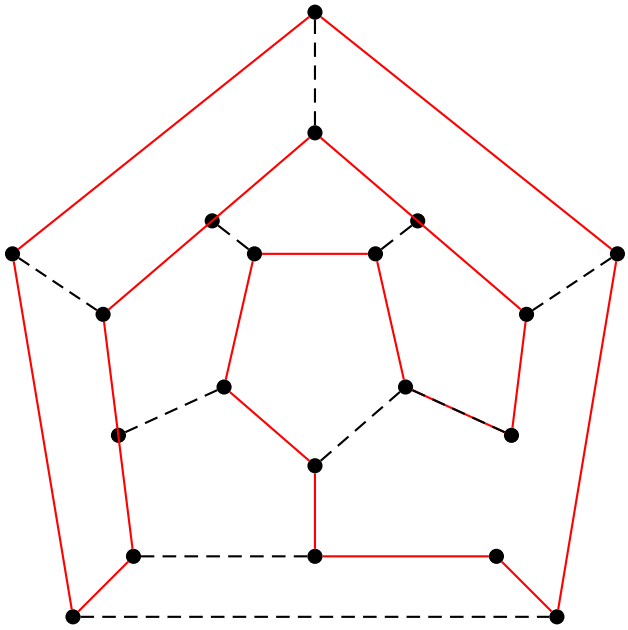
Zyklus (Kreis) im Graph Ein *Zyklus* oder *Kreis* in einem Graphen ist ein *geschlossener Pfad*, der von einem Knoten ausgeht und wieder im selben Knoten endet.

Im nächsten Graph bildet der rote Pfad einen Zyklus.



Hamiltonkreis:
Ein Hamiltonkreis ist ein geschlossener Pfad in einem Graphen, der *jeden Knoten genau einmal* enthält.

Das folgende Bild zeigt einen Hamiltonkreis als rote Kanten.



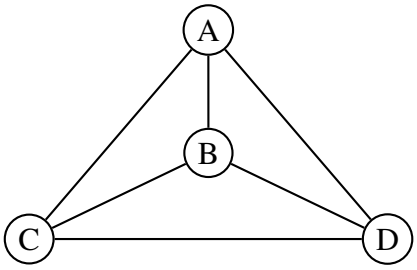
Ein Hamiltonkreis

Die Frage, ob es in einem gegebenen Graphen einen Hamiltonkreis gibt, oder nicht, gehört zu den sog. NP-vollständigen Problemen, für die bisher kein effizienter Algorithmus bekannt ist.

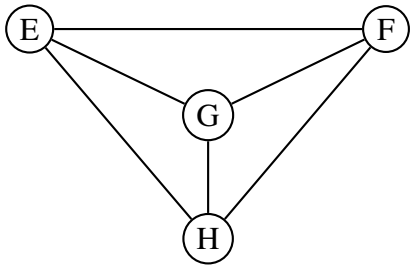
3.4 Zusammenhangskomponenten

Manche Graphen zerfallen in disjunkte Teilgraphen. Das Straßennetz in Großbritannien, z.B. zerfällt in den Teil auf der Hauptinsel (England, Schottland Wales) und dem Teil in Nordirland. Es gibt keine Straße von Nordirland zur Hauptinsel. Die Teile eines Graphen, zwischen denen es keinen Pfad gibt,

bezeichnet man als *Zusammenhangskomponente*. Zur Verwaltung von Zusammenhangskomponenten eines Graphen eignet sich eine *Union-Find Struktur*.



Zusammenhangskomponente 1



Zusammenhangskomponente 2

4 Datenstrukturen für Graphen

Endliche Graphen kann man auf unterschiedliche Weise im Rechner speichern. Welche davon die beste ist, hängt sehr vom Algorithmus ab, der auf dem Graphen arbeitet.

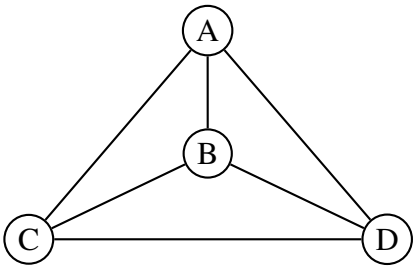
4.1 Adjazenzlisten

In dieser Darstellung speichert man einfach für jeden Knoten die Information über dessen Umgebung. Es gibt dabei verschiedene Möglichkeiten:

In ungewichteten Graphen speichert man einfach die Liste der Nachbarknoten. Die Kanten sind dabei implizit durch die Information über die Nachbarknoten gegeben. Bei gerichteten Graphen muss man zwischen den ausgehenden Kanten und den eingehenden Kanten unterscheiden.

In gewichteten Graphen braucht man eine explizite Datenstruktur für die Kanten. Jeder Knoten bekommt dann die Liste der Kanten. Jede Kante speichert den Start- und Endknoten sowie das Gewicht.

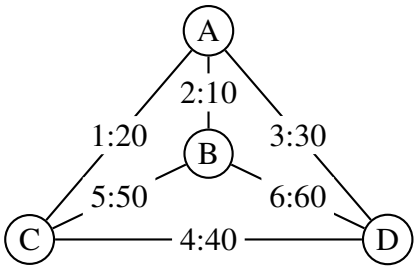
Für den linken Graphen würde die Adjazenzliste so aussehen:



A $\rightarrow$ B,C,D
B $\rightarrow$ A,C,D
C $\rightarrow$ A,B,D
D $\rightarrow$ A,B,C

Natürlich enthalten die Listen keine Information über das Layout des Graphen wenn er gezeichnet wird. Außer für die Zeichenroutinen selbst sind diese Layout-Daten für fast alle Algorithmen nicht relevant.

Für den folgenden gewichteten Graphen wurden die Kanten durchnummeriert. 1:20, z.B. soll bedeuten, dass die Kante 1 das Gewicht 20 hat.



Die Adjazenzlisten sehen dann so aus:

A	→	1,2,3
B	→	2,5,6
C	→	1,4,5
D	→	3,4,6

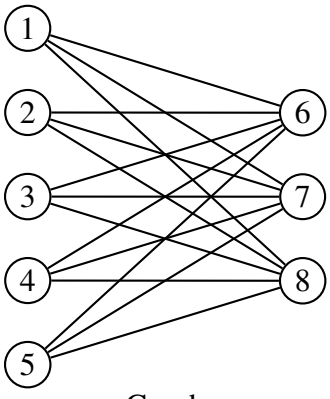
Kante	Startknoten	Endknoten	Gewicht
1	A	C	20
2	A	B	10
3	A	D	30
4	C	D	40
5	B	C	50
6	B	D	60

Bei ungerichteten Graphen ist es natürlich egal, welchen Knoten man als Start- und welchen Knoten man als Endknoten bezeichnet.

4.2 Adjazenzmatrix

Eine Adjazenzmatrix für einen Graphen mit n Knoten ist eine quadratische $n \times n$-Matrix mit Booleschen Einträgen. Die Knoten müssen dafür durchnummeriert sein. Ein Eintrag 1 bei Spalte i und Zeile j bedeutet, dass es eine Kante zwischen Knoten i und Knoten j gibt.

Die Adjazenzmatrix für den Graphen unten steht rechts.



Graph

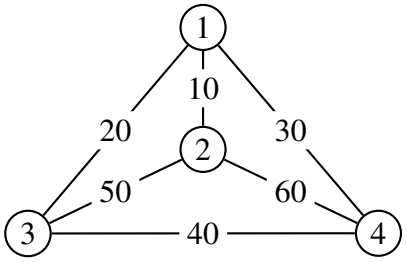
	1	2	3	4	5	6	7	8
1	0	0	0	0	0	1	1	1
2	0	0	0	0	0	1	1	1
3	0	0	0	0	0	1	1	1
4	0	0	0	0	0	1	1	1
5	0	0	0	0	0	1	1	1
6	1	1	1	1	1	0	0	0
7	1	1	1	1	1	0	0	0
8	1	1	1	1	1	0	0	0

Adjazenzmatrix

Bei ungerichteten Graphen ist die Matrix immer symmetrisch zur Diagonalen.

Auch gewichtete Graphen kann man mit einer Adjazenzmatrix repräsentieren. Anstelle von 0 und 1 als Einträge in die Matrix kann man die Gewichte eintragen. Eine 0 als Gewicht bedeutet dann eben „keine Kante“.

Ein Beispiel:



Gewichteter Graph

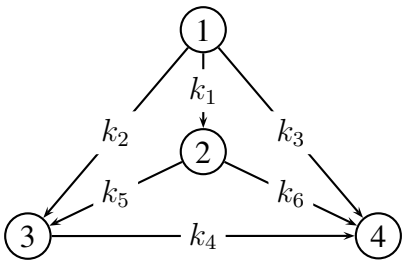
	1	2	3	4
1	0	10	20	30
2	10	0	50	60
3	20	50	0	40
4	30	60	40	0

Adjazenzmatrix

4.3 Inzidenzmatrix

Bei einer Inzidenzmatrix entsprechen die Spalten den Kanten und die Zeilen den Knoten. Ein nicht-Null Eintrag in Zelle (Kante i, Knoten j) der Matrix bedeutet, dass j ein Endknoten der Kante i ist. Bei gerichteten Graphen kann man die Richtung mit -1 und +1 als Matrixeintrag kennzeichnen.

In der Inzidenzmatrix des gerichteten Graphen unten bedeutet +1 Startknoten der Kante und -1 Endknoten der Kante.



Gerichteter Graph

	k_1	k_2	k_3	k_4	k_5	k_6
1	+1	+1	+1	0	0	0
2	-1	0	0	0	+1	+1
3	0	-1	0	+1	-1	0
4	0	0	-1	-1	0	1

Inzidenzmatrix

Offensichtlich ist in jeder Spalte genau eine +1 und eine -1. Jede Kante hat halt einen Startknoten und einen Endknoten. (In Hypergraphen wäre das anders).

5 Algorithmische Probleme mit Graphen

In den letzten Jahrzehnten wurde eine ganz Reihe von Problemen untersucht, bei denen die Daten als Graphen modelliert werden können, und Algorithmen die auf diesen Graph-Datenstrukturen arbeiten.

Einige der wichtigsten sind:

- Suche den kürzesten Weg zwischen zwei Knoten in einem gewichteten Graphen. Jedes Navigationsgerät muss dieses Problem lösen. Algorithmen dazu sind der Dijkstra-Algorihtmus und der A*-Algorithmus.
- Suche einen zyklusfreien Teilgraph in einem gewichteten Graphen, der alle Knoten enthält, und dessen Summe der Gewichte minimal ist (sog. Spanning Tree). Beim Design von Telefonverbindungen, Wasserleitungen, Abflussleitungen usw. mit minimalen Kosten muss man solche Probleme lösen. Kruskal und Prim haben Algorithmen dazu entwickelt.
- Für gewichtete Graphen, deren Gewichte Durchflusskapazitäten darstellen, möchte man den maximal möglichen Fluss zwischen zwei Knoten berechnen. Für das Internet bedeutet das z.B. die maximale Übertragungskapazität zwischen zwei Routern zu berechnen, unter Ausnutzung aller parallelen Leitungen. Ein Algorithmus dafür ist der Algorithmus von Dinic.
- Berühmt/berüchtigt ist auch das Handlungsreisendenproblem (Travelling Salesman Problem). Hier geht es darum, einen optimalen Hamiltonkreis zu finden, d.h. einen mit kürzester Gesamtweglänge.

Für diese Probleme siehe das Miniskript *GewichteteGraphen*.

Weitere Probleme sind u.A. ein Test, ob ein Graph planar gezeichnet werden kann. Überhaupt, das automatisierte Zeichnen von Graphen ist eine äußert komplexe Aufgabe, und ebenfalls ein interessantes Teilgebiet der Informatik.

6 Ausblick: Bäume

Eine spezielle, aber wichtige Art von Graphen sind *Bäume*. Über Bäume gibt es ebenfalls viel zu sagen. Dies wird im Miniskript zu *Bäumen*, sowie in den Miniskripten zu den speziellen Baumtypen behandelt: *ALC-Bäume*, *B-Bäume*, *R-Bäume*

Stichwortverzeichnis