Einführung in die Graphentheorie

Alexandra Schwartz

Julius-Maximilians-Universität Würzburg
Institut für Mathematik
Emil-Fischer-Straße 30
97074 Würzburg

schwartz@mathematik.uni-wuerzburg.de
http://www.mathematik.uni-wuerzburg.de/∼schwartz

Vorlesungsskript
Wintersemester 2012/13

Stand: 16. August 2013

Vorwort

Das vorliegende Skript ist als Grundlage der zweistündigen Vorlesung „Einführung in die Graphentheorie“ im Rahmen des Unitags WS 2012/13 gedacht. Es richtet sich also an Schüler und andere Leser ohne größere Vorkenntnisse im Bereich Mathematik oder Informatik und soll einen ersten Einblick in das weite Feld der Graphentheorie geben. Daher wird an manchen Stellen in Definitionen und Beweisen zugunsten anschaulicher Argumente auf höchste mathematische Präzision verzichtet. Auch werden nicht alle auftretenden Konzepte in größtmöglicher Allgemeinheit eingeführt und manche relevante Themen, wie etwa die Komplexität von Algorithmen oder Probleme in Netzwerken, werden komplett ausgeklammert. Wer sich diesbezüglich mehr wünscht, dem sei besonders das Buch [9] ans Herz gelegt.

Das Skript basiert größtenteils auf vier Büchern, die im Folgenden kurz vorgestellt werden sollen. Aus Gründen der Lesbarkeit ist im Skript allerdings nicht an jeder Stelle seperat kenntlich gemacht, aus welchem der Bücher ein Resultat, Beispiel oder Beweis stammt.

Das Buch „Graphentheoretische Konzepte und Algorithmen“ [9] von Sven Oliver Krumke und Hartmut Noltemeier[1] bietet eine sehr gute und umfangreiche Einführung in die Graphentheorie. In dem Buch wird großer Wert auf eine saubere Notation und präzise Beweise gelegt, es ist daher für Leser ohne eine gewisse Vertrautheit mit mathematischen Texten keine leichte Lektüre. Dafür gibt es einen Einblick in viele Bereiche der Graphentheorie, immer mit einem Blick auf Lösungsalgorithmen und deren Implementierung. Viele Definitionen und Beweise im vorliegenden Skript stammen aus diesem Buch.

Im Gegensatz zum vorherigen richtet sich das Buch „Graphen und Netzwerkoptimierung“ [1] von Christina Büsing hauptsächlich an Schüler und ist ohne Vorkenntnisse verständlich. Hier liegt der Fokus mehr auf einem anschaulichen Verständnis von Konzepten und Zusammenhängen. Dieses Buch deckt ebenfalls ein breites Spektrum an Fragestellungen und viele zentrale Resultate ab, auch wenn bei dem einem oder anderen der (dann im allgemeinen eher längliche) Beweis fehlt. Abgerundet wird alles durch eine Vielzahl an Beispielen und Übungsaufgaben, mit denen man sein Verständnis jederzeit überprüfen kann. Einige der Beispiele und Aufgaben finden sich auch in diesem Skript wieder.

„Kombinatorische Optimierung erleben“ [6] von den Herausgebern Stephan Hußmann und Brigitte Lutz-Westphal ist eine Sammlung von Kapiteln verschiedener Autoren, die sich mit Themen der kombinatorischen Optimierung beschäftigen. Der Großteil dieser Fragestellungen gehört in den Bereich der Graphentheorie. Das Buch richtet sich hauptsächlich an Lehrpersonal und gibt eine Anleitung, wie man sich im Selbstversuch, ausgehend von

[1] Prof. Noltemeier hatte bis zu seiner Emeritierung 2008 einen Lehrstuhl für Graphentheorie am Institut für Informatik an der Universität Würzburg inne.

alltäglichen Problemen, diesen Fragestellungen nähern und, über den einen oder anderen Irrweg, schließlich Lösungen finden kann. Es ist aber auch für Schüler und Studierende gut verständlich und gibt einen schönen Einblick darin, wie mathematische Sätze und Algorithmen entstehen. Einige der Beispiele im vorliegenden Skript stammen aus diesem Buch.

Das letzte Werk, dem dieses Skript vor allem seinen Aufbau verdankt, ist das Vorlesungsskript „Operations Research“ [7] von Christian Kanzow. Es widmet sich nur in der zweiten Hälfte der Graphentheorie, zeigt dort aber, im Gegensatz zu den vorherigen Büchern, immer wieder die Zusammenhänge zwischen graphentheoretischen Problemen und einer Klasse von Optimierungsproblemen, den sogenannten linearen Programmen, auf. Diesen Zusammenhang werden wir im vorliegenden Skript der nötigen Vorkenntnisse wegen nicht beleuchten. Wer sich dafür interessiert, dem sei das „Operations Research“-Skript empfohlen, in dessen erster Hälfte lineare Programme und das Simplex-Verfahren behandelt werden. Es richtet sich allerdings an Studierende der Mathematik, setzt also eine gewisse Vertrautheit mit mathematischer Notation und Beweiskonzepten voraus.

Inhaltsverzeichnis

Grundlagen

Aussagenlogik

In der Mathematik geht es häufig darum Aussagen zu beweisen. Eine *Aussage* ist zum Beispiel „Alle Schafe in Deutschland sind weiß.“ und wird häufig mit einem einzelnen Buchstaben, zum Beispiel A, abgekürzt. Eine Aussage ist immer wahr oder falsch, eine weitere Möglichkeit gibt es nicht. Die obige Aussage zum Beispiel ist falsch, denn es gibt in Deutschland auch schwarze Schafe.

Zu einer Aussage A gibt es immer auch die *Verneinung* dieser Aussage, bezeichnet mit $\neg A$. Bei der Verneinung von Aussagen muss man ein wenig aufpassen, so ist die Verneinung der obigen Aussage „Nicht alle Schafe in Deutschland sind weiß.“ oder „Es gibt Schafe in Deutschland, die nicht weiß sind.“ und *nicht* „Kein Schaf in Deutschland ist weiß.“ Es gilt immer: Ist die Aussage A wahr, so ist $\neg A$ falsch, und umgekehrt.

Oft muss man, um eine Aussage zu beweisen, d.h. um zu zeigen, dass sie wahr ist, mehrere Aussagen verknüpfen. Zwei wichtige Verknüpfungen sind die *Verundung* und die *Veroderung*. Sind A und B zwei Aussagen, so sagt man A *und* B sind wahr, wenn beide wahr sind, und A *oder* B sind wahr, wenn mindestens eine von beiden wahr ist, vergleiche auch Tabelle 1. Für A und B schreibt man $A \wedge B$ und für A oder B ist die Notation $A \vee B$. Betrachtet man zum Beispiel die Aussagen A: „Dolly ist ein Schaf.“ und B: „Dolly ist eine Ziege.“, so ist $A \wedge B$ auf jeden Fall falsch und $A \vee B$ ist richtig, falls Dolly ein Schaf oder eine Ziege ist.

A	B	$A \wedge B$
w	w	w
w	f	f
f	w	f
f	f	f

(a) A und B

A	B	$A \vee B$
w	w	w
w	f	w
f	w	w
f	f	f

(b) A oder B

Tabelle 1: Wahrheitstafeln für $A \wedge B$ und $A \vee B$

Eine weitere zentrale Verknüpfung ist die *Implikation*. Seien A und B zwei Aussagen. Man sag A *impliziert* B oder *aus* A *folgt* B, wenn dann, wenn die Aussage A wahr ist, immer auch die Aussage B wahr ist, vergleiche Tabelle 2. Man schreibt dafür $A \implies B$. Ist also A die Aussage „Dolly ist ein Schaf.“ und B die Aussage „Dolly ist ein Säugetier.“, so gilt $A \implies B$, aber nicht $A \impliedby B$. Man darf also Implikationen bzw. Folgepfeile nicht einfach umdrehen.

Es ist jedoch möglich sie umzudrehen, wenn man gleichzeitig die Aussagen verneint, d.h. statt $A \Longrightarrow B$ kann man auch sagen $\neg B \Longrightarrow \neg A$, vergleiche auch Tabelle 2. Man nennt dies *Kontraposition*. Offensichtlich folgt aus $\neg B$: „Dolly ist kein Säugetier." sofort $\neg A$: „Dolly ist kein Schaf."

A	B	$\neg A$	$\neg B$	$A \Longrightarrow B$	$\neg B \Longrightarrow \neg A$
w	w	f	f	w	w
w	f	f	w	f	f
f	w	w	f	w	w
f	f	w	w	w	w

(a) $(A \Longrightarrow B)$ in Vergleich zu $(\neg B \Longrightarrow \neg A)$

A	B	$A \Longrightarrow B$	$A \Longleftarrow B$	$A \Longleftrightarrow B$
w	w	w	w	w
w	f	f	w	f
f	w	w	f	f
f	f	w	w	w

(b) $A \Longleftrightarrow B$

Tabelle 2: Wahrheitstafel für Äquivalenz und Kontraposition

Gelten tatsächlich beide Richtungen, d.h. $A \Longrightarrow B$ und $A \Longleftarrow B$, so nennt man die Aussagen A und B *äquivalent* und schreibt $A \Longleftrightarrow B$. Man sagt auch A ist *genau dann* wahr, wenn B wahr ist. Ein Beispiel für äquivalente Aussagen wäre „Dolly ist ein Schaf." und „Dolly gehört zur Säugetiergattung Ovis."

Beweiskonzepte

Einer der zentralen Bestandteile der Mathematik sind Sätze (sowie Lemmas und Korollare) und Beweise. Ein *Satz* ist meistens von der Struktur: Wenn gewisse Aussagen $A_1, A_2, \ldots$ wahr sind, dann ist auch die Aussage B wahr. Die Aussagen A_i nennt man hierbei *Voraussetzungen*. Folgt ein Satz mehr oder weniger sofort aus einem anderen, so nennt man ihn oft *Korollar*. Ist ein Satz hingegen für sich alleine nicht so wichtig, aber nützlich um weitere Sätze zu beweisen, so bezeichnet man ihn häufig als *Lemma*.

In einem *Beweis* geht es dann darum die in einem Satz behauptete Implikation $A_1 \wedge A_2 \wedge \ldots \Longrightarrow B$ nachzuweisen, indem man solange Hilfsaussagen $C_1, C_2, \ldots$ zwischenschiebt, bis man eine Kette aus bekannten Implikationen

$$A_1 \wedge A_2 \wedge \ldots \Longrightarrow C_1 \Longrightarrow C_2 \Longrightarrow \ldots \Longrightarrow B$$

gefunden hat. Dies nennt man auch einen *direkten Beweis*. Wollen wir zum Beispiel den Satz

Wenn ihre Mutter ein weißes Schaf (A_1) *und ihr Vater ein schwarzes Schaf war* (A_2)*, weibliche Lämmer von verschiedenfarbigen Eltern weiß und männliche Lämmer von verschiedenfarbigen Eltern schwarz sind* (A_3) *und Dolly ein weibliches Schaf ist* (A_4)*, so ist Dolly ist ein weißes Schaf* (B)*.*

beweisen, so funktioniert dies wie folgt: $A_1 \wedge A_2 \Longrightarrow C_1$ mit C_1: „Dollys Eltern sind verschiedenfarbig." Weiter gilt $C_1 \wedge A_3 \Longrightarrow C_2 \vee C_3$ mit C_2: „Dolly ist ein weibliches weißes Schaf." und C_3: „Dolly ist ein männliches schwarzes Schaf." Schließlich folgt $(C_2 \vee C_3) \wedge A_4 \Longrightarrow B$.

Manchmal ist es einfacher statt einen Satz direkt zu beweisen eine *Widerspruchsannahme* zu treffen. Wollen wir $A \Longrightarrow B$ zeigen, so können wir auch annehmen, dass $A \wedge (\neg B)$ wahr ist und versuchen daraus etwas zu folgern, von dem wir wissen, dass es nicht gilt. Wenn uns das gelingt, wissen wir, dass unsere Voraussetzungen falsch gewesen sein müssen, d.h. dass A und $\neg B$ nicht gleichzeitig wahr sein können. Dies bedeutet aber, dass immer, wenn A wahr ist, auch B wahr sein muss, also $A \Longrightarrow B$.

Im obigen Beispiel hätten wir annehmen können, dass Dolly ein schwarzes Schaf ist. Zusammen damit, dass ihre Eltern verschiedenfarbig sind, wäre dann gefolgt, dass Dolly ein männliches Schaf ist, ein Wiederspruch zu Voraussetzung A_4.

Versuchen wir nun den folgenden Satz zu beweisen:

Wenn Dollys Urururururururur-Großeltern Schafe waren (A_1) *und die Kinder von Schafen auch immer Schafe sind* (A_2)*, dann ist Dolly ein Schaf* (B)*.*

Der Beweis ist klar: Wenn Dollys Urururururururur-Großeltern Schafe waren (A_1) dann waren nach Voraussetzung A_2 auch Dollys Urururururururur-Großeltern Schafe. Dieses Argument können wir jetzt noch 9 mal wiederholen und erhalten dann die gesuchte Aussage B. Man nennt dies einen *Induktionsbeweis*. Formal besteht dieser immer aus drei Teilen: einer Induktionsvoraussetzung, einer Induktionsannahme und einem Induktionsschluss.

Im obigen Beispiel ist die Induktionsvoraussetzung, das Dollys Urururururururur-Großeltern Schafe waren. Die Induktionsannahme besagt, dass man von einer Generation von Dollys

Vorfahren weiß, dass sie Schafe waren. Der Induktionsschluss ist schließlich, dass dann auch die nachfolgende Generation aus Schafen bestehen muss. Interessanterweise kann man so nicht nur zeigen, dass Dolly ein Schaf ist, sondern auch, dass alle Kinder, Enkel, Ur-Enkel,... von Dolly Schafe sein werden.

Man kann sich einen Induktionsbeweis auch wie eine beliebig hohe Leiter vorstellen, die man hochklettern will: Die Induktionsvoraussetzung sorgt dafür, dass man es schafft auf die erste Sprosse zu kommen, einer Induktionsannahme besagt, dass man es geschafft hat eine Stufe zu erreichen und der Induktionsschluss schließlich garantiert, dass man dann *immer* auch auf die nächste Sprosse klettern kann.

Es gibt auch Sätze, die besagen, dass gewisse Aussagen A, B, C, D äquivalent sind, oder genauer, dass je zwei von diesen Aussagen äquivalent sind. Versucht man nun alle möglichen Implikationen zwischen zwei dieser Aussagen zu beweisen, so ist man eine Weile beschäftigt. Stattdessen kann man aber auch versuchen eine möglichst kleine Menge von Implikationen zu beweisen, so dass sich alle anderen aus diesen kombinieren lassen. Es reicht zum Beispiel die folgende Implikationskette zu zeigen:

$$A \Longrightarrow B \Longrightarrow C \Longrightarrow D \Longrightarrow A.$$

Aus dieser Kette folgen auch alle anderen Implikationen wie zum Beispiel $D \Longrightarrow C$. Man nennt dies einen *Ringschluss*.

Notation

Zahlenräume

$\mathbb{N}$	die natürlichen Zahlen (1, 2, 3, ...)
$\mathbb{R}$	die reellen Zahlen

Mengen

$\{x\}$	die Menge bestehend aus dem Element x
$\emptyset$	die leere Menge, d.h. eine Menge ohne Elemente
$\|S\|$	die Mächtigkeit oder Kardinalität der Menge S, d.h. die Anzahl der Elemente von S
$S_1 \cap S_2$	der Schnitt von S_1 und S_2, d.h. die Elemente, die in beiden Mengen enthalten sind
$S_1 \cup S_2$	die Vereinigung von S_1 und S_2, d.h. alle Elemente, die in mindestens einer der beiden Mengen enthalten sind
$S_1 \setminus S_2$	die Menge aller Elemente von S_1, die nicht in S_2 enthalten sind
$S_1 \subseteq S_2$	S_1 ist eine Teilmenge S_2, d.h alle Elemente von S_1 sind auch Elemente von S_2
$S_1 \subsetneq S_2$	S_1 ist eine echte Teilmenge von S_2, d.h. $S_2 \setminus S_1$ ist nicht leer

Vektoren und Matrizen

$x \in \mathbb{R}^n$	ein Vektor im Vektorraum $\mathbb{R}^n$, auch geschrieben als $x = (x_1, x_2, \ldots, x_n)$
x_i	die i-te Komponente von x
$A \in \mathbb{R}^{n \times m}$	eine Matrix im Vektorraum $\mathbb{R}^{n \times m}$, auch geschrieben als

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix}$$

$a_{i,j}$	die i,j-te Komponente von A, d.h. das j-te Element in der i-ten Zeile (Merkregel für die Reihenfolge der Indizes: „Zeilen zuerst, Spalten später“)

1 Definitionen und Beispiele

In diesem Kapitel wollen wir formal definieren, was wir unter einem Graphen verstehen und einige erste elementare Eigenschaften von Graphen kennenlernen. Bevor wir dies jedoch tun, betrachten wir einige Beispiele, die zu graphentheoretischen Problemen führen. Diese werden später an geeigneter Stelle wieder aufgegriffen, präzisiert und in Teilen auch gelöst.

1.1 Beispiele

Beispiel 1.1 (Rubik's Würfel) Viele haben einen zuhause rumliegen, weniger können ihn lösen: den Rubik's Würfel. Angenommen, man würde einen solchen Würfel auseinander und beliebig wieder zusammen bauen. Wäre er dann immer noch lösbar? Und wenn ja, wieviele Drehungen würde man mindestens brauchen? ♢

Beispiel 1.2 (Telefonkette) Die Lehrerin einer Schulklasse plant einen Wandertag. Da die Abfahrtszeit des hierfür gemieteten Busses sich kurzfristig ändern kann, beschließt sie eine Telefonkette zu organisieren. Dafür lässt sie sich von allen Schülern sagen, von welchen anderen Schülern sie die Telefonnummer kennen. Wie soll sie nun festlegen, wer wen anrufen soll? Und wieviele Anrufe müssen mindestens getätigt werden? ♢

Beispiel 1.3 (Magnetschwebebahn) Angenommen, in Deutschland sollten Magnetschwebebahnen gebaut werden. Da diese sehr teuer sind, hat man einige große Städte ausgewählt, die mit dieser neuen Technologie verbunden werden sollen, vergleiche Abbildung 1.1. Auf Grund der hohen Kosten kann man jedoch auch hier nicht jede dieser Städte mit allen anderen verbinden. Trotzdem soll es möglich sein (mit Umsteigen) zwischen allen Städten hin- und her zu reisen. Wie lässt sich dies möglichst kostengünstig realisieren und wieviele Strecken müssen mindestens gebaut werden? ♢

Beispiel 1.4 (Routenplanung) Ein Autofahrer befindet sich am Würzburger Bahnhof und sucht einen möglichst kurzen Weg zum Mathematischen Institut. Welche Route sollte er, natürlich unter Beachtung der Verkehrsregeln wie z.B. Einbahnstraßen, nehmen? In Würzburg ist das Problem noch relativ einfach zu lösen, aber wie bestimmt man kürzeste Routen in größeren Städten oder bei längeren Reisen? Denken wir an Navigationssysteme für Autos, so erwarten wir, dass eine Route innerhalb von Sekunden berechnet wird. Es müssen also nicht nur sehr große Probleme gelöst werden, dies soll auch in möglichst kurzer Zeit geschehen. ♢

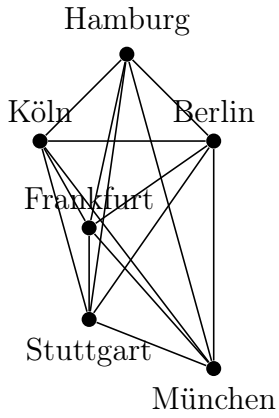
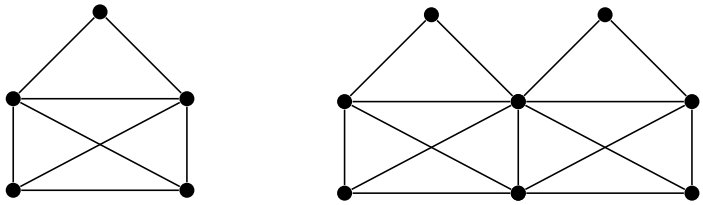


Abbildung 1.1: Wo soll die Bahn fahren?

Beispiel 1.5 (Hochzeitsproblem) Stellen Sie sich vor, Ihnen gehörte eine Heiratsagentur. Sie stehen also vor der Aufgabe eine Menge junger Frauen und Männer zu verheiraten, wobei nicht jede Ehe möglich ist (unterschiedliche Interessen, ...). Wie bekommen Sie möglichst viele Ihrer Kunden unter die Haube? ◊

Beispiel 1.6 (Das Haus vom Nikolaus) Wer kennt es nicht, das Haus vom Nikolaus, vergleiche Abbildung 1.2(a). Aber warum kann man eigentlich das Haus vom Nikolaus zeichnen ohne einmal mit dem Stift abzusetzen, bekommt aber Probleme, sobald der Weihnachtsmann beschließt nebendran einzuziehen, vergleiche Abbildung 1.2(b)? Und ist es beim Haus vom Nikolaus egal, in welcher Ecke man anfängt zu zeichnen? ◊



(a) Das ist das Haus vom Niko-laus

(b) und ne-ben-dran vom Weih-nachts-mann

Abbildung 1.2: Was der Nikolaus mit Graphentheorie zu tun hat

Beispiel 1.7 (Archäologie) Der Ägyptologe Sir William Matthew Flinders Petrie (1853–1942) entdeckte im Jahr 1894 in Oberägypten, genauer Naqada, eine ca. 3.000 Gräber umfassende Nekropole. Um zu rekonstruieren, in welcher Reihenfolge die Gräber entstanden waren, entwickelte er folgenden Ansatz: Den meisten Gräbern waren größere Mengen verschiedener Keramik beigegeben. Diese teilte er in 9 Kategorien auf, von denen in einem Grab entweder Exemplare enthalten waren oder nicht. Auf Basis dieser Daten bestimmte er für je zwei Gräber einen Abstand, der umso kleiner war, je mehr gleiche Keramik ent-

halten war. Seine zentrale Idee war nun, dass zwei Gräber, die einen kleinen Abstand, also ähnliche Keramikbeigaben, haben, auch zeitlich nahe beieinander entstanden sein müssen. Deshalb versuchte er die Gräber so anzuordnen, dass die Summe der Abstände zwischen aufeinanderfolgenden Gräbern möglichst gering wurde. Wie findet man eine solche Reihenfolge? ◊

Beispiel 1.8 (Leiterbahnen auf Platinen) Auf Platinen müssen Bauteile (Widerstände, Kondensatoren, Transistoren, ...) möglichst platzsparend aufgebracht und gemäß einem vorgegebenen Schaltplan durch Leitungen miteinander verbunden werden. Da die Leiterbahnen sich nicht kreuzen dürfen, müssen die Lage der Bauteile und Wege der Leitungen sorgfältig geplant werden. Aber woher weiß man, wie man die Bauteile anordnen muss und welche Schaltungen überhaupt umsetzbar sind? ◊

Beispiel 1.9 (Stundenplanerstellung) Zu Beginn eines jeden Schuljahres stellt sich die gleiche, oft unbeliebte, Frage: Wie findet man in Zeiten von Lehrer- und Raummangel einen Stundenplan, vergleiche Abbildung 1.1, bei dem die Schüler möglichst früh nach Hause können, d.h. möglichst wenige Freistunden haben? Hierbei ist natürlich zu beachten, dass es nicht möglich ist, dass ein Lehrer mehrere Klassen gleichzeitig unterrichtet oder umgekehrt eine Klasse gleichzeitig Unterricht bei mehreren Lehrern hat. ◊

Zeit	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
1. Stunde					Graphen-
2. Stunde					theorie
⋮					

Tabelle 1.1: Wie entsteht ein Stundenplan?

Wir haben nun viele Beispiele gesehen, die sich, auch wenn man es vielleicht noch nicht sieht, mit Hilfe von Graphen formulieren und lösen lassen.

Als Nächstes wollen wir definieren, was genau wir unter einem Graphen verstehen. Je nachdem, ob Übergänge zwischen zwei Orten/Zuständen immer in beide Richtungen möglich sind oder nicht unterscheidet man zwischen gerichteten und ungerichteten Graphen.

1.2 Gerichtete Graphen

Wir beginnen mit gerichteten Graphen. Gerichtete Graphen braucht man zum Beispiel, wenn man Routenplanung betreiben möchte und dabei Einbahnstraßen beachten muss.

Definition 1.10 *Ein* gerichteter Graph[1] *ist ein Paar* $G = (V, E)$ *mit einer endlichen*

[1] In manchen Zusammenhängen ist es interessant Graphen mit unendlich vielen Knoten oder Kanten zu betrachten, diese heißen dann *unendliche Graphen*. Wir werden dies in der Vorlesung nicht tun, betrachten also nur *endliche Graphen*.

Menge $V \neq \emptyset$ und einer endlichen Menge $E \subseteq \{(u,v) \mid u,v \in V, u \neq v\}$. Die Elemente von V heißen Knoten *(auf englisch: vertex) von G. Die Elemente von E heißen* Kanten *oder* Pfeile *(auf englisch: edge) von G. Eine Kante[2] $e \in E$ ist also von der Form $e = (u,v)$ mit gewissen $u,v \in V$, $u \neq v$, wobei u als* Anfangsknoten *und v als* Endknoten *von e bezeichnet wird.*

Zwei Kanten $e_1, e_2 \in E$ mit $e_1 = (u,v)$ und $e_2 = (u,v)$ heißen parallel*; zwei Kanten $e_1, e_2 \in E$ mit $e_1 = (u,v)$ und $e_2 = (v,u)$ heißen* invers*. Enthält G keine parallelen Kanten, so heißt G* einfach*.*

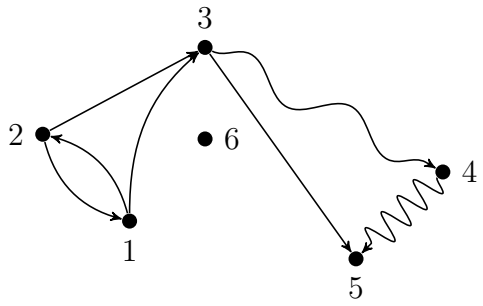


Abbildung 1.3: Ein gerichteter Beispielgraph G

Zur Illustration betrachten wir den Graphen $G = (V, E)$ mit Knoten $V = \{1,2,3,4,5,6\}$ und Kanten $E = \{(1,2),(2,1),(1,3),(2,3),(3,4),(3,5),(4,5)\}$, siehe Abbildung 1.3.

Wenn man einen Graphen zeichnet, ist es egal, welche Form die Knoten haben und wo sie platziert sind. Ebenso können die Kanten gerade Linien sein oder gekrümmt. Wichtig ist nur, dass die Struktur des Graphen richtig ist, d.h. dass er die richtige Anzahl an Knoten hat und die Kanten zwischen den richtigen Knoten verlaufen.

Um die Lesbarkeit einer Zeichnung zu erhöhen, versucht man allerdings meistens die folgenden Punkte zu berücksichtigen: Knoten sollten nicht zu dicht beieinander liegen, da sonst Kanten dazwischen schlecht erkennbar sind. Gleichzeitig sollte der gesamte Graph aber nicht zu groß werden. Kanten zeichnet man wenn möglich als gerade Linien und gekrümmt nur dann, wenn dadurch Schnittpunkte mit anderen Kanten oder Knoten, welche nicht die Endknoten sind, vermieden werden. Ebenso versucht man durch eine geschickte Platzierung der Knoten derartige Schnittpunkte zu vermeiden, auch wenn dies nicht immer möglich ist. Auf dieses Problem werden wir in Kapitel 9 zurückkommen.

Je nach Zusammenhang können die Knoten eines Graphen mit Zahlen, Koordinaten, Buchstaben aus diversen Alphabeten oder einer Mischung davon bezeichnet werden. In der Vorlesung werden wir Knoten in Bildern meistens mit Zahlen und in Beweisen mit Buchstaben wie u, v, s, t bezeichnen.

Um in einem Graphen die Beziehung verschiedener Elemente zueinander beschreiben zu können, führen wir die folgende Sprechweise ein.

[2] Allgemeiner kann man auch zulassen, dass Anfangs- und Endknoten einer Kante gleich sind, also $e = (v,v)$. Man nennt e dann eine *Schleife*. Wir werden in der Vorlesung aber nur schleifenfreie Graphen betrachten.

Definition 1.11 *Es sei $G = (V, E)$ ein gerichteter Graph. Ein Knoten $v \in V$ und eine Kante $e \in E$ heißen* inzident*, wenn v der Anfangs- oder Endknoten von e ist. Zwei Kanten $e_1, e_2 \in E$ heißen* benachbart *oder* adjazent*, wenn es einen Knoten $v \in V$ gibt, der mit e_1 und e_2 inzidiert. Zwei Knoten $u, v \in V$ heißen* benachbart *oder* adjazent*, wenn es eine Kante $e \in E$ gibt, die mit u und v inzidiert.*

Ob zwei Knoten/Kanten benachbart sind oder ein Knoten und eine Kante inzidieren hängt also nicht von der Richtung der Pfeile im Graphen ab.

In dem Graphen G aus Abbildung 1.3 sind zum Beispiel die Knoten 5 und 3 benachbart/adjazent, 5 und 1 hingegen nicht. Die Kanten $(1, 2)$ und $(2, 3)$ sind benachbart/adjazent, der Knoten 4 und die Kante $(4, 5)$ sind inzident.

Nachdem wir nun definiert haben, wann zwei Knoten benachbart sind, wollen wir noch eine Notation einführen, mit der man ausdrücken kann, mit wie vielen anderen Knoten ein Knoten $v \in V$ benachbart ist.

Definition 1.12 *Es sei $G = (V, E)$ ein gerichteter Graph und $v \in V$ ein beliebiger Knoten. Dann bezeichnet man mit $g^+(v)$ den* Außengrad *des Knotens v, d.h die Anzahl der Kanten $e \in E$ mit Anfangsknoten v, und mit $g^-(v)$ den* Innengrad *des Knotens v, d.h die Anzahl der Kanten $e \in E$ mit Endknoten v. Beides zusammen, also $g(v) := g^+(v) + g^-(v)$, nennt man den* Grad *von v.*

Betrachten wir wieder den Graphen G aus Abbildung 1.3, so ist z.B.

$$g^+(1) = 2, \quad g^-(1) = 1 \quad \text{und} \quad g^+(4) = 1, \quad g^-(4) = 1.$$

Die Betrachtung von Knotengraden kann hilfreich sein um die Frage nach der Existenz bestimmter Graphen zu beantworten, wie zum Beispiel: Gibt es einen (gerichteten) Graphen mit 5 Knoten und den Knotengraden $1, 2, 3, 4, 5$? Es gilt folgendes Resultat:

Lemma 1.13 (Handshaking-Lemma) *Es sei $G = (V, E)$ ein gerichteter Graph. Dann gilt*

$$\sum_{v \in V} g^+(v) = \sum_{v \in V} g^-(v) = |E|,$$

wobei $|E|$ die Anzahl der Kanten in G bezeichnet. Insbesondere gilt also

$$\sum_{v \in V} g(v) = 2|E|,$$

d.h. die Summe aller Knotengrade ist gerade.

Beweis Nach Definition ist $g^+(v)$ die Anzahl von Kanten $e \in E$ mit Anfangsknoten v. Da jede Kante genau einen Anfangsknoten hat und an diesem genau einmal gezählt wird, folgt

$$\sum_{v \in V} g^+(v) = |E|.$$

Die Aussage über die Innengrade folgt analog und die Aussage über die Grade aus der Definition $g(v) = g^+(v) + g^-(v)$. □

Zur Namensgebung: Man nennt den zweiten Teil dieses Resultat, Handshaking-Lemma, weil man die Aussage wie folgt illustrieren kann: Wann immer sich eine Menge von Menschen die Hände schütteln, so ist unabhängig davon, wer wem die Hand gibt, die Gesamtzahl der geschüttelten Hände gerade. Dies liegt daran, dass an jedem Händeschütteln zwei Hände beteiligt sind.

Damit können wir die obige Frage beantworten: Es kann keinen Graphen mit 5 Knoten und den Knotengraden $1, 2, 3, 4, 5$ geben, denn die Summe der Knotengrade in diesem Graphen wäre 15, also ungerade. Es ist jedoch nicht einmal nötig die Summe der Knotengrade zu bestimmen um die Frage zu beantworten, man kann stattdessen auch die nachfolgende Folgerung aus Lemma 1.13 verwenden.

Korollar 1.14 *Es sei $G = (V, E)$ ein gerichteter Graph. Dann ist die Anzahl von Knoten v mit ungeradem Grad $g(v)$ gerade.*

Beweis Jeder Knoten $v \in V$ hat entweder geraden oder ungeraden Grad. Wir können also V aufteilen in die Mengen

$$V_1 := \{v \in V \mid g(v) \text{ ist ungerade}\} \quad \text{und} \quad V_2 := \{v \in V \mid g(v) \text{ ist gerade}\}.$$

Dann ist $V_1 \cup V_2 = V$ und $V_1 \cap V_2 = \emptyset$, d.h. V_1 und V_2 bilden eine Partition von V. Mit Lemma 1.13 folgt

$$\sum_{v \in V_1} \underbrace{g(v)}_{\text{ungerade}} = \underbrace{2|E|}_{\text{gerade}} - \underbrace{\sum_{v \in V_2} g(v)}_{\text{gerade}}.$$

Damit die Summe der ungeraden Zahlen $g(v)$, $v \in V_1$, gerade wird, muss die Anzahl der Elemente von V_1 gerade sein. □

Wenden wir dieses Resultat an, so sieht man die Antwort auf die obige Frage noch schneller: Ein Graph mit 5 Knoten und den Knotengraden $1, 2, 3, 4, 5$ hätte 3 Knoten mit ungeradem Grad, also eine ungerade Anzahl von Knoten mit ungeradem Grad.

Wie man an unserem Beispielgraphen G sieht, ist die Darstellung eines Graphen als lange Listen von Knoten V und Kanten E nicht sehr übersichtlich, besonders bei größeren Graphen. Etwas kompakter lassen sich Graphen mit Hilfe von Matrizen darstellen.

Definition 1.15 *Es sei $G = (V, E)$ ein gerichteter Graph mit Knotenmenge $V = \{v_1, \ldots, v_n\}$ und Kantenmenge $E = \{e_1, \ldots, e_m\}$.*

- *Die* Adjazenzmatrix *des Graphen G ist die $n \times n$ Matrix A mit den Einträgen*

$$a_{j,k} = \text{ Anzahl der Kanten mit Anfangspunkt } v_j \text{ und Endpunkt } v_k.$$

- *Die* Inzidenzmatrix *des Graphen G ist die $n \times m$ Matrix I mit den Einträgen*

$$i_{j,k} = \begin{cases} 1 & \text{falls } v_j \text{ der Anfangspunkt der Kante } e_k \text{ ist,} \\ -1 & \text{falls } v_j \text{ der Endpunkt der Kante } e_k \text{ ist,} \\ 0 & \text{sonst.} \end{cases}$$

Betrachten wir wieder unseren Beispielgraphen G aus Abbildung 1.3, so ist

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{und} \quad I = \begin{pmatrix} 1 & -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Mit Hilfe dieser Matrizen kann man zum Beispiel sehr schnell Grade von Knoten bestimmen. So ist der Außengrad eines Knotens gleich der Anzahl der 1 in der zugehörigen Zeile der Adjazenzmatrix und der Innengrad gleich der Anzahl der 1 in der zugehörigen Spalte der Adjazenzmatrix. In der Inzidenzmatrix ist der Außengrad eines Knotens gleich der Anzahl der 1 in der zugehörigen Zeile und der Innengrad gleich der Anzahl der -1 in der zugehörigen Zeile.

1.3 Ungerichtete Graphen

In manchen Problemstellungen ist nur die Existenz von Kanten zwischen Knoten wichtig, nicht jedoch ihre Richtung. Bei U-Bahn-Netzen zum Beispiel fährt die U-Bahn im allgemeinen in beide Richtungen, relevant ist daher nur, ob eine Verbindung zwischen zwei Haltestellen existiert oder nicht. In diesen Fällen betrachtet man sogenannte ungerichtete Graphen.

Definition 1.16 *Ein* Graph *ist ein Paar $G = (V, E)$ mit einer endlichen Menge $V \neq \emptyset$ und einer endlichen Menge $E \subseteq \{\{u, v\} \mid u, v \in V, v \neq w\}$. Die Elemente von V heißen* Knoten *von G. Die Elemente von E heißen* Kanten *von G. Eine Kante $e \in E$ ist also von der Form $e = \{u, v\}$ mit gewissen $u, v \in V$, welche als* Endknoten *von e bezeichnet werden.*

Zwei Kanten $e_1, e_2 \in E$ mit $e_1 = \{u, v\}$ und $e_2 = \{u, v\}$ heißen parallel. *Enthält G keine parallelen Kanten, so heißt G* einfach.

Zur Illustration betrachten wir wieder den Beispielgraphen G aus Abbildung 1.3, aber diesmal als ungerichteten Graphen $G' = (V, E')$ mit den Knoten $V = \{1, 2, 3, 4, 5, 6\}$ und den Kanten $E' = \{\{1, 2\}, \{2, 1\}, \{1, 3\}, \{2, 3\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$, vergleiche Abbildung 1.4.

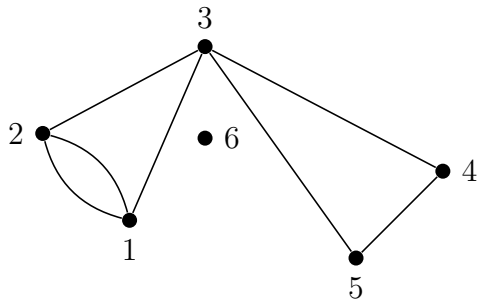


Abbildung 1.4: Ein ungerichteter Beispielgraph G'

In einem ungerichteten Graphen besitzen die Kanten keine Richtung oder Orientierung. Daher beschreibt man eine Kante e mit Hilfe der Menge ihrer beiden Endknoten u, v, wobei die Reihenfolge der Endknoten egal ist. Man kann also für eine Kante zwischen diesen beiden Endknoten $e = \{u, v\}$ oder $e = \{v, u\}$ schreiben.

Für ungerichtete Graphen kann man analog zu gerichteten Graphen definieren, wann ein Knoten und eine Kante inzidieren oder zwei Knoten/zwei Kanten benachbart sind. Einen Außen- und Innengrad gibt es hier natürlich nicht mehr, aber der Grad ist ebenso wie bei gerichteten Graphen definiert und die zugehörigen Ergebnisse aus Lemma 1.13 und Korollar 1.14 gelten auch für ungerichtete Graphen.

Außerdem kann man jedem gerichteten Graphen einen ungerichteten Graphen zuordnen, so wie wir das mit unserem Beispielgraphen G gemacht haben, vergleiche Abbildung 1.3 und 1.4.

Definition 1.17 *Es sei $G = (V, E)$ ein gerichteter Graph. Dann heißt $G' = (V, E')$ der G* zugeordnete ungerichtete Graph*, wenn gilt*

$$E' = \{\{u, v\} \mid (u, v) \in E\}.$$

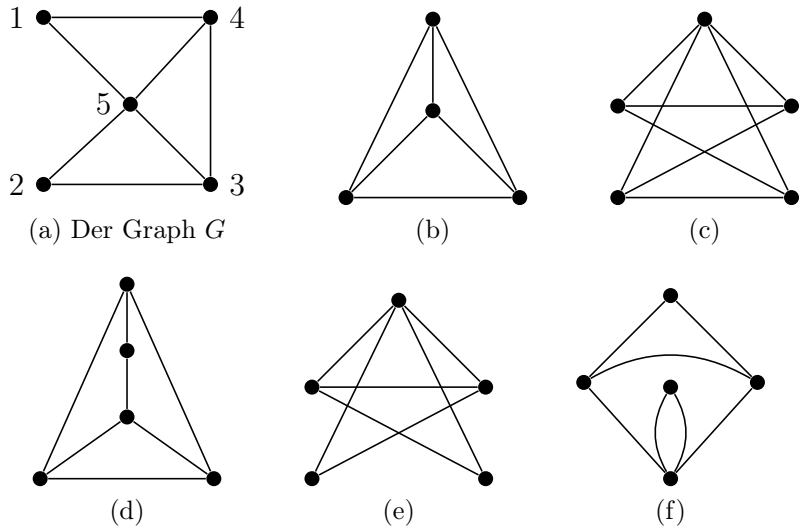
Umgekehrt kann man auch jedem ungerichteten Graphen $G = (V, E)$ einen gerichteten Graphen $G' = (V, E')$ zuordnen, indem man die Kanten *orientiert*. Da man eine ungerichtete Kante immer in zwei Richtungen orientieren kann, ist eine *Orientierung* G' nicht eindeutig bestimmt. Im Gegensatz dazu ist der zugeordnete ungerichtete Graph eindeutig.

1.4 Aufgaben

Aufgabe 1.1 Welche der Graphen in Abbildung 1.5 (b) – (f) stellen den Graphen G aus Abbildung 1.5 (a) dar? Begründen Sie Ihre Antworten.

Aufgabe 1.2 Beschreiben Sie das Haus vom Nikolaus mit einem ungerichteten Graphen $G = (V, E)$. Sind V und E eindeutig bestimmt?

Aufgabe 1.3



(a) Der Graph G (b) (c)

(d) (e) (f)

Abbildung 1.5: Wo versteckt sich der Graph G aus Aufgabe 1?

(a) Überlegen Sie sich, wie man für ungerichtete Graphen eine Adjazenzmatrix und eine Inzidenzmatrix definieren kann. Wie kann man in diesen den Grad eines Knotens ablesen?

(b) Bestimmen Sie für das Haus vom Nikolaus (vergleiche Aufgabe 2) die Inzidenzmatrix und die Adjazenzmatrix.

Aufgabe 1.4 Es sei $G = (V, E)$ ein ungerichteter einfacher Graph mit $|V| \geq 2$, d.h. mit mindestens zwei Knoten. Zeigen Sie, dass es dann mindestens zwei Knoten $u \neq v$ gibt mit $g(u) = g(v)$.

Aufgabe 1.5 An den Schülerprojekttagen nehmen 42 SchülerInnen aus Unterfranken teil. Diese sollen auf 6 Projektgruppen mit jeweils 7 Teilnehmern aufgeteilt werden, so dass sie genau 3 Mitglieder ihrer Gruppe schon (von vorherigen Veranstaltungen, aus der Schule, ...) kennen und 3 noch nicht. Wir wollen davon ausgehen, dass „sich kennen“ symmetrisch ist. Ist eine solche Aufteilung möglich?

2 Suchstrategien

Auch wenn wir sie immer wieder so darstellen werden, sollten wir uns nicht zu sehr daran gewöhnen, uns Graphen als Bilder bestehend aus Punkten und Linien vorzustellen. In vielen Anwendungen ist zwar genau bekannt, was die Knoten eines Graphen sind und welche Nachbarn ein Knoten hat, es gibt jedoch kein Bild des gesamten Graphen. Man besitzt also nur lokale Informationen, keine globalen. Betrachte hierzu die folgenden Beispiele.

Beispiel 2.1 (Rubik's Würfel) Erinnern wir uns wieder an unseren Rubik's Würfel, den wir auseinander und beliebig wieder zusammengebaut haben. Wenn wir wissen wollen, ob man ihn jetzt noch lösen kann und wieviele Drehungen man dafür mindestens braucht, dann können wir den folgenden Graphen betrachten: Jeder möglichen Konfiguration der Farben auf dem Würfel ordnen wir einen Knoten zu. Dann fügen wir zwischen zwei Knoten eine Kante ein, wenn sich der eine Zustand aus dem anderen durch eine beliebige Drehung einer Ebene erzeugen lässt. Finden wir nun eine Folge von Kanten von dem gelösten Würfel zu unserem neu zusammengebauten, so ist der Würfel noch lösbar.

Bedenkt man allerdings, dass der Würfel durch die Drehung einer Ebene in 27 verschiedene Zustände versetzet werden kann, so kann man schon ahnen, dass diese Frage nicht einfach zu beantworten sein wird. Tatsächlich lässt sich ein Rubik's Würfel auf ungefähr $519 \cdot 10^{18}$ verschiedene Arten zusammenbauen. Der entstehende Graph ist also riesig und die Suche nach einem Lösungsweg entsprechend schwierig.

Die erste Frage kann man allerdings schon mit Hilfe der Anzahl der Knoten beantworten: Nicht jeder beliebig zusammengebaute Würfel ist lösbar, denn ein korrekt zusammengebauter Rubik's Würfel lässt sich nur in ca. $43 \cdot 10^{18}$ verschiedene Konfigurationen drehen. Genauer gibt es in dem riesigen Graphen 12 Teilmengen von Knoten, innerhalb derer man sich durch Drehungen bewegen kann. Es ist jedoch nicht möglich von einer dieser Teilmengen in eine andere zu wechseln ohne den Würfel erneut auseinander zu bauen.

Auch die andere Frage ist inzwischen, zumindest in gewissem Sinne beantwortet: Im Jahr 2010 wurde unter massivem Computereinsatz gezeigt, dass jede lösbare Farbkonfiguration mit höchstens 20 Drehungen lösbar ist. Seit 1995 ist eine Farbkonfiguration, der sogenannte Superflip, bekannt, für die man auch mindestens 20 Drehungen braucht. ♢

Beispiel 2.2 (Suche nach Mr. X) Stellen Sie sich vor, jemand gibt Ihnen einen Brief, der für einen zufällig ausgewählten Menschen auf der Erde ist, nennen wir ihn Mr. X. Dieser Brief soll Mr. X erreichen, darf aber immer nur zwischen Menschen weitergegeben werden, die sich kennen. Wie finden Sie einen Weg Mr. X den Brief zukommen zu lassen?

Dieses Problem lässt sich mit Hilfe von Graphen wie folgt beschreiben: Man ordnet jedem Menschen auf der Erde einen Knoten zu und fügt zwischen zwei Knoten genau dann

eine Kante ein, wenn die zugehörigen Menschen sich kennen. Dann erhält man einen sehr großen ungerichteten Graphen mit ca. 7 Milliarden Knoten und muss einen Weg von dem eigenen Knoten zu dem von Mr. X finden. ◊

Auch wenn es möglich ist, aus diesen Informationen eine visuelle Darstellung des Graphen zu erzeugen, so ist dies nicht immer sinnvoll. Eine Bild eines Graphen kommt dem Menschen entgegen, der dieses Bild in seiner Gesamtheit wahrnehmen und nützliche Strukturen erkennen kann. So kann ein Mensch zum Beispiel den kürzesten Weg zwischen zwei Knoten eines Graphens einfach "sehen". Im Gegensatz dazu kennt ein Computer immer nur die Inzidenz- oder Adjazenzmatrix eines Graphen, nicht aber seine gesamte Struktur. Das führt dazu, dass er zwar von jedem Knoten die direkten Nachbarn weiß, aber keine größeren Zusammenhänge erkennt.

Eine Aufgabe wie zum Beispiel Wege zwischen Knoten zu finden ist für einen Computer daher deutlich schwerer als für einen Menschen. Warum also sollte man sich die Mühe machen einen Algorithmus zu finden, der es dem Computer ermöglicht dieses Problem zu lösen? Der Vorteil des Menschen beruht darin, dass er einen ganzen Graphen auf einmal wahrnehmen und verarbeiten kann. In vielen Anwendungen sind die Graphen jedoch sehr groß und kompliziert, man denke zum Beispiel an das europäische Eisenbahn-, Strom- oder Gasnetz oder die beiden obigen Beispiele. In diesen Fällen kann auch ein Mensch den Graph nicht mehr als ganzes verarbeiten und muss sich Lösungsstrategien überlegen. Im Gegensatz dazu ist es für den Computer, abgesehen von Laufzeit und Speicherplatzbedarf, egal wie groß der Graph ist. Hat man einen Algorithmus gefunden, der es dem Computer ermöglicht ein bestimmtes Problem zu lösen, so kann dieses Verfahren im Prinzip auf Graphen beliebiger Größe angewendet werden.

2.1 Wege und Zusammenhang

Bevor wir uns daran machen, in einem Graphen Wege zwischen Knoten zu finden und damit das Problem aus Beispiel 2.2 zu lösen, sollten wir uns erst überlegen, was wir mit einem Weg genau meinen.

Definition 2.3 *Es sei $G = (V, E)$ ein gerichteter bzw. ein ungerichteter Graph. Ein* (gerichteter) Weg *(auf englisch: path) in G ist eine endliche Folge*

$$P = (v_0, e_1, v_1, e_2, v_2, \ldots, e_l, v_l),$$

wobei $v_i \in V$ $(i = 0, \ldots, l)$ Knoten und $e_i \in E$ $(i = 1, \ldots, l)$ Kanten mit Anfangsknoten v_{i-1} und Endknoten v_i bzw. Endknoten v_{i-1} und v_i sind. Man nennt v_0 den Startknoten*, v_l den* Endknoten *und l die* Länge *des Weges. Ist $v_0 = v_l$, so nennt man P einen* Kreis*.*

In gerichteten Graphen definiert man zusätzlich einen ungerichteten Weg *als eine endliche Folge*

$$P = (v_0, e_1, v_1, e_2, v_2, \ldots, e_l, v_l),$$

wobei $v_i \in V$ $(i = 0, \ldots, l)$ Knoten und $e_i \in E$ $(i = 1, \ldots, l)$ Kanten mit Anfangsknoten v_{i-1} und Endknoten v_i oder umgekehrt sind.

Ein Weg P heißt einfach[1]*, wenn er keine Kante und (abgesehen von eventuell $v_1 = v_l$) keinen Knoten mehrfach durchläuft.*

Ein ungerichteter Graph G heißt kreisfrei*, wenn er keine einfachen Kreise enthält; ein gerichteter Graph G heißt* kreisfrei*, wenn der zugeordnete ungerichtete Graph kreisfrei ist.*

Die Bedingung, dass einfache Wege nicht nur keine Knoten mehrfach durchlaufen, sondern auch keine Kanten, benötigt man, um in ungerichteten Graphen auszuschließen, dass Wege wie $P = (v_1, \{v_1, v_2\}, v_2, \{v_1, v_2\}, v_1)$ einfach sind. Dieser Weg ist ein Kreis und durchläuft abgesehen von dem Start-/Endknoten keinen Knoten mehrfach.

Zur Illustration betrachten wir wieder den Beispielgraphen G aus Abbildung 1.3, vergleiche Abbildung 2.1. Ein Weg ist hier zum Beispiel

$$P_1 = (2, (2,3), 3, (3,4), 4).$$

Dieser Weg hat den Startknoten 2 und den Endknoten 4 und ist einfach, da er keinen Kanten und keine Knoten mehrfach durchläuft. Der Weg

$$P_2 = (1, (1,2), 2, (2,1), 1)$$

ist ein einfacher Kreis, daher ist G nicht kreisfrei. Der Weg

$$P_3 = (3, (3,5), 5, (4,5), 4)$$

ist ein Beispiel für einen einfachen ungerichteten Weg.

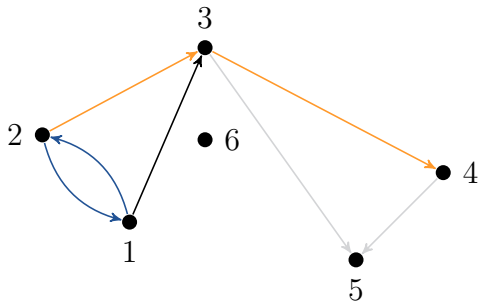


Abbildung 2.1: Wege im Beispielgraphen G aus Abbildung 1.3

Nachdem wir nun definiert haben, was wir unter einem Weg verstehen wollen, drängt sich die folgende Frage auf: Wir betrachten einen beliebigen (gerichteten oder ungerichteten) Graphen $G = (V, E)$ und zwei beliebige Knoten $s, t \in V$. Können wir dann in G immer einen Weg mit Startknoten s und Zielknoten t finden? Ein kurzer Blick auf den Beispielgraphen G aus Abbildung 2.1 verrät die Anwort: Der Knoten 6 hat Grad 0, es gibt also

[1] Manche Bücher unterscheiden zwischen Wegen, die nur keine Kanten mehrfach durchlaufen, und solchen, die zusätzlich keine Knoten mehrfach durchlaufen. Macht man diese Unterscheidung, so nennt man die ersten *einfach* und die zweiten *elementar*.

keinen Weg mit Start- oder Zielknoten 6. In gerichteten Graphen kann es jedoch auch vorkommen, dass es selbst zwischen Knoten, zwischen denen Kanten existieren, keinen (gerichteten) Weg gibt. Betrachtet man zum Beispiel die Knoten 3 und 5, so gibt es mehrere Wege mit Startknoten 3 und Endknoten 5, aber keinen einzigen Weg mit Startknoten 5 und Endknoten 3. Diese Überlegung gibt Anlass zur folgenden Definition.

Definition 2.4 *Es sei $G = (V, E)$ ein ungerichteter Graph. Dann heißt G* zusammenhängend*, wenn zwischen je zwei Knoten $s, t \in V$ ein Weg P in G existiert.*

Ist $G = (V, E)$ ein gerichteter Graph, so heißt G stark zusammenhängend*, wenn für je zwei Knoten $s, t \in V$ ein (gerichteter) Weg P mit Anfangsknoten s und Endknoten t existiert, und* schwach zusammenhängend*, wenn für je zwei Knoten $s, t \in V$ ein ungerichteter Weg P mit Anfangsknoten s und Endknoten t existiert.*

Ein Knoten $s \in V$ heißt Wurzel *von G, wenn für alle Knoten $t \in V$ ein gerichteter Weg P mit Anfangsknoten s und Endknoten t existiert.*

Ein gerichteter Graph, der eine Wurzel besitzt, ist also immer schwach zusammenhängend, aber nicht jeder schwach zusammenhängende Graph besitzt eine Wurzel. Ein gerichteter Graph ist stark zusammenhängend genau dann, wenn jeder Knoten eine Wurzel ist.

Der gerichtete Beispielgraph G aus Abbildung 2.1 ist also weder stark noch schwach zusammenhängend. Würde der Knoten 6 nicht zum Graphen gehören, so wären die Knoten 1 und 2 Wurzeln.

2.2 Breitensuche

Kommen wir nun wieder auf das Problem aus Beispiel 2.2 zurück. Angenommen, der zugehörige Graph wäre zusammenhängend, wie finden Sie einen Weg von dem eigenen Knoten zu dem von Mr. X? Und das nur auf Basis von Nachbarschaftsbeziehungen, d.h. Sie wissen zunächst nur, wen Sie selbst kennen und auch jeder andere Mensch weiß nur, wen er kennt.

Eine Lösungsmöglichkeit ist die folgende: Sie überlegen sich zunächst, wen Sie alles kennen. Ist Mr. X darunter, so haben Sie das Problem gelöst. Kennen Sie Mr. X nicht, so fragen Sie alle Menschen, die Sie kennen, ob diese Mr. X kennen. Kennt einer Ihrer Bekannten Mr. X, so geben Sie ihm den Brief und er gibt ihn Mr. X. Kennen Ihre Bekannten Mr. X auch nicht, so bitten Sie sie, ihre Bekannten zu fragen, ob die ihn kennen und so weiter.

Wir illustrieren dies am Beispiel des Graphen $G = (V, E)$ in Abbildung 2.2 und versuchen ausgehend von Knoten 1 einen Weg zu Knoten 10 in V zu finden. Dann würde das oben beschriebene Verfahren wie folgt aussehen: Für die Knoten $2, 7$ und 11 ist klar, wie ein Weg aussieht, denn es existiert eine direkte Verbindung. Ausgehend vom Knoten 2 lassen sich die Knoten 3 und 5 erreichen, ausgehend von Knoten 7 die Knoten 5 (den wir aber schon hatten), 8 und 12, sowie ausgehend von Knoten 11 der Knoten 8 (zu dem wir ebenfalls schon einen Weg gefunden haben). Betrachtet man nun die Knoten $3, 5, 8$ und 12 und deren noch nicht erreichte Nachbarn, so erreicht man die Knoten $4, 6, 9$ und 13 und noch eine Runde später schließlich den Knoten 10. Auf diese Weise findet man einen Weg vom

Knoten 1 zu Knoten 10, tatsächlich zu allen anderen Knoten im Graphen, und verwendet dabei ausschließlich das Wissen über die Nachbarn eines Knotens.

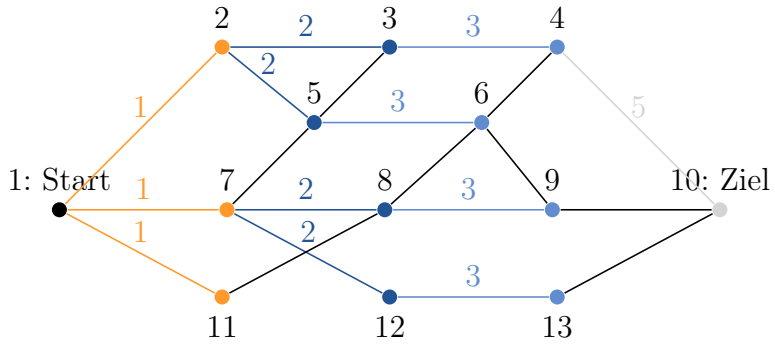


Abbildung 2.2: Breitensuche ausgehend vom Knoten 1

Will man vom Startknoten aus einen Weg zu allen anderen Knoten im Graphen bestimmen, so ergibt dieses Verfahren gerade die sogenannte *Breitensuche*, die in Algorithmus 1 formalisiert wird. Sucht man nur einen Weg vom Startknoten zu einem bestimmten Zielknoten, so kann man das gleiche Verfahren anwenden und abbrechen, sobald der gewünschte Knoten gefunden wurde.

Algorithmus 1 Breitensuche

Input: Ein zusammenhängender ungerichteter Graph $G = (V, E)$ und ein Startknoten s oder ein gerichteter Graph $G = (V, E)$ mit einer Wurzel s.

Setze die Warteschlange L auf $L = (s)$, die Abstände auf $d(s) = 0$, $d(v) = \infty$ für alle $v \in V \setminus \{s\}$ und die Vorgänger auf $\pi(v) = 0$ für alle $v \in V$.
while $L \neq \emptyset$ **do**
  Entferne das erste Element v aus der Warteschlange L
  for all $u \in V$ mit $\{v, u\} \in E$ bzw. $(v, u) \in E$ **do**
    if $d(u) = \infty$ **then**
      Setze $d(u) = d(v) + 1$ und $\pi(u) = v$.
      Füge u an das Ende der Warteschlange L an.
    end if
  end for
end while

Output: Die Vorgänger $\pi(v)$ und die Abstände $d(v)$ für alle $v \in V$.

Der Algorithmus 1 gibt zwei Vektoren zurück: Die Abstände $d(v)$ geben für jeden Knoten $v \in V$ an, wie groß sein Abstand zum Startknoten s ist, d.h. wie viele Kanten der gefundene Weg zwischen den beiden Knoten s und v hat. Wie wir im folgenden Resultat sehen werden, ist Algorithmus 1 nicht nur wohldefiniert und liefert die gewünschten Ergebnisse. Er findet zusätzlich auch noch den *kürzesten Weg* vom Startknoten s zu allen anderen Knoten, d.h.

den Weg mit den wenigsten Kanten. Die *Anzahl der Kanten* in einem solchen kürzesten Weg von s nach v bezeichnet man mit $\text{dist}(s, v)$.

Der Vorgänger $\pi(v)$ enthält für jeden Knoten $v \neq s$ den vorletzten Knoten auf dem Weg von s nach v, also den Vorgänger von v auf diesem Weg. Kennt man den Vorgänger $\pi(v)$, so kann man dessen Vorgänger auslesen und so den Weg zum Startknoten s zurückverfolgen. So erhält man den sogenannten *Vorgängergraphen* $G_\pi := (V_\pi, E_\pi)$ mit

$$\begin{aligned} V_\pi &:= \{v \in V \mid \pi(v) \neq 0\} \cup \{s\}, \\ E_\pi &:= \{(\pi(v), v) \mid v \in V_\pi \setminus \{s\}\}. \end{aligned}$$

Satz 2.5 *Es sei $G = (V, E)$ ein zusammenhängender ungerichteter Graph mit einem Startknoten s oder ein gerichteter Graph mit einer Wurzel s. Dann ist Algorithmus 1 wohldefiniert und Zahlen $d(v)$ geben für alle $v \in V$ die kürzeste Distanz vom Knoten v zum Startknoten s an.*

Beweis Um zu zeigen, dass der Algorithmus wohldefiniert ist, müssen wir zeigen, dass die Warteschlange irgendwann leer ist und deshalb das Verfahren abbricht. In die Warteschlange werden nur Knoten v mit $d(v) = \infty$ aufgenommen, d.h. Knoten, die vorher noch nicht erreicht wurden. In dem Moment, in dem ein Knoten v in die Warteschlange aufgenommen wird, wird jedoch das zugehörige $d(v)$ auf einen endlichen Wert gesetzt. Folglich kann jeder Knoten nur einmal in die Warteschlange aufgenommen werden und die Berechnung neuer Abstände in Zeile 6 ist wohldefiniert. Da in jeder Iteration der erste Knoten aus der Warteschlange entfernt wird, bricht der Algorithmus nach spätestens $|V|$ Iterationen ab.

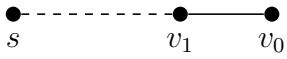


Abbildung 2.3: Illustration des Weges P im Beweis von Satz 2.5

Es bleibt zu zeigen, dass für alle Knoten $v \in V$ gilt $d(v) = \text{dist}(s, v)$. Nach Voraussetzung existiert für alle $v \in V$ ein Weg von s nach v, d.h. $\text{dist}(s, v)$ ist wohldefiniert und endlich. Da es nach Konstruktion für alle $v \in V$ einen Weg von s nach v mit $d(v)$ Kanten gibt (oder $d(v) = \infty$ gilt), gilt immer $\text{dist}(s, v) \leq d(v)$. Angenommen, es gilt nicht für alle $v \in V$ der Gleichheit. Dann wählen wir einen Knoten $v_0 \in V$ mit

$$\text{dist}(s, v_0) = \min\{\text{dist}(s, v) \mid \text{dist}(s, v) < d(v)\},$$

d.h. einen Knoten mit falschem $d(v)$, der möglichst dicht am Startknoten s liegt. Sei nun P ein kürzester Weg von s nach v_0 und v_1 der Vorgänger von v_0 auf diesem Weg. Dann gelten auf Grund der Wahl von v_0

- $d(v_1) = \text{dist}(s, v_1)$, denn $\text{dist}(s, v_1) = \text{dist}(s, v_0) - 1 < \text{dist}(s, v_0)$,
- $\text{dist}(s, v_0) = \text{dist}(s, v_1) + 1$ und
- $d(v_0) \leq d(v_1) + 1$, denn v_0 ist von v_1 aus erreichbar.

Insgesamt folgt also

$$d(v_0) \leq d(v_1) + 1 = \text{dist}(s, v_1) + 1 = \text{dist}(s, v_0),$$

ein Widerspruch zur Wahl von v_0. Folglich gilt für alle $v \in V$ wie behauptet $d(v) = \text{dist}(s, v)$. □

2.3 Tiefensuche

Wie der Name Breitensuche schon impliziert, wird bei dieser Suchstrategie zuerst in die Breite und dann, wenn nötig, in die Tiefe gesucht. In manchen Anwendungen ist aber ein anderes Suchschema intuitiver. Betrachte hierzu folgendes Beispiel:

Beispiel 2.6 (Labyrinth) Stellen Sie sich vor, Sie machen einen Ausflug und besuchen einen Irrgarten. Betrachtet man die Wege als Kanten und jede Kreuzung oder Sackgasse als Knoten, so kann man diesen Irrgarten abstrakt als Graphen darstellen, vergleiche Abbildung 2.4. Wie findet man nun den Weg vom Eingang in den Irrgarten zum Ausgang? ♢

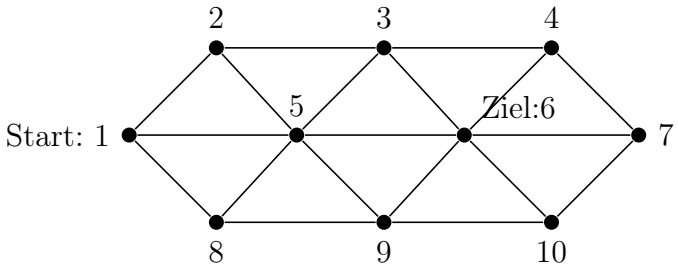


Abbildung 2.4: Ein Beispiel eines Irrgartens

In dem Beispiel aus Abbildung 2.4 befindet sich der Eingang des Labyrinths bei Knoten 1 und der Ausgang bei Knoten 6. Um einen Weg vom Start zum Ziel zu finden, könnte man theoretisch die Breitensuche aus dem letzten Abschnitt verwenden. Wie man sich leicht überlegt, würde dies aber dazu führen, dass man viel hin- und herlaufen müsste, denn man würde von Knoten 1 zu Knoten 2, zurück zu Knoten 1 und dann zu Knoten 5, wieder zurück zu Knoten 1 und nun zu Knoten 8, ... laufen.

Andererseits ist ein viel einfacherer Weg bekannt, wie man in einem Labyrinth sein Ziel findet, die sogenannte „rechte Hand"–Regel. Man legt bei Betreten des Labyrinths die rechte (natürlich geht auch die linke) Hand auf die Wand und folgt dann den Wegen ohne die Hand von der Wand zu entfernen. Gerät man in eine Sackgasse oder eine Kreuzung, an der man schon war, so führt dieses Verhalten dazu, dass man den gleichen Weg, den man gekommen ist, wieder zurück geht, bis man eine neue Abzweigung gefunden hat. Im obigen Beispiel 2.6 würde das zu dem in Abbildung 2.5 dargestellten Weg führen. Hierbei sind die

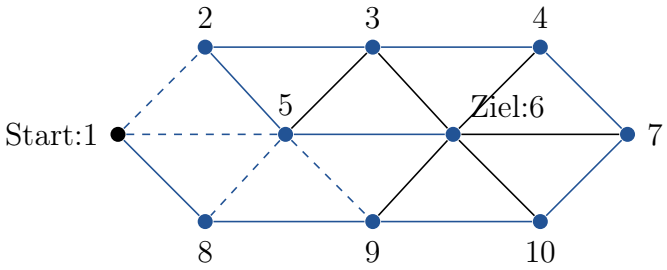


Abbildung 2.5: Tiefensuche mit Startknoten 1 und Zielknoten 6

Kanten gestrichelt dargestellt, die hin- und wieder zurück gelaufen werden, da sie zu einem schon bekannten Knoten führen.

Formalisiert man diesen Ansatz, so erhält man die sogenannte *Tiefensuche*, vergleiche Algorithmus 2. In diesem Algorithmus werden wieder, ausgehend von einem Startknoten, Wege zu allen anderen Knoten im Graphen gesucht. Sucht man nur einen Weg zu einem bestimmten Knoten, so kann man den Algorithmus abbrechen, sobald der Zielknoten gefunden wurde.

Algorithmus 2 Tiefensuche: Hauptprogramm

Input: Ein zusammenhängender ungerichteter Graph $G = (V, E)$ und ein Startknoten s oder ein gerichteter Graph $G = (V, E)$ mit einer Wurzel s.

1 Setze die Farbe auf $c(v) = 0$ und den Vorgänger auf $\pi(v) = 0$ für alle $v \in V$.
2 Führe das Unterprogramm visit(s) aus.

Output: Die Vorgänger $\pi(v)$ für alle $v \in V$.

Algorithmus 3 Tiefensuche: Unterprogramm visit(v)

```
1 Setze c(v) = 1.
2 for all u ∈ V mit (v,u) ∈ E bzw. {v,u} ∈ E do
3   if c(u) = 0 then
4     Setze π(u) = v.
5     Führe das Unterprogramm visit(u) aus.
6   end if
7 end for
```

Algorithmus 2 gibt die schon von der Breitensuche bekannten Vorgänger $\pi(v)$ zurück, mit dessen Hilfe man einen Weg vom Startknoten zu jedem anderen Knoten im Graphen rekonstruieren kann. Intern verwendet der Algorithmus außerdem die Farben $c(v)$, mit dessen Hilfe gespeichert wird, ob ein Knoten schon besucht wurde. Hierbei bedeutet $c(v) = 0$, dass der Knoten v noch weiß, also unbesucht ist. Hat man den Knoten v einmal aufgesucht, so färbt man ihn schwarz, d.h man setzt $c(v) = 1$. Der rekursive Aufruf des Unterprogramms `visit(`v`)` bildet das anfangs besprochene Verhalten nach: Man geht solange weiter zu

unbesuchten Knoten, bis man in einer Sackgasse, d.h. einem Knoten, von dem man alle Nachbarn schon besucht hat, landet. Dann zieht man sich zurück auf den letzten Knoten davor, der noch unbesuchte Nachbarn hat, und geht von da aus wieder so weit wie möglich.

Wie wir im folgenden Resultat sehen werden, ist die Tiefensuche wohldefiniert und findet vom Startknoten Wege zu allen anderen Knoten im Graph. Im Gegensatz zur Breitensuche sind dies jedoch nicht immer die kürzesten Wege, d.h. die Wege mit den wenigsten Kanten, vergleiche Abbildung 2.5.

Satz 2.7 *Es sei $G = (V, E)$ ein zusammenhängender ungerichteter Graph mit einem Startknoten s oder ein gerichteter Graph mit einer Wurzel s. Dann ist Algorithmus 2 wohldefiniert und findet für alle Knoten in $v \in V$ einen Weg von s nach v.*

Beweis Wir müssen zeigen, dass der Algorithmus nach endlich vielen Aufrufen des Unterprogramms `visit(`v`)` abbricht und dass am Ende alle Knoten besucht wurden.

Das Unterprogramm `visit(`v`)` wird nur für Knoten v mit $c(v) = 0$ aufgerufen. Da in Zeile 1 des Unterprogramms $c(v) = 1$ gesetzt wird, kann `visit(`v`)` also für jeden Knoten $v \in V$ höchstens einmal aufgerufen werden. Daher bricht Algorithmus 2 nach endlich vielen Aufrufen des Unterprogramms `visit(`v`)` ab.

Abbildung 2.6: Illustration des Weges P im Beweis von Satz 2.7

Angenommen, es gäbe nach Abbruch des Algorithmus noch einen Knoten v_0, der nicht besucht wurde, d.h. $c(v_0) = 0$. Da G zusammenhängend ist, gibt es einen (gerichteten) Weg P von s nach v_0 und auf diesem Weg wurde mindestens s besucht. Folglich gibt es auf dem Weg (mindestens) ein Paar aufeinanderfolgender Knoten v_1 und v_2 mit $c(v_1) = 1$ und $c(v_2) = 0$, vergleiche Abbildung 2.6. Andererseits bedeutet $c(v_1) = 1$, dass das Unterprogramm `visit(`v_1`)` aufgerufen wurde. Wegen $(v_1, v_2) \in E$ bzw. $\{v_1, v_2\} \in E$ wurde dann aber auch `visit(`v_2`)` aufgerufen und darin $c(v_2) = 1$ gesetzt, ein Widerspruch. Daher gilt nach Abbruch des Algorithmus $c(v) = 1$ für alle $v \in V$, d.h. alle Knoten wurden besucht. □

2.4 Aufgaben

Aufgabe 2.1 Betrachten Sie den gerichteten Graphen in Abbildung 2.7.

(a) Bestimmen Sie mit Hilfe der Breitensuche ausgehend von Knoten 1 Wege zu allen anderen Knoten.

(b) Bestimmen Sie mit Hilfe der Tiefensuche ausgehend von Knoten 1 Wege zu allen anderen Knoten.

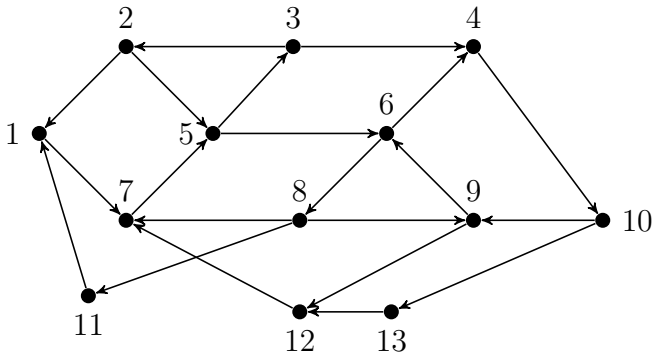


Abbildung 2.7: Ein gerichteter Beispielgraph G

Aufgabe 2.2 In Abbildung 2.8 (a) – (d) wird viermal der gleiche Graph dargestellt. In welcher der Abbildungen wurden die eingezeichneten Wege (= dicke Kanten) ausgehend vom Startknoten 1 mit der Breitensuche, der Tiefensuche oder mit keinem der beiden Algorithmen bestimmt? Begründen Sie Ihre Antworten und geben Sie gegebenenfalls die Reihenfolge an, in der die Knoten von der Suchstrategie gefunden wurden.

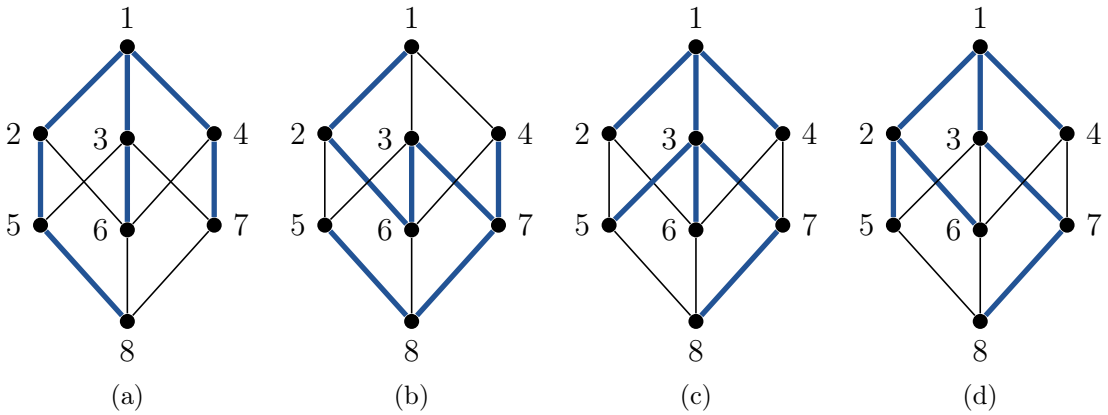


Abbildung 2.8: Welche Wege wurden mit Breiten- oder Tiefensuche erzeugt?

Aufgabe 2.3 Im Gegensatz zur Breitensuche bestimmt die Tiefensuche nicht notwendig einen kürzesten Weg von s zu allen Knoten.

(a) Gibt es Graphen, in denen auch die Tiefensuche einen kürzesten Weg von s zu allen Knoten bestimmt?

(b) Gibt es immer wenigstens einen Knoten $v \neq s$, zu dem die Tiefensuche einen kürzesten Weg bestimmt?

(c) Gibt es immer einen Knoten v, zu dem die Tiefensuche einen längsten einfachen Weg bestimmt?

Aufgabe 2.4 Betrachten Sie das Labyrinth aus Abbildung 2.9.

(a) Bestimmen Sie mit der „rechten Hand“-Regel einen Weg vom Eingang zur Mitte.

(b) Stellen Sie das Labyrinth als einen Graphen dar.

(c) Bestimmen Sie mit Hilfe der Tiefensuche einen Weg vom Eingang zur Mitte.

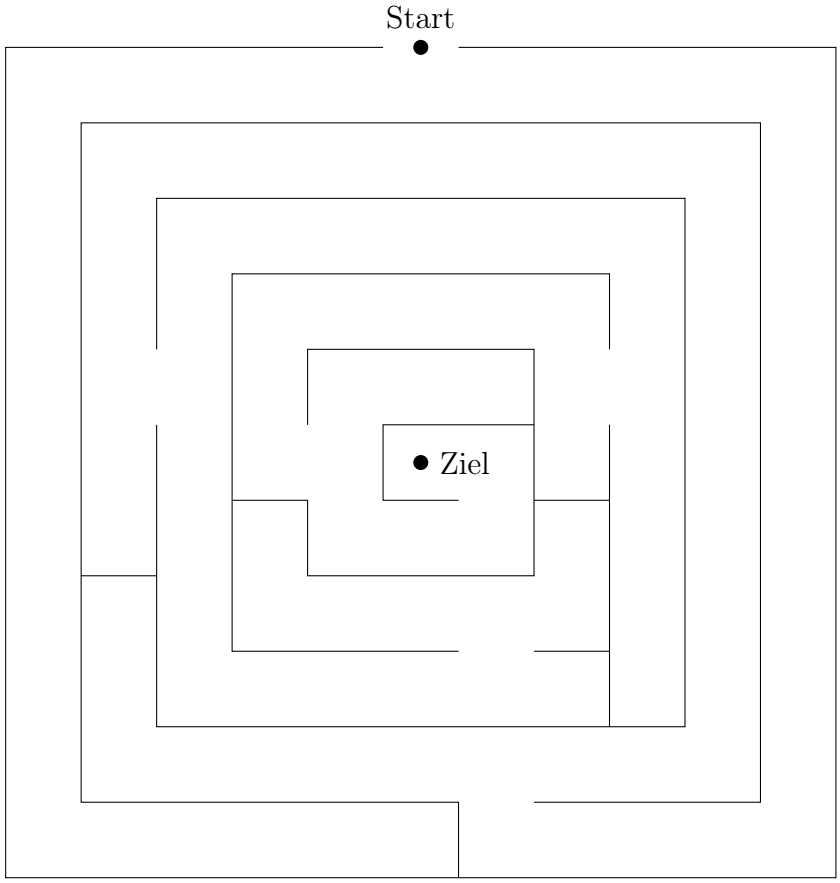


Abbildung 2.9: Ein Labyrinth

Aufgabe 2.5 Sie besitzen 2 Krüge: Krug 1 hat 3 Liter Fassungsvermögen, Krug 2 hat 5 Liter Fassungsvermögen. Sie können Krüge ganz füllen, komplett ausleeren oder Wasser von einem Krug in den anderen füllen, bis dieser voll oder der erste leer ist. Außerdem haben Sie hinreichend viel Wasser um die Krüge beliebig oft aufzufüllen.

(a) Welche Kombinationen von Füllständen sind, ausgehend von zwei leeren Krügen, möglich?

(b) Stellen Sie mit Hilfe eines Graphen dar, wie diese in einander übergehen können, wenn Sie eine der Aktionen ausführen, also einen Krug füllen, einen ausgießen oder von einem in den anderen umfüllen.

(c) Wie messen Sie, ausgehend von zwei leeren Krügen am schnellsten $1, 2$ und 4 Liter ab?

3 Bäume

Betrachtet man die von der Breitensuche und Tiefensuche erzeugten Teilgraphen (eine genaue Definition von Teilgraphen kommt gleich), vergleiche Abbildung 2.2 und 2.5, so stellt man fest, dass beide zwei gemeinsame Eigenschaften haben: Es gibt einen Knoten, von dem aus Wege zu allen anderen Knoten im Graphen existieren und es gibt keine (ungerichteten) Kreise. Man kann zeigen, dass dies nicht nur bei unseren Beispielen zufällig so ist, sondern immer der Fall ist.

Dies wollen wir im folgenden nicht tun, sondern uns mit speziellen Graphen, sogenannten Bäumen, befassen, die die beiden oben genannten Eigenschaften haben. Bäume spielen in diversen Anwendungen eine Rolle:

Beispiel 3.1 (Telefonkette) Kommen wir wieder auf das Beispiel mit der Telefonkette zurück, so können wir das Problem mit Hilfe eines gerichteten Graphen darstellen: Wir ordnen jedem Schüler einen Knoten zu und ziehen Kanten von jedem Schüler zu allen anderen, von denen er die Telefonnummer kennt. Wollen wir nun eine Telefonkette bilden, so müssen wir eine Teilmenge von Kanten auswählen, so dass ein gerichteter Weg von einem Schüler, der von der Lehrerin angerufen wird, zu allen anderen Schülern existiert. Da wir außerdem mit so wenig Anrufen wie möglich auskommen wollen, versuchen wir Kreise zu vermeiden. ◊

Beispiel 3.2 (Stammbäume) Wenn Sie einen Familienstammbaum betrachten, werden Sie feststellen, dass auch er diese beiden Eigenschaften hat: Es gibt einen Vorfahren, von dem die ganze Familie abstammt, mit dem also jeder verwandt ist. Wenn Sie nicht zu weit in die Vergangenheit zurückgehen, gibt es außerdem keine Kreise, da dies bedeuten würde, dass enge Verwandte untereinander geheiratet hätten. ◊

Beispiel 3.3 (Straßenbau) Betrachten wir nun eine kleine Südseeinsel, auf der es 5 Siedlungen gibt. Zur Förderung des Tourismus beschließen die Inselbewohner zwischen den Siedlungen Wege zu befestigen. Da die Natur soweit wie möglich unberührt bleiben soll, wollen sie so wenige Straßen wie möglich bauen. Wieviele Wege müssen Sie mindestens befestigen? ◊

Nachdem wir gerade schon mehrfach den Begriff Teilgraph erwähnt haben, wollen wir zunächst definieren, was damit gemeint ist.

Definition 3.4 *Es sei $G = (V, E)$ ein gerichteter oder ungerichteter Graph. Ein Graph $G' = (V', E')$ heißt* Teilgraph *von G, wenn $V' \subseteq V$ und $E' \subseteq E$ gilt. Gilt $V' = V$, so nennt man G' einen* erzeugenden/aufspannenden Teilgraphen *von G.*

Die von der Breiten- bzw. Tiefensuche erzeugten Teilgraphen sind also erzeugend.

Um Bäume zu definieren, müssen wir gerichtete und ungerichtete Graphen getrennt betrachten. Wir beginnen mit dem einfacheren Fall, mit ungerichteten Graphen.

3.1 Zusammenhang in ungerichteten Graphen

Wir haben zuvor schon definiert, wann wir einen Graphen zusammenhängend nennen. Im folgenden wollen wir uns noch etwas näher mit dem Zusammenhang von Teilgraphen eines, selber nicht notwendig zusammenhängenden, Graphen beschäftigen.

Definition 3.5 *Es sei $G = (V, E)$ ein ungerichteter Graph und $v \in V$. Dann nennt man die Menge*

$$Z(v) := \{u \in V \mid \text{es gibt in } G \text{ einen Weg von } v \text{ nach } u\}$$

die Zusammenhangskomponente *von v.*

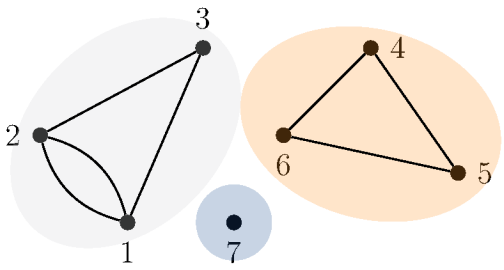


Abbildung 3.1: Zusammenhangskomponenten eines ungerichteten Graphen

Wie man an Abbildung 3.1 sehen kann, haben die Zusammenhangskomponenten eines Graphen die folgende Eigenschaft.

Lemma 3.6 *Es sei $G = (V, E)$ ein ungerichteter Graph und $v \in V$. Dann bilden die Zusammenhangskomponenten von G eine Partition von V, d.h. es gilt $\bigcup_{v \in V} Z(v) = V$ und für $v, u \in V$ gilt entweder $Z(v) = Z(u)$ oder $Z(v) \cap Z(u) = \emptyset$.*

Beweis Für alle $v \in V$ gilt $v \in Z(v)$, denn v ist mit sich durch den trivialen Weg $P = (v)$ verbunden. Daraus folgt sofort $\bigcup_{v \in V} Z(v) = V$. Seien nun $u^*, v^* \in V$ beliebig. Ist $Z(u^*) \cap Z(v^*) = \emptyset$, so ist nichts zu zeigen. Angenommen, es gibt ein $u \in Z(u^*) \cap Z(v^*)$. Dann gibt es in G einen Weg P_1 von u^* nach u, einen Weg P_2 von v^* nach u und für alle $v \in Z(v^*)$ in G einen Weg P_3 von v^* nach v. Schaltet man P_1, P_2 und P_3 in geeigneter Richtung hintereinander, so erhält man einen Weg von u^* (über u und v^*) nach v. Da $v \in Z(v^*)$ beliebig war, folgt $Z(v^*) \subseteq Z(u^*)$. Mit den gleichen Argumenten erhält man auch die umgekehrte Inklusion und damit die gewünschte Gleichheit. □

Dank dieses Lemmas kann man einen ungerichteten Graphen $G = (V, E)$ immer in zusammenhängende Teilgraphen $G_1, \ldots, G_p$ mit $G_i = (V_i, E_i)$ aufteilen, wobei $V_1, \ldots, V_p$ die

durch die Zusammenhangskomponenten definierte Partition von V ist und die Kantenmengen definiert sind als

$$E_i := \{e = \{v, u\} \in E \mid v, u \in V_i\}.$$

Dann gilt nicht nur $\bigcup_{i=1}^{p} V_i = V$, sondern auch $\bigcup_{i=1}^{p} E_i = E$, denn zwischen zwei verschiedenen Zusammenhangskomponenten V_i, V_j kann es offensichtlich keine Kanten geben. Daher werden die Teilgraphen $G_1, \ldots, G_p$ als die *Zusammenhangskomponenten* von G bezeichnet.

Ein ungerichterer Graph ist also genau dann zusammenhängend, wenn er nur eine Zusammenhangskomponente besitzt. Aus dem Zusammenhang eines ungerichteten Graphen kann man Aussagen über die Anzahl seiner Kanten ableiten.

Lemma 3.7 *Es sei $G = (V, E)$ ein ungerichteter zusammenhängender Graph. Dann gilt $|E| \geq |V| - 1$.*

Beweis Gilt $|V| = 1$, so ist nichts zu zeigen. Es sei daher $|V| \geq 2$ und $v_0 \in V$ ein beliebiger Knoten sowie $Z := \{v_0\}$. Da G zusammenhängend ist, gibt es eine Kante von v_0 zu einem $v_1 \neq v_0$ in V. Füge v_1 zu Z hinzu. Da G zusammenhängend ist, lässt sich dieses Argument erneut anwenden, solange $Z \neq V$ gilt, und unter Verwendung einer neuen Kante ein neuer Knoten zu Z hinzufügen. Nach $|V| - 1$ Wiederholungen und unter Verwendung von $|V| - 1$ paarweise verschiedenen Kanten aus E gilt $Z = V$. Folglich hat G mindestens $|V| - 1$ Kanten. □

Achtung: Die Umkehrung des obigen Resultats gilt im Allgemeinen nicht, d.h. man kann nicht von der Anzahl der Kanten auf den Zusammenhang eines Graphen schließen.

3.2 Bäume in ungerichteten Graphen

Wir wollen nun ungerichtete Graphen betrachten, die zwei gegenläufigen Bedingungen genügen sollen. Zum einen sollen sie zusammenhängend sein, müssen nach Lemma 3.7 also mindestens $|V| - 1$ Kante haben, zum anderen aber möglichst wenige Kanten besitzen.

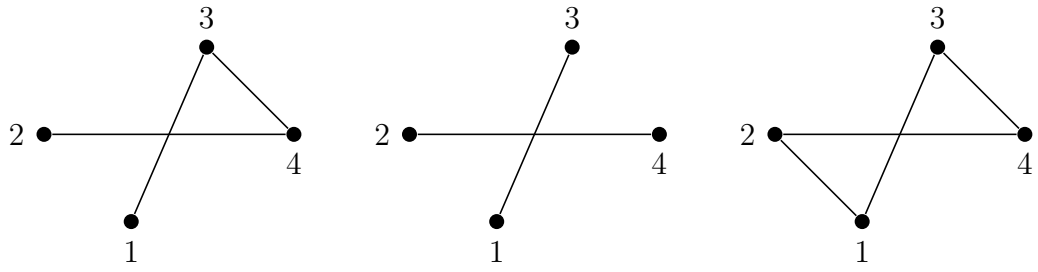
Zur Erinnerung: Ein ungerichteter Graph heißt kreisfrei, wenn er keine einfachen Kreise enthält.

Definition 3.8 *Es sei $T = (V, E)$ ein ungerichteter Graph. Dann heißt T ein* Baum*, wenn T zusammenhängend und kreisfrei ist.*

In der Definition von Bäumen ist wichtig, dass man „nur“ fordert, dass sie kreisfrei sind, also keine einfachen Kreise enthalten. Würde man stattdessen die stärkere Forderung stellen, dass sie überhaupt keine Kreise enthalten, so gäbe es keine Bäume, denn jeder Baum mit mindestens einer Kante enthält (triviale) Kreise.

In manchen Fällen lässt sich die obige Definition nicht gut nachprüfen. Daher wollen wir noch ein paar äquivalente Charakterisierungen betrachten.

Satz 3.9 *Es sei $T = (V, E)$ ein ungerichteter Graph. Dann sind äquivalent:*



(a) Baum (b) nicht zusammenhängend (c) nicht kreisfrei

Abbildung 3.2: Welche Graphen sind Bäume?

(a) T *ist ein Baum, d.h. zusammenhängend und kreisfrei.*

(b) T *ist kreisfrei und durch das Hinzufügen einer beliebigen Kante* $e \notin E$ *entsteht ein einfacher Kreis.*

(c) Je zwei Knoten $v, u \in V$, $v \neq u$, *sind durch genau einen einfachen Weg in* T *verbunden.*

(d) T *ist zusammenhängend und für jede Kante* $e \in E$ *ist* $T' = (V, E \setminus \{e\})$ *nicht zusammenhängend.*

(e) T *ist zusammenhängend und* $|E| = |V| - 1$.

(f) T *ist kreisfrei und* $|E| = |V| - 1$.

Beweis Wir zeigen die Aussage durch einen Ringschluss, d.h. wir zeigen, dass aus Aussagen (a) Aussage (b) folgt, aus Aussage (b) Aussage (c), ..., und aus Aussage (f) schließlich wieder Aussage (a).

(a) $\Longrightarrow$ (b) Sei $e \notin E$ eine beliebige neue Kante, die wir zu T hinzufügen. Dann hat e Endknoten $u, v \in V$. Da T nach Voraussetzung zusammenhängend und kreisfrei ist, gibt es in T einen einfachen Weg P von u nach v. Nimmt man zu P noch die Kante e hinzu, so entsteht ein einfacher Kreis.

(b) $\Longrightarrow$ (c) Angenommen, es gäbe zwei Knoten $u, v \in V$, die nicht durch einen Weg verbunden wären. Dann könnte man zu T die Kante $\{u, v\}$ hinzufügen, ohne dass ein Kreis entsteht, ein Widerspruch zu (b). Also sind in T alle Knoten durch mindestens einen Weg verbunden und dieser ist einfach, da T kreisfrei ist.

Angenommen, es gäbe zwei Knoten $u, v \in V$, die durch mehr als einen einfachen Weg verbunden wären. Dann gäbe es zwei verschiedene einfache Wege P_1, P_2 von u nach v. Verknüpft man diese beiden Wege, so entsteht ein Kreis. Und da $P_1 \neq P_2$ ist, enthält dieser Kreis auch einen einfachen Kreis, ein Widerspruch zu (b).

(c) $\Longrightarrow$ (d) Nach Voraussetzung ist T zusammenhängend. Angenommen, es gäbe eine Kante $e = \{u, v\} \in E$, so dass in $T' = (V, E \setminus \{e\})$ immer noch alle Knoten durch einen Weg verbunden sind. Dann gibt es auch einen einfachen Weg P von u nach v in T' welcher

offensichtlich auch in T liegt. Andererseits sind u und v in T auch durch den einfachen Weg $(u, \{u, v\}, v)$ verbunden, ein Widerspruch zu (c).

(d) $\Longrightarrow$ (e) Nach Voraussetzung ist T zusammenhängend. Nach Lemma 3.7 gilt daher $|E| \geq |V| - 1$. Wir zeigen die noch fehlende Ungleichung $|E| \leq |V| - 1$ durch Induktion über die Anzahl der Knoten $n := |V|$. Für $n = 1$ und $n = 2$ ist sie offensichtlich richtig. Angenommen, sie sei für alle Graphen mit $n - 1$ Knoten, die (d) genügen, richtig. Dann betrachten wir einen Graphen $T = (V, E)$ mit n Knoten. Sei $e \in E$ eine beliebige Kante. Dann ist nach Voraussetzung der Graph $T' = (V, E \setminus \{e\})$ nicht mehr zusammenhängend. Daher zerfällt T' in mindestens zwei Zusammenhangskomponenten $T_1, \ldots, T_p$, $p \geq 2$. Jede Zusammenhangskomponente $T_i = (V_i, E_i)$ hat höchstens $n - 1$ Knoten und ist nach Definition zusammenhängend. Entfernt man aus T_i eine beliebige Kante, so ist T_i nicht mehr zusammenhängend. Sonst könnte man diese Kante auch aus T entfernen, ohne dass T in mehrere Zusammenhangskomponenten zerfallen würde. **(klar?)** Nach Induktionsannahme gilt daher für alle T_i die Formel $|E_i| = |V_i| - 1$. Damit folgt auch für T

$$|E| = \sum_{i=1}^{p} |E_i| + 1 = \sum_{i=1}^{p} (|V_i| - 1) + 1 = |V| - p + 1 \leq |V| - 1.$$

(e) $\Longrightarrow$ (f) Nach Voraussetzung gilt die Gleichung $|E| = |V| - 1$ und T ist zusammenhängend. Angenommen, T besäße einen einfachen Kreis P. Dann könnte man eine in P enthaltene Kante e aus T entfernen und der entstehende Graph $T' = (V, E \setminus \{e\})$ wäre immer noch zusammenhängend **(klar?)**. Dann wäre $T' = (V, E')$ ein zusammenhängender Graph mit $|E'| = |V| - 2$. Nach Lemma 3.7 gilt aber $|E'| \geq |V| - 1$, ein Widerspruch.

(f) $\Longrightarrow$ (a) Nach Voraussetzung ist T kreisfrei, wir müssen also nur zeigen, dass T zusammenhängend ist. Seien dafür wieder $T_1, \ldots, T_p$ die Zusammenhangskomponenten von T. Diese enthalten dann offensichtlich keine einfachen Kreise und sind zusammenhängend, also Bäume. Auf Grund der bisher gezeigten Implikationen folgt aus (e) für alle $T_i = (V_i, E_i)$ die Gleichung $|E_i| = |V_i| - 1$. Aufsummieren dieser Gleichungen ergibt zusammen mit der Voraussetzung $|E| = |V| - 1$

$$|V| - 1 = |E| = \sum_{i=1}^{p} |E_i| = \sum_{i=1}^{p} (|V_i| - 1) = |V| - p.$$

Daher muss $p = 1$ gelten, d.h. T hat nur eine Zusammenhangskomponente und ist folglich zusammenhängend. □

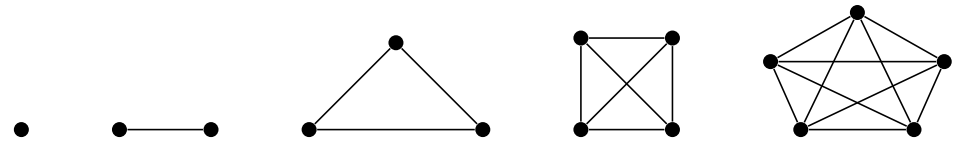
Häufig betrachten wir Bäume, die Teilgraphen eines größeren Graphen sind. Dazu wollen wir zwei bekannte Definitionen kombinieren.

Definition 3.10 *Es sei $G = (V, E)$ ein ungerichteter Graph und $T = (V', E')$ ein Teilgraph von G. Dann heißt T ein* spannender Baum *von G, wenn $V' = V$ gilt und T ein Baum ist.*

Ein Graph besitzt im Allgemeinen mehr als einen spannenden Baum, manchmal sogar sehr viele. Um dies zu illustrieren betrachten wir Graphen mit möglichst vielen Kanten.

Definition 3.11 *Ein ungerichteter Graph $G = (V, E)$ heißt* vollständig*, wenn für je zwei Knoten $v, u \in V$ gilt $\{v, u\} \in E$.*

Einen ungerichteten, einfachen und vollständigen Graphen mit n Knoten bezeichnet man mit K_n.



(a) K_1 (b) K_2 (c) K_3 (d) K_4 (e) K_5

Abbildung 3.3: Einige vollständige Graphen

Wir wollen nun untersuchen, wieviele spannende Bäume ein vollständiger Graph enthält. Dazu benötigen wir das folgende Resultat über Bäume, einen Beweis dazu wollen wir uns in Aufgabe 2 überlegen.

Lemma 3.12 *Es sei $T = (V, E)$ ein ungerichteter Baum mit $|V| \geq 2$. Dann besitzt T mindestens zwei Knoten $v \in V$ mit $g(v) = 1$. Diese nennt man* Blätter.

Mit diesem Hilfsresultat können wir nun den Satz von Cayley beweisen, der angibt, wieviele verschiedene spannende Bäume ein vollständiger Graph mit eindeutig markierten Knoten hat. Sind die Knoten eines vollständigen Graphen nicht markiert, so kann man manche spannenden Bäume nicht mehr unterscheiden, es gibt also weniger.

Satz 3.13 (Satz von Cayley) *Für $n \geq 2$ gibt es n^{n-2} verschiedene spannende Bäume in einem vollständigen Graphen K_n mit eindeutig markierten Knoten.*

Beweis Ein Prüfer-Code ist ein Vektor $(a_1, a_2, \ldots, a_{n-1})$ mit $1 \leq a_i \leq n$ für alle $i = 1, \ldots, n-1$. Wir werden nun zeigen, dass man jedem spannenden Baum von K_2 einen eindeutigen Prüfer-Code zuordnen kann.

Da die Knoten von K_n eindeutig markiert sind, können wir sie durchnummerieren, d.h. sei $V = \{v_1, v_2, \ldots, v_n\}$.

Wie kommt man nun von einen Baum T zum Prüfer-Code? Eine Möglichkeit ist das in Algorithmus 4 beschriebene Verfahren. Nach Lemma 3.12 besitzt T immer mindestens zwei Blätter. In jeder Iteration gibt es daher ein eindeutig bestimmtes Blatt v_i mit minimalem Index. Da v_i ein Blatt ist, also Grad 1 hat, besitzt es einen eindeutigen Nachbarn und durch das Entfernen von v_i erhält man wieder einen Baum mit einem Knoten weniger. Folglich ist der Algorithmus wohldefiniert, bricht nach $n-1$ Iterationen ab und erzeugt einen Prüfer-Code.

Da jeder Baum mindestens zwei Blätter hat, ist der Index des Blattes mit niedrigstem Index nie n. Daher ist v_n der Knoten, der bis zum Schluss im Graphen T verbleibt, d.h. es gilt immer $a_{n-1} = n$. Nun sieht man schon, warum Prüfer-Codes interessant sind, denn es gibt genau n^{n-2} Prüfer-Codes $(a_1, \ldots, a_{n-2}, a_{n-1})$ mit $a_{n-1} = n$.

Seien nun $(a_1, \ldots, a_{n-1})$ der Prüfer-Code zu einem Baum T und $(b_1, \ldots, b_{n-1})$ die Indizes der entfernten Blätter in der Reihenfolge des Entfernens. Mit T_i bezeichnen wir den Graphen T vor dem Entfernen des Knotens b_i, d.h. zu Beginn der Prüfer-Code Erzeugung ist $T = T_1$ und am Ende bleibt T_{n-1} übrig. Dann gilt für alle $i = 1, \ldots, n-1$

$$b_i = \min \underbrace{\{k \in \{1, \ldots, n\} \mid k \neq b_1, \ldots, b_{i-1}, a_i, \ldots, a_{n-1}\}}_{=:M_i},$$

d.h. b_i ist der kleinste Index, der nicht gleich dem Index eines bereits entfernten Blattes oder gleich dem Index einer zukünftig notierten Nachbarknoten ist. Dass b_i nicht gleich einem dieser Indizes sein kann, folgt daraus, dass bereits entfernte Blätter nicht mehr im Graphen sind und zukünftig markierte Nachbarknoten noch nicht entfernt werden können. (Beachte, dass die b_i zwar paarweise verschieden sind, die a_i jedoch nicht notwendig.) Wir müssen also nur zeigen, dass alle Knoten v_j mit Index $j \in M_i$ im Graphen T_i Blätter sind. Für $i = n-1$ ist dies offensichtlich der Fall, da

$$M_{n-1} = \{k \in \{1, \ldots, n\} \mid k \neq b_1, \ldots, b_{n-2}, a_{n-1}\}$$

gilt und folglich genau ein Element, nämlich b_{n-1}, hat. Gilt für ein M_i, dass alle Knoten v_j mit Index $j \in M_i$ im Graphen T_i Blätter sind, so zeigen wir, dass dies dann auch für M_{i-1} gilt. Es ist $M_{i-1} = (M_i \setminus \{a_{i-1}\}) \cup \{b_{i-1}\}$. Alle Knoten v_j mit $j \in M_i \setminus \{a_{i-1}\}$ haben in T_{i-1} den gleichen Grad wie in T_i, sind also Blätter. Wir müssen also nur zeigen, dass auch der Knoten mit Index b_{i-1} in T_{i-1} ein Blatt ist. Dies ist aber offensichtlich der Fall, da $v_{b_{i-1}}$ das aus dem Graphen T_{i-1} entfernte Blatt ist.

Dies bedeutet aber, dass jeder Baum T aus seinem Prüfer-Code eindeutig rekonstruierbar ist, zum Beispiel mit Algorithmus 5. Wenn wir jetzt noch zeigen, dass jeder nach Algorithmus 5 erzeugte Graph ein Baum ist, so haben wir gezeigt, dass es genau so viele Bäume mit n Knoten gibt wie Prüfer-Codes, also n^{n-2}. Da die Bäume mit n Knoten aber genau die spannenden Bäume von K_n sind, folgt der Satz von Cayley.

Sei daher P ein beliebiger Prüfer-Code und T der zugehörige von Algorithmus 5 erzeugte Graph. Dann gilt nach Konstruktion $b_i \neq b_j$ für alle $i \neq j$, d.h. T ist ein einfacher Graph mit n Knoten und $n-1$ Kanten. Wenn wir also zeigen können, dass T zusammenhängend ist, folgt wie gewünscht, dass T ein Baum ist. Wir zeigen dies per Induktion über die Knotenzahl n. Für $n = 2$ ist der entstehende Graph T offensichtlich zusammenhängend. Ist der entstehende Graph für $n-1$ Knoten zusammenhängend, so betrachten wir einen erzeugten Graphen mit n Knoten. Hier gilt $b_1 \neq a_i$ für alle $i = 1, \ldots, n-1$, d.h. der Knoten mit Index b_1 ist nur zu a_1 benachbart. Entfernen wir also diesen Knoten und die dazu inzidente Kante aus T, wird der resultierende Graph T' von dem Prüfer-Code $(a_2, \ldots, a_{n-2})$ über der Knotenmenge $V \setminus \{v_{b_1}\}$ erzeugt und ist nach Induktionsvoraussetzung zusammenhängend. Daher ist auch T zusammenhängend und folglich ein Baum. □

Algorithmus 4 Prüfer-Code-Generator

Input: Ein Baum $T = (V, E)$ mit nummerierten Knoten $V = \{v_1, v_2, \ldots, v_n\}$.

1 Setze den Zähler $k = 1$.
2 **while** $|V| \geq 2$ **do**
3 Bestimme das Blatt v_i mit dem kleinsten Index i.
4 Notiere den Index j des eindeutigen Nachbarn v_j von v_i in a_k.
5 Entferne den Knoten v_i und die dazu inzidente Kante $\{v_i, v_j\}$ aus T und erhöhe den Zähler k um 1.
6 **end while**

Output: Ein Prüfer-Code $P = (a_1, \ldots, a_{n-1})$

Algorithmus 5 Prüfer-Code-Interpreter

Input: Ein Prüfer-Code $P = (a_1, \ldots, a_{n-1})$.

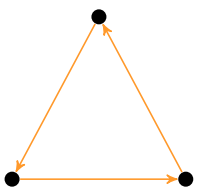
1 Setze $V = \{v_1, v_2, \ldots, v_n\}$ und $E = \emptyset$.
2 **for** $i = 1$ **to** $n - 1$ **do**
3 Bestimme $b_i = \min\{k \in \{1, \ldots, n\} \mid k \neq b_1, \ldots, b_{i-1}, a_i, \ldots, a_{n-1}\}$.
4 Füge die Kante $\{v_{b_i}, v_{a_i}\}$ zu E hinzu.
5 **end for**

Output: Ein Baum $T = (V, E)$.

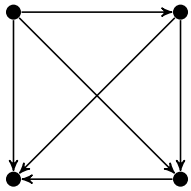
3.3 Bäume in gerichteten Graphen

Nun wollen wir definieren, was man unter Bäumen in gerichteten Graphen versteht und wo die Unterschiede liegen. Zur Erinnerung: Ein ungerichteter Graph ist ein Baum genau dann, wenn er zusammenhängend und kreisfrei ist.

Möchte man diese Definition auf gerichtete Graphen übertragen, so muss man sich zum einen überlegen, ob ein gerichteter Baum stark oder schwach zusammenhängend sein soll, und zum anderen, ob er nur keine gerichteten einfache Kreise enthalten darf oder auch keine ungerichteten einfachen Kreise.



(a) stark zusammenhängend, mit gerichtetem einfachem Kreis



(b) schwach zusammenhängend, ohne gerichtete einfache Kreise

Abbildung 3.4: Zusammenhängende gerichtete Graphen

Unglücklicherweise widersprechen sich diese Forderungen teilweise, denn jeder stark zusammenhängende Graph mit mindestens zwei Knoten enthält sogar gerichtete einfache

Kreise, vergleiche Abbildung 3.4 (a). Daher macht es in gerichteten Graphen keinen Sinn zu fordern, dass Bäume stark zusammenhängend sind.

Eine schöne Eigenschaft von Bäumen $T = (V, E)$ in ungerichteten Graphen ist, dass sie immer genau $|E| = |V| - 1$ Kanten haben. Ein gerichteter, schwach zusammenhängender Graph, der keine gerichteten einfachen Kreise enthält, kann aber mehr Kanten enthalten, vergleiche Abbildung 3.4 (b). (Weniger Kanten kann er wegen des schwachen Zusammenhangs nicht haben.) Daher stellt man die stärkere Forderung, dass ein Baum auch keine ungerichteten einfachen Kreise enthalten darf. Dies führt zur folgenden Definition.

Definition 3.14 *Es sei $T = (V, E)$ ein gerichteter Graph. T heißt* Baum*, wenn der zugeordnete ungerichtete Graph ein Baum ist.*

(a) Baum (b) Wurzelbaum

Abbildung 3.5: Gerichtete Bäume

Bäume in gerichteten Graphen sind also schwach zusammenhängend und kreisfrei, d.h. sie enthalten keine ungerichteten einfachen Kreise. Der Rückzug auf schwachen Zusammenhang hat allerdings den Nachteil, dass in einem gerichteten Graphen $T = (V, E)$ für zwei beliebige Knoten $s, t \in V$ nicht garantiert ist, dass es in T einen gerichteten Weg von s nach t gibt, vergleiche Abbildung 3.5(a). In manchen Anwendungen, z.B. dem Telefonketten-Beispiel, ist dies jedoch wichtig. Daher betrachtet man häufig die folgenden Graphen, vergleiche auch Abbildung 3.5(b)

Definition 3.15 *Es sei $T = (V, E)$ ein gerichteter Graph. T heißt* Wurzelbaum *mit Wurzel s, wenn T ein Baum ist, in dem s eine Wurzel ist.*

Wurzelbäume lassen sich auch wie folgt charakterisieren:

Satz 3.16 *Es sei $T = (V, E)$ ein gerichteter Graph und $s \in V$. Dann sind äquivalent:*

(a) T ist ein Wurzelbaum mit Wurzel s.

(b) T ist ein Baum, $g^-(s) = 0$ und $g^-(v) = 1$ für alle $v \in V \setminus \{s\}$.

(c) s ist eine Wurzel in T, $g^-(s) = 0$ und $g^-(v) \leq 1$ für alle $v \in V \setminus \{s\}$.

Beweis Wir zeigen die Aussage wieder durch einen Ringschluss.

(a) $\Longrightarrow$ (b) Da alle Knoten $v \in V \setminus \{s\}$ von s aus durch einen gerichteten Weg erreichbar sind, folgt für sie $g^-(v) \geq 1$. Weil außerdem der zu T gehörende ungerichtete Graph ein Baum ist, folgt

$$|V| - 1 = |E| = \underbrace{g^-(s)}_{\geq 0} + \sum_{v \in V \setminus \{s\}} \underbrace{g^-(v)}_{\geq 1} \geq |V| - 1.$$

Folglich muss $g^-(s) = 0$ und $g^-(v) = 1$ für alle $v \neq s$ gelten.

(b) $\Longrightarrow$ (c) Wir müssen nur zeigen, dass s eine Wurzel von T ist. Sei dafür $v \in V$ beliebig. Dann müssen wir zeigen, dass v durch einen gerichteten Weg von s aus erreichbar ist. Für $v = s$ ist nicht zu zeigen, sei also $v \neq s$. Da T schwach zusammenhängend ist, gibt es einen, möglicherweise ungerichteten, einfachen Weg $P = (s = v_0, e_1, v_1, e_2, \ldots, e_l, v_l = v)$. Wegen $g^-(s) = 0$ folgt $e_1 = (v_0, v_1)$, d.h. die erste Kante auf dem Weg zeigt in die richtige Richtung. Wegen $g^-(v_1) = 1$ und $v_2 \neq v_0$ muss dann aber auch $e_2 = (v_1, v_2)$ gelten, d.h. auch die zweite Kante zeigt in die richtige Richtung. Iterativ zeigt man so, dass P tatsächlich ein gerichteter Weg ist.

(c) $\Longrightarrow$ (a) Wir müssen zeigen, dass T ein Baum ist. Nach Voraussetzung ist s eine Wurzel in T, daher ist T schwach zusammenhängend und es gilt $|E| \geq |V| - 1$. Andererseits gilt auch

$$|E| = \underbrace{g^-(s)}_{=0} + \sum_{v \in V \setminus \{s\}} \underbrace{g^-(v)}_{\leq 1} \leq |V| - 1,$$

daher ist T ein Baum. □

3.4 Aufgaben

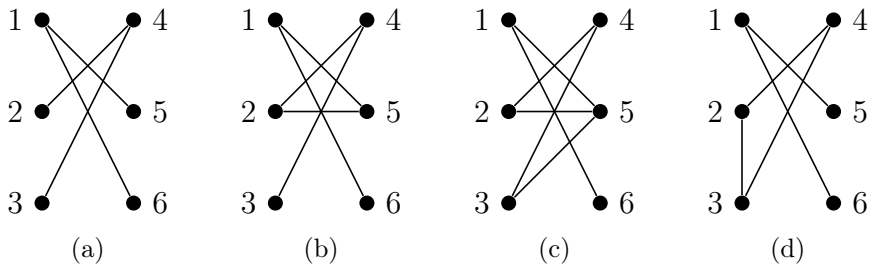


Abbildung 3.6: Welche Graphen sind Bäume?

Aufgabe 3.1 Welche der ungerichteten Graphen in Abbildung 3.6 sind Bäume? Begründen Sie Ihre Antworten.

Aufgabe 3.2 Beweisen Sie Lemma 3.12: Es sei $T = (V, E)$ ein ungerichteter Baum mit $|V| \geq 2$. Dann besitzt T mindestens zwei verschiedene Knoten $v, w \in V$ mit $g(v) = g(w) = 1$. Diese nennt man *Blätter*.

Aufgabe 3.3 Gibt es einen Baum mit 6 Knoten und $0, 1, 2, 3, 4, 5, 6$ Blättern? Begründen Sie Ihre Antworten.

Aufgabe 3.4 Folgern Sie aus Aufgabe 3.3 einen Zusammenhang zwischen der Anzahl von Blättern und dem höchsten auftretenden Knotengrad in einem Baum und beweisen Sie diesen.

Sendemast	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
Koordinaten	(0,2)	(2,0)	(3,4)	(5,1)	(4,3)	(7,4)	(8,2)	(10,2)
Sendestärke	4	4	2	3	5	4	3	5

Tabelle 3.1: Koordinaten und Sendereichweite der Funkmasten

Aufgabe 3.5 Sie besitzen 8 Sendemasten, deren Position und Sendestärke Sie Tabelle 3.1 entnehmen können. Jeder Mast kann ein Signal aussenden, welches von allen andern Masten innerhalb seines Senderadius empfangen werden kann. Mit Hilfe dieser Masten wollen Sie ein Signal von Mast A an alle anderen Masten senden.

(a) Beschreiben Sie das Problem graphentheoretisch.

(b) Aus Kostengründen wollen Sie das Signal von möglichst wenigen Masten weitersenden. Von wievielen Masten müssen Sie es mindestens senden?

(c) Die Kosten dafür, das Signal von einem Mast aus zu senden, seien proportional zu seiner Reichweite. Welche Masten verwenden Sie zu Weiterleitung, wenn Sie die Gesamtkosten möglichst klein halten wollen?

4 Minimale spannende Bäume

In vielen Anwendungsproblemen geht es darum gewisse Knoten mit möglichst wenigen Kanten so zu verbinden, dass zwischen allen Paaren von Knoten ein Weg existiert.

Beispiel 4.1 (Magnetschwebebahn) Erinnern wir uns wieder an das Beispiel mit der Magnetschwebebahn, so können wir alle Städte als Knoten modellieren und die dazwischen technisch möglichen Verbindungen als Kanten. Wenn wir davon ausgehen, dass die Bahn immer in beide Richtungen fährt, erhalten wir so einen ungerichteten Graphen. Aus Kostengründen wollen wir nur möglichst wenige Strecken tatsächlich bauen, d.h. wir suchen einen spannenden Baum. Wir wissen also schon, dass wir mindestens „Anzahl der Städte -1“ Strecken realisieren müssen. Nun kosten die möglichen Strecken aber nicht alle gleich viel. Welchen spannenden Baum sollen wir also wählen? ♢

Beispiel 4.2 (Internetanbindung) Vor einem ähnlichen Problem stehen viele Telefongesellschaften aktuell: Als vor langer Zeit die ersten Telefonkabel verlegt wurden, dachte natürlich noch niemand daran, dass diese Kabel irgendwann dazu dienen würden, Haushalten den Zugang zum Internet zu ermöglichen. Daher liegen vielerorts nicht die für schnelles Internet nötigen Glasfaserkabel sondern alte Kupferkabel. Da der Bedarf an schnellen Internetzugängen immer weiter steigt, sehen sich die Telefongesellschaften gezwungen die alten Leitungen zumindest teilweise durch neue zu ersetzten. Dies ist aber ein kostspieliges Unterfangen, weniger wegen der Kosten für die neuen Glasfaserkabel als wegen der vor allem in Städten hohen Baukosten. Daher haben die Telefongesellschaften ein großes Interesse daran nur an strategisch wichtigen Stellen neue Leitungen zu verlegen. ♢

Beispiel 4.3 (Broadcasting) Im Gegensatz zu den beiden vorherigen ungerichteten Beispielen gibt es auch ähnliche Probleme in denen Verbindungen nur in eine Richtung genutzt werden können. Betrachten wir zum Beispiel ein Netzwerk von Sendemasten unterschiedlicher Reichweiten, in dem eine Information kostengünstig von einem Mast an alle anderen weitergegeben werden soll, so erhalten wir einen gerichteten Graphen, in dem wir einen spannenden Wurzelbaum suchen. ♢

Obwohl es also auch solche Anwendungen gibt, bei denen man spannende Bäume in gerichteten Graphen sucht, wollen wir uns in diesem Kapitel wie schon bei Bäumen auf ungerichtete Graphen einschränken. Viele der hier vorgestellten Ideen lassen sich aber auch in geeigneter Art auf gerichtete Graphen übertragen.

4.1 Gewichtete Graphen und minimale spannende Bäume

Erinnern wir uns an die Breiten- und Tiefensuche, so waren die von diesen Algorithmen erzeugten Vorgängergraphen $G_\pi = (V_\pi, E_\pi)$ spannende Bäume in ungerichteten Graphen beziehungsweise spannende Wurzelbäume in gerichteten Graphen. Warum also betrachten wir dieses Problem erneut?

Wie vorhin schon erwähnt, verursacht die Realisierung einer Kante in Anwendungsbeispielen nicht immer für alle Kanten die gleichen Kosten. Man interessiert sich daher nicht für irgendeinen spannenden Baum, sondern sucht einen kostenminimalen. Was genau wir darunter verstehen wollen, definieren wir jetzt.

Definition 4.4 *Es sei $G = (V, E)$ ein ungerichteter oder gerichteter Graph. G heißt* gewichtet*, wenn jeder Kante $e \in E$ ein Gewicht $w(e)$ zugeordnet ist.*

Definition 4.5 *Es sei $G = (V, E)$ gewichteter ungerichteter Graph. Ein Teilgraph $T = (V, E_T)$ von G heißt* minimaler spannender Baum *von G, wenn sein Gewicht $w(T) := \sum_{e \in E_T} w(e)$ minimal unter allen spannenden Bäumen von G ist.*

Der Begriff minimal bezieht sich hier also nicht auf die Anzahl der Kanten, die ja in jedem Baum gleich ist, sondern auf das Gesamtgewicht des Baumes.

Da es in jedem Graphen nur endlich viele spannende Bäume gibt, vergleiche Satz 3.13, gibt es immer mindestens einen minimalen spannenden Baum. Aber wie kann man einem spannenden Baum ansehen, ob er minimal ist?

Satz 4.6 *Es sei $G = (V, E)$ ein ungerichteter, gewichteter und zusammenhängender Graph mit $n = |V|$ Knoten und Kantengewichten $w(e)$ für alle $e \in E$ und $T = (V, E_T)$ ein spannender Baum. Dann sind äquivalent:*

(a) T ist ein minimaler spannender Baum.

(b) Für jede Kante $e \in E \setminus E_T$ gilt: e ist eine Kante mit maximalem Gewicht in dem eindeutigen einfachen Kreis, der entsteht, wenn man e zu T hinzufügt.

(c) Für jede Kante $e_T \in E_T$ gilt: Entfernt man e_T aus T, so zerfällt T in zwei Zusammenhangskomponenten Z_1 und Z_2. e_T ist eine Kante mit minimalem Gewicht, die in beiden Zusammenhangskomponenten jeweils einen Endknoten hat.

Beweis Wir zeigen die Aussage durch einen Ringschluss.

(a) $\Longrightarrow$ (b) Sei T ein minimaler spannender Baum und $e = \{v, u\} \in E \setminus E_T$ beliebig. Da die Knoten v, u im Baum T durch einen eindeutigen einfachen Weg P verbunden sind, entsteht durch das Hinzufügen der Kante e zu T ein eindeutiger einfacher Kreis. Angenommen, es gibt eine Kante $e_T \in P$ mit $w(e_T) > w(e)$. Dann betrachten wir den Graphen $T' = (V, (E \setminus \{e_T\}) \cup \{e\})$. Es gilt offensichtlich $w(T') < w(T)$. Außerdem hat

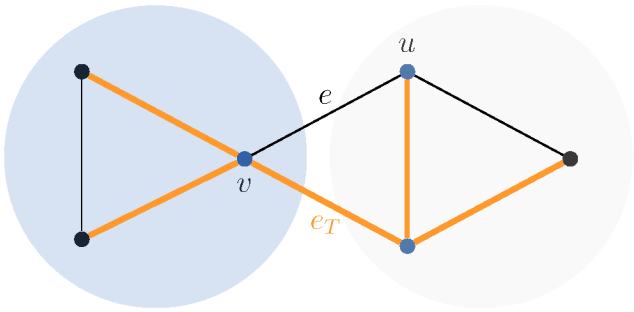


Abbildung 4.1: Illustration des Beweises von Satz 4.6

T' wieder $n-1$ Kanten und ist zusammenhängend. **(klar?)** Folglich ist T' ein spannender Baum mit einem kleineren Gewicht als T, ein Widerspruch.

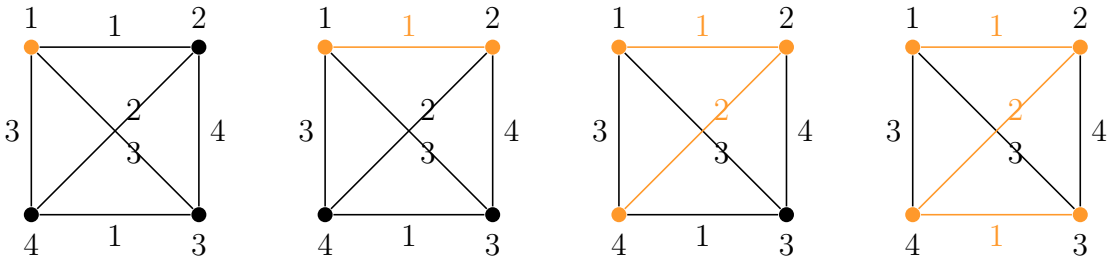
(b) $\Longrightarrow$ **(c)** Es sei T ein spannender Baum mit der Eigenschaft aus (b). Angenommen, es gibt eine Kante $e_T \in E_T$, für die (c) verletzt ist, d.h. es gibt eine Kante e mit Endknoten in Z_1 und Z_2, für die gilt $w(e) < w(e_T)$. Da e und e_T beide die gleiche Zusammenhangskomponenten verbinden, muss $e \notin E_T$ gelten. Fügen wir die Kante e zu T hinzu, so entsteht ein eindeutiger einfacher Kreis, auf dem auch e_T liegen muss, da beide Kanten die Zusammenhangskomponenten Z_1 und Z_2 verbinden. Dies ist aber ein Widerspruch zu (b), da e nicht die Kante mit maximalem Gewicht auf diesem Kreis ist.

(c) $\Longrightarrow$ **(a)** Es sei T ein spannender Baum mit Eigenschaft (c) und $T^* = (V, E^*)$ ein minimaler spannender Baum, der maximal viele Kanten mit T gemeinsam hat. Da T und T^* spannende Bäume sind, haben sie gleich viele Kanten. Gilt $T \neq T^*$, so muss es also mindestens eine Kante $e_T \in E_T \setminus E^*$ geben. Entfernen wir e_T aus T, so zerfällt T in zwei Zusammenhangskomponenten Z_1 und Z_2. Da T^* ein spannender Baum ist, muss es eine Kante $e^* \in E^*$ geben, die Endknoten in Z_1 und Z_2 hat. Wegen $e_T \notin E^*$ folgt $e^* \neq e_T$ und daher $w(e^*) \geq w(e_T)$. Betrachten wir nun den Teilgraphen $T' = (V, (E^* \setminus \{e^*\}) \cup \{e_T\})$, so gilt folglich $w(T') \leq w(T^*)$ und T' ist ebenfalls ein spannender Baum (vergleiche (a) $\Longrightarrow$ (b)). Da T^* ein minimaler spannender Baum ist, muss sogar $w(T') = w(T^*)$ gelten. Wir haben also einen minimalen spannenden Baum T' konstruiert, der eine Kante mehr mit T gemeinsam hat als T^*, ein Widerspruch. Folglich muss $T = T^*$ gelten, d.h. T ist ein minimaler spannender Baum. □

Da ein Graph nur endlich viele spannende Bäume enthält, könnte man einen minimalen finden, indem man alle spannenden Bäume bestimmt und daraus den mit dem kleinsten Gewicht wählt. In größeren Graphen gibt es jedoch sehr viele spannende Bäume, vergleiche Satz 3.13. Es ist daher zu zeitaufwändig alle zu bestimmen und zu vergleichen. Wie also kann man einen minimalen spannenden Baum schneller finden?

4.2 Algorithmus von Prim

Eine einfache Idee ist die folgende: Starte mit einem beliebigen Knoten. Da ein minimaler spannender Baum zusammenhängend ist, muss von diesem Knoten eine Kante ausgehen.



(a) Start (b) 1. Iteration (c) 2. Iteration (d) 3. Iteration

Abbildung 4.2: Illustration des Algorithmus von Prim

Wähle hierfür diejenige mit kleinstem Gewicht. Nun hat man einen Teilgraphen mit zwei Knoten und einer Kante. Von diesem Teilgraphen muss wiederum eine Kante ausgehen zu einem noch nicht enthaltenen Knoten. Wähle hierfür wieder die mit kleinstem Gewicht und setze das Verfahren fort, bis alle Knoten verbunden sind, vergleiche Abbildung 4.2.

Formalisiert man dieses Verfahren, so erhält man den *Algorithmus von Prim*, vergleiche Algorithmus 6.

Algorithmus 6 Algorithmus von Prim

Input: Ein ungerichteter, gewichteter und zusammenhängender Graph $G = (V, E)$ mit $n := |V|$ Knoten und Kantengewichten $w(e)$ für alle $e \in E$.

1 Setze die Startknotenmenge auf $V_1 = \{s\}$ mit einem beliebigen $s \in V$ und die Startkantenmenge auf $E_1 = \emptyset$.
2 **for** $k = 1$ **to** $n - 1$ **do**
3 Wähle aus allen Kanten $e = \{v, u\} \in E$ mit $v \in V_k$ und $u \in V \setminus V_k$ diejenige Kante $e^* = \{v^*, u^*\}$ mit minimalem Gewicht $w(e^*)$.
4 Setze $V_{k+1} = V_k \cup \{u^*\}$ und $E_{k+1} = E_k \cup \{e^*\}$
5 **end for**

Output: Den Teilgraphen $T := (V_n, E_n)$

Satz 4.7 *Es sei $G = (V, E)$ ein ungerichteter, gewichteter und zusammenhängender Graph mit $n := |V|$ Knoten und Kantengewichten $w(e)$ für alle $e \in E$. Dann ist der Algorithmus 6 wohldefiniert und der erzeugte Teilgraph $T := (V_n, E_n)$ ein minimaler spannender Baum von G.*

Beweis Nach Konstruktion sind alle Teilgraphen $T_k := (V_k, E_k)$ kreisfrei und zusammenhängend. Jedes T_k enthält k Knoten, d.h. solange $k < n$ gilt, ist $V \setminus V_k \neq \emptyset$. Da G zusammenhängend ist, gibt es folglich Kanten $e = \{v, u\} \in E$ mit $v \in V_k$ und $u \in V \setminus V_k$. Daher ist der Algorithmus wohldefiniert und $T = T_n$ ist ein spannender Baum von G.

Wir zeigen per Induktion über k, dass jedes T_k Teilgraph eines minimalen spannenden Baums von G ist. Da $T = T_n$ selbst ein spannender Baum ist, folgt dann, dass T ein minimaler spannender Baum von G ist. Für $k = 1$ ist $T_1 = (\{v_1\}, \emptyset)$ ein Teilgraph jedes

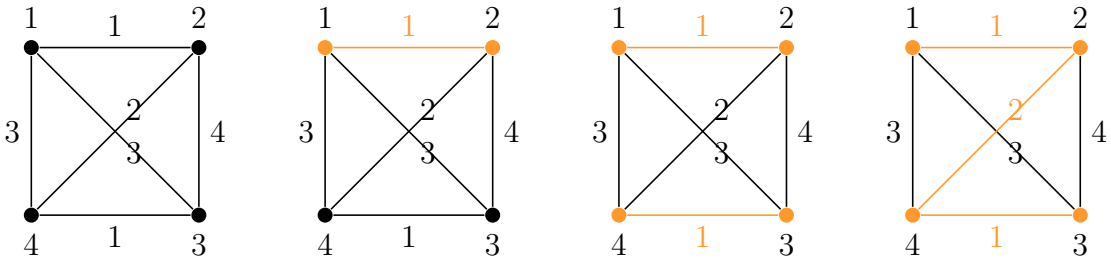
minimalen spannenden Baums von G, da dieser alle Knoten enthalten muss. Sei nun für ein k das zugehörige T_k Teilgraph eines minimalen spannenden Baums $T_G = (V, E_G)$ von T. Dann betrachten wir T_{k+1} mit $V_{k+1} = V_k \cup \{u^*\}$ und $E_{k+1} = E_k \cup \{e^*\}$. Ist $e^* \in E_G$, so ist auch V_{k+1} Teilgraph des minimalen spannenden Baums T_G. Ist $e^* \notin E_G$, so enthält der Graph $(V, E_G \cup \{e^*\})$ nach Satz 3.9 (b) einen einfachen Kreis P, d.h. es gibt außer e^* noch eine andere Kante $\tilde{e} \in P$, die aus V_k heraus führt. Betrachte nun den Graphen $T'_G := (V, (E_G \setminus \{\tilde{e}\}) \cup \{e^*\})$. Dieser enthält keine einfachen Kreise und hat $n-1$ Kanten, ist also ein spannender Baum von G. Für sein Gewicht gilt nach Wahl von e^*

$$w(T_G) \leq w(T'_G) = w(T_G) - w(\tilde{e}) + w(e^*) \leq w(T_G) - w(\tilde{e}) + w(\tilde{e}) = w(T_G),$$

d.h. T'_G ist ein minimaler spannender Baum von G, der nach Konstruktion T_k enthält. □

Wir wollen noch einmal kurz zur Breitensuche zurückgehen und diese mit dem Algorithmus von Prim vergleichen. Beide Verfahren gehen von einem Startknoten aus und erzeugen eine Folge von zusammenhängenden Teilgraphen, die schließlich in einem spannenden Baum endet. Die Breitensuche operiert auf ungewichteten Graphen und erzeugt einen spannenden Baum, der kürzeste (im Sinne von Kantenzahl) Wege vom Startknoten zu allen anderen Knoten enthält. Im Gegensatz dazu arbeitet der Algorithmus von Prim mit gewichteten Graphen und erzeugt einen spannenden Baum, dessen Gesamtgewicht minimal ist. In diesem Baum sind die eindeutigen Wege vom Startknoten zu allen anderen Knoten nicht notwendig am kürzesten (im Sinne von Gewicht), vergleiche Abbildung 4.2. Kürzeste Wege in gewichteten Graphen werden wir im nächsten Kapitel untersuchen.

4.3 Algorithmus von Kruskal



(a) Start (b) 1. Iteration (c) 2. Iteration (d) 3. Iteration

Abbildung 4.3: Illustration des Algorithmus von Kruskal

Der Algorithmus von Prim erzeugt ausgehend von einem Knoten eine immer größere Zusammenhangskomponente mit möglichst kleinem Gewicht. Bei den entstehenden Teilgraphen wird also der Fokus darauf gelegt, dass sie zusammenhängend sind. Dass sie gleichzeitig auch kreisfrei sind, folgt daraus, dass man die Zusammenhangskomponente in jedem Schritt vergrößern will.

Man kann sich daher die Frage stellen, ob man die Prioritäten auch umkehren kann: Kann man stattdessen eine Folge von Teilgraphen erzeugen, bei denen man immer eine Kante mit möglichst kleinem Gewicht dazu nimmt, solange der Graph kreisfrei bleibt, vergleiche Abbildung 4.3? Die zwischendrin entstehenden Teilgraphen wären dann nicht notwendig zusammenhängend. Der letzte jedoch muss zusammenhängend sein, da man sonst noch eine weitere Kante hinzu nehmen könnte.

Formalisiert man diese Idee, so erhält man den Algorithmus von Kruskal, siehe Algorithmus 7.

Algorithmus 7 Algorithmus von Kruskal

Input: Ein ungerichteter, gewichteter und zusammenhängender Graph $G = (V, E)$ mit $n := |V|$ Knoten und Kantengewichten $w(e)$ für alle $e \in E$.

1 Setze die Startknotenmenge auf $V_1 = \emptyset$ und die Startkantenmenge auf $E_1 = \emptyset$ und $T_1 = (V_1, E_1)$.

2 **for** $k = 1$ **to** $n - 1$ **do**

3 Wähle aus allen Kanten $e = \{v, u\} \in E \setminus E_k$ diejenige Kante $e^* = \{v^*, u^*\}$ mit minimalem Gewicht $w(e^*)$, so dass $T_{k+1} = (V_{k+1}, E_{k+1})$ mit $V_{k+1} = V_k \cup \{v^*, u^*\}$ und $E_{k+1} = E_k \cup \{e^*\}$ keine einfachen Kreise enthält.

4 **end for**

Output: Den Teilgraphen $T := (V_n, E_n)$.

Satz 4.8 *Es sei $G = (V, E)$ ein ungerichteter, gewichteter und zusammenhängender Graph mit $n := |V|$ Knoten und Kantengewichten $w(e)$ für alle $e \in E$. Dann ist der Algorithmus 7 wohldefiniert und der erzeugte Teilgraph $T := (V_n, E_n)$ ein minimaler spannender Baum von G.*

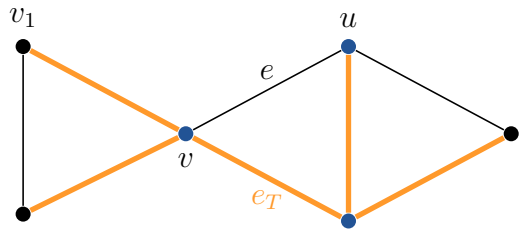


Abbildung 4.4: Illustration des Beweises von Satz 4.8

Beweis Nach Konstruktion bestehen alle Teilgraphen $T_k := (V_k, E_k)$ aus $k-1$ Kanten und enthalten keine einfachen Kreise. Solange $k < n$ gilt, ist $E \setminus E_k \neq \emptyset$, da G als zusammenhängender Graph mindestens $n - 1$ Kanten enthält, vergleiche Lemma 3.7. Folglich gibt es auch immer mindestens eine Kante in $E \setminus E_k \neq \emptyset$, so dass T_{k+1} keinen Kreis enthält, denn sonst wäre T_k schon ein spannender Baum von G mit $k - 1 < n - 1$ Kanten, ein Widerspruch zu Lemma 3.7. Folglich ist der Algorithmus wohldefiniert und $T = T_n$ ist ein

Teilgraph mit $n-1$ Kanten, der keine einfachen Kreise enthält. Nach Satz 3.9 (f) ist T daher ein spannender Baum von G.

Es bleibt zu zeigen, dass T ein minimaler spannender Baum ist. Wir verwenden dafür Satz 4.6. Sei $e = \{v, u\} \in E \setminus E_n$ eine beliebige Kante. Dann gibt es in T einen eindeutigen einfachen Weg P von v nach u. Statt jeder Kante $e_T \in P$ hätte in der Iteration, in der e_T ausgewählt wurde, auch die Kante e ausgewählt werden können, denn $T' = (V, (E_n \setminus \{e_T\}) \cup \{e\})$ ist ebenfalls ein spannender Baum, insbesondere also kreisfrei. Da aber e_T ausgewählt wurde, muss $w(e_T) \leq w(e)$ gelten, d.h. e ist die Kante mit maximalem Gewicht auf dem aus P und e gebildeten einfachen Kreis. Da dies für alle Kanten $e \in E \setminus E_n$ gilt, ist T nach Satz 4.6 ein minimaler spannender Baum.

□

Die Algorithmen von Prim und Kruskal sind sogenannte Greedy-Verfahren. Als *Greedy-Verfahren* bezeichnet man einen Algorithmus, der versucht ein Problem zu lösen, indem er in jeder Iteration das tut, was gerade am besten erscheint. Dies muss global gesehen nicht die beste Entscheidung sein und führt auch nicht immer zum Erfolg, wie man an dem folgenden Beispiel sieht.

Ein Greedy-Verfahren um von einem Berggipfel herunter zu finden wäre zum Beispiel immer in die Richtung des steilsten Abstiegs[1] zu gehen. Wenn man Glück hat, kommt man so am Fuße des Berges an. Hat man hingegen Pech, so landet man in einer Senke, aus der man mit diesem Ansatz nicht mehr herausfindet.

Dass Greedy-Verfahren zur Bestimmung minimaler spannender Bäume funktionieren liegt, etwas vereinfacht, daran, dass man einen minimalen spannenden Baum eines Graphen aus minimalen spannenden Bäumen von Teilgraphen zusammenbauen kann.

4.4 Aufgaben

Aufgabe 4.1 Welche der spannenden Bäume in Abbildung 4.5 sind minimal? Wenn nicht, wie lassen Sie sich zu minimalen spannenden Bäumen abändern?

Aufgabe 4.2 Es sei $G = (V, E)$ ein ungerichteter, zusammenhängender und gewichteter Graph mit zwei Sätzen Kantengewichten $w(e), c(e)$ für alle $e \in E$, so dass für alle $e, e' \in E$ gilt:

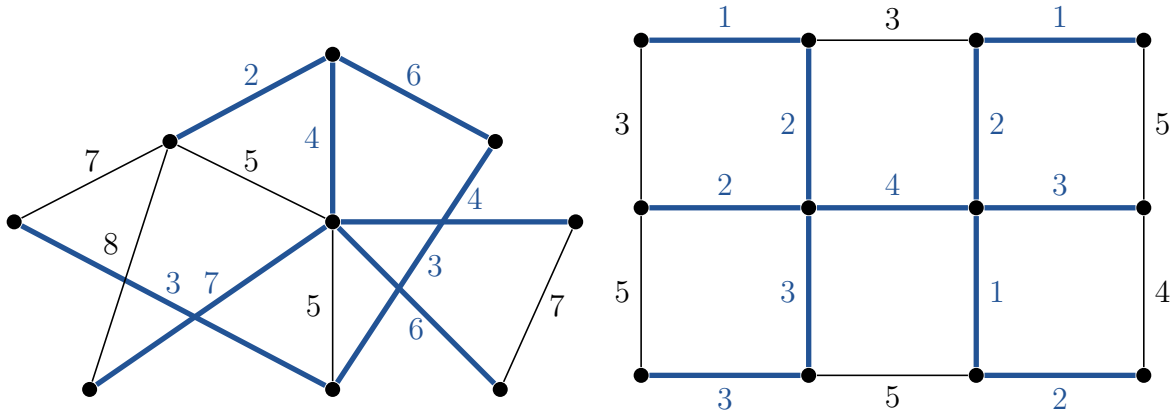
$$w(e) \leq w(e') \quad \Longleftrightarrow \quad c(e) \leq c(e').$$

Zeigen Sie: Dann ist T genau dann ein minimaler spannender Baum bezüglich der Gewichte $w(e)$, wenn T ein minimaler spannender Baum bezüglich der Gewichte $c(e)$ ist.[2]

Aufgabe 4.3 Betrachten Sie den gewichteten Graphen aus Abbildung 4.6.

[1] Diese Idee ist übrigens die Grundlage des Gradienten-Verfahrens zur Minimierung von Funktionen $f : \mathbb{R}^n \to \mathbb{R}$ ohne Nebenbedingungen.

[2] Dieses Resultat erlaubt es die Gewichte in Bäumen zu vereinfachen, z.B. zu skalieren oder zu verschieben, ohne dass sich die minimalen spannenden Bäume ändern.



(a) (b)

Abbildung 4.5: Minimale spannende Bäume?

(a) Bestimmen Sie einen minimalen spannenden Baum mit dem Algorithmus von Prim.

(b) Bestimmen Sie einen minimalen spannenden Baum mit dem Algorithmus von Kruskal.

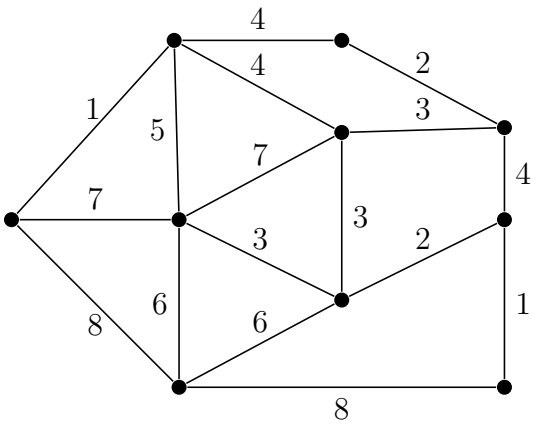


Abbildung 4.6: Wie sieht ein minimaler spannender Baum aus?

Aufgabe 4.4 Beweisen oder widerlegen (mit einem Gegenbeispiel) Sie die folgenden beiden Aussagen:

(a) Es sei $G = (V, E)$ ein ungerichteter, zusammenhängender und gewichteter Graph mit eindeutigen Kantengewichten, d.h. $w(e) \neq w(e')$ für $e \neq e'$. Dann besitzt G einen eindeutigen minimalen spannenden Baum.

(b) Es sei $G = (V, E)$ ein ungerichteter, zusammenhängender und gewichteter Graph mit einem eindeutigen minimalen spannenden Baum. Dann hat G eindeutige Kantengewichte, d.h. es gilt $w(e) \neq w(e')$ für $e \neq e'$.

Aufgabe 4.5 Es sei G der Graph aus Abbildung 4.7 mit den Kantengewichten

$$1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6.$$

(a) Wie können Sie die Kantengewichte verteilen, so dass G einen eindeutigen minimalen spannenden Baum hat?

(b) Wie können Sie die Kantengewichte verteilen, so dass G mehr als einen minimalen spannenden Baum hat?

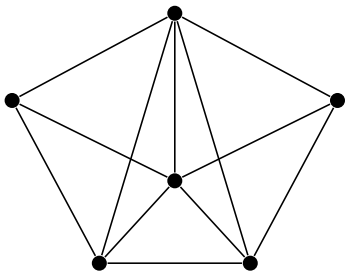


Abbildung 4.7: Wann hat G einen eindeutigen minimalen spannenden Baum?

5 Kürzeste Wege

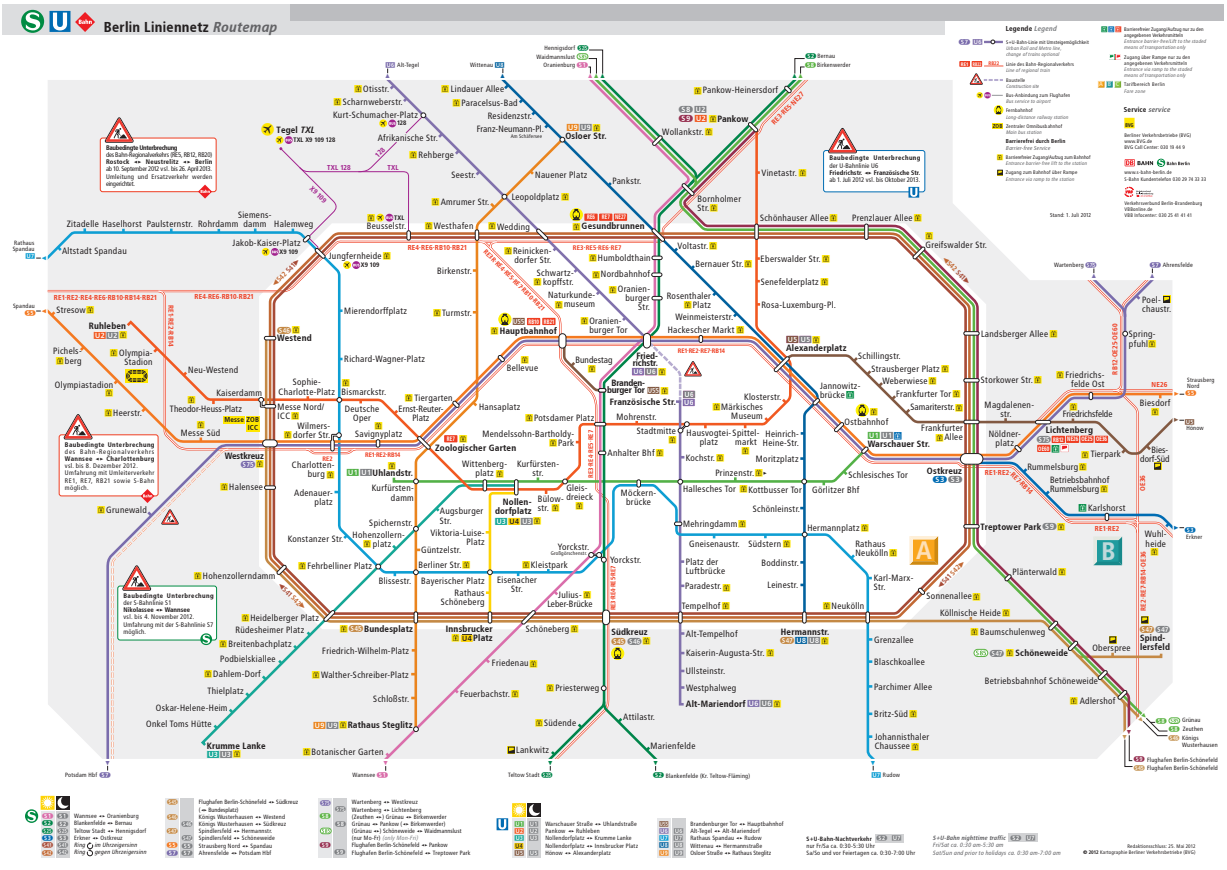


Abbildung 5.1: Ein Ausschnitt des Berliner U-Bahnnetzes (www.bvg.de)

Im letzten Kapitel haben wir in ungerichteten gewichteten Graphen zusammenhängende Teilgraphen mit minimalem Gewicht bestimmt. Diese spannenden Bäume hatten zwar insgesamt ein möglichst kleines Gewicht, aber die enthaltenen Verbindungen zwischen zwei Knoten waren nicht unbedingt die kürzesten. Suchen wir also kürzeste Verbindungen zwischen zwei Knoten, wie in den folgenden Beispielen, so helfen uns diese Verfahren nicht weiter.

Beispiel 5.1 (Routenplanung) Erinnern wir uns noch einmal an unser Problem einen möglichst kurzen Weg vom Würzburger Bahnhof zum Mathematischen Institut zu finden.

Diese Problem können wir als einen Graphen modellieren, indem wir die Straßen als Kanten betrachten und an Kreuzungen Knoten einfügen. Da wir Verkehrsregeln und Einbahnstraßen beachten müssen, erhalten wir einen gerichteten Graphen, indem wir einen möglichst kurzen Weg zwischen zwei Knoten suchen. ◊

Beispiel 5.2 (U-Bahn) Ein ähnliches Problem, bei dem man den zugrundeliegenden Graphen sogar immer zu Gesicht bekommt, stellt sich, wenn man im Urlaub in eine größere Stadt reist: Man kommt mit dem Zug am Hauptbahnhof an und muss nun erstmal einen Weg mit der U-Bahn zum Hotel finden. Und da man schwer mit Gepäck beladen ist, soll es auch hier nicht irgendein Weg sein, sondern ein möglichst kurzer. Da U-Bahnen im allgemeinen in beide Richtungen fahren, betrachten wir hier also einen ungerichteten Graphen, vergleiche Abbildung 5.1. ◊

An dieser Stelle sollten wir kurz innehalten. Was bedeutet möglichst kurz eigentlich? Soll der Weg streckenmäßig möglichst kurz sein? Oder in kürzester Zeit zurück gelegt werden? Oder soll er aus möglichst wenigen Teilstrecken bestehen, d.h. mit möglichst wenig Umsteigen auskommen? Je nach Kontext können verschiedene Definitionen von „möglichst kurz“ sinnvoll sein. Hat man sich für eine Definition entschieden, so gewichtet man die Kanten im betrachteten Graphen entsprechend. Man sucht also einen Weg, bei dem das Gesamtgewicht möglichst klein ist. Aber wie findet man den nun?

Definition 5.3 *Es sei $G = (V, E)$ ein gerichteter oder ungerichteter gewichteter Graph mit Gewichten $w(e)$ für alle $e \in E$ und $v, s \in V$. Ein Weg P mit Startknoten s und Endknoten v heißt* kürzester Weg von s nach v*, wenn sein Gewicht (seine Länge) $w(P) := \sum_{e \in P} w(e)$ unter allen Wegen von s nach v minimal ist.*

Mit $dist(s, v) := w(P)$ bezeichnen wir den Abstand *von s und v, also die Länge eines kürzesten Weges P. Gibt es keinen kürzesten Weg, so setzt man $dist(s, v) = -\infty$.*

Ein Teilgraph $G' = (V, E')$ von G heißt Baum kürzester Wege bezüglich (der Wurzel) s*, wenn G' ein spannender Baum mit Wurzel s ist und für alle $v \in V$ einen kürzesten Weg von s nach v enthält.*

Erinnern wir uns noch einmal an die Breitensuche. Diese hatte die schöne Eigenschaft gehabt, dass sie in ungewichteten Graphen vom Startknoten s zu allen Knoten v kürzeste Wege im Sinne von Kantenzahl gefunden hat. Die Kantenzahl eines solchen kürzesten Weges hatten wir ebenfalls mit $\text{dist}(s, v)$ bezeichnet. Dies entspricht kürzesten Wegen in einem gewichteten Graphen, wenn wir alle Kanten mit dem Gewicht 1 versehen.

Bevor wir Eigenschaften kürzester Wege betrachten, sollten wir uns kurz Gedanken über die Existenz kürzester Wege machen. Sind alle Kantengewichte nichtnegativ, so sind kürzeste Wege, wenn sie existieren, immer einfach. Würden sie einen Kreis enthalten, so könnte man diesen weglassen ohne den Weg zu verlängern, da der Kreis eine nichtnegative Länge hätte. Da es zwischen zwei Knoten immer nur endlich viele einfache Wege gibt, gibt es in diesem Fall zwischen zwei Knoten, zwischen denen es einen Weg gibt, immer auch einen kürzesten Weg.

Gibt es hingegen auch negative Kantengewichte, so ist die Existenz kürzester Wege nicht mehr gesichert. Genauer tritt ein Problem auf, wenn es Kreise negativer Länge gibt, vergleiche Abbildung 5.2. Dann gibt es zwischen zwei Knoten s, v, die über einen Weg verbunden werden können, der einen Kreis negativer Länge enthält, keinen kürzesten Weg, da jeder weitere Durchlauf dieses Kreises die Länge des Weges verringert.

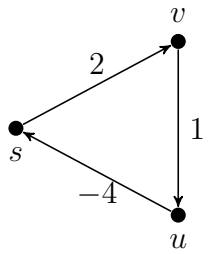


Abbildung 5.2: Gewichteter Graph mit Kreis negativer Länge

Kürzeste Wege haben die folgende nützliche Eigenschaft, die wir in diesem Kapitel häufig verwenden werden.

Lemma 5.4 *Es sei $G = (V, E)$ ein gerichteter oder ungerichteter gewichteter Graph mit Gewichten $w(e)$ für alle $e \in E$ und $v_0, v_l \in V$. Ist $P = (v_0, e_1, v_1, \ldots, e_l, v_l)$ ein kürzester Weg mit Startknoten v_0 und Endknoten v_l, so ist für alle $0 \leq i < j \leq l$ der Teilweg $P_{ij} = (v_i, e_i, v_{i+1}, \ldots, e_j, v_j)$ ein kürzester Weg mit Startknoten v_i und Endknoten v_j.*

Beweis Angenommen, es gäbe zwei Knoten v_i, v_j auf dem Weg P, zwischen denen ein kürzerer Weg P'_{ij} als P_{ij} existiert. Dann wäre der Weg P', der aus P entsteht, wenn man den Teilweg P_{ij} durch P'_{ij} ersetzt, ein Weg in G mit Startknoten v_0 und Endknoten v_l, der kürzer als P ist, ein Widerspruch. □

Hat man einen Wurzelbaum gefunden, von dem man glaubt, dass er ein Baum kürzester Wege ist, so stellt sich die Frage, wie man das möglichst einfach überprüfen kann. Hier hilft das folgende Resultat.

Satz 5.5 *Es sei $G = (V, E)$ ein gerichteter oder ungerichteter, zusammenhängender und gewichteter Graph mit Kantengewichten $w(e)$ für alle $e \in E$ und $s \in V$. Ein Teilgraph $T = (V, E')$ von G ist genau dann ein Baum kürzester Wege bezüglich s, wenn T ein spannender Baum mit Wurzel s ist und für alle Kanten $e = (v, u) \in E \setminus E'$ gilt*

$$d(u) \leq d(v) + w(e),$$

wobei für alle $v \in V$ der Wert $d(v)$ die Länge des eindeutigen (!) Weges in T von s nach v bezeichnet.

Beweis $\Longrightarrow$ Es sei T ein Baum kürzester Wege bezüglich s. Dann ist T offensichtlich ein spannender Baum. Angenommen, es gibt eine Kante $e = (v, u) \in E \setminus E'$ mit $d(u) >$

$d(v) + w(e)$. Dann würde aber folgen, dass es in G einen Weg von s nach u mit der Länge $d(v)+w(e) < d(u)$ gibt, nämlich den Weg von s nach v im Teilgraphen T verlängert um die Kante e. Folglich enthielte T keinen kürzesten Weg von s nach u, ein Widerspruch dazu, dass T ein Baum kürzester Wege bezüglich s ist.

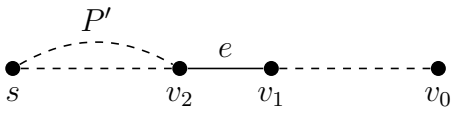


Abbildung 5.3: Illustration des Weges P im Beweis von Satz 5.5

$\Longleftarrow$ Nun sei umgekehrt T ein spannender Baum mit Wurzel s und

$$d(u) \leq d(v) + w(e)$$

für alle Kanten $e = (v, u) \in E \setminus E'$. Für alle $v \in V$ bezeichnet $\text{dist}(s, v)$ die Länge des tatsächlich kürzesten Weges in G von s nach v. Dann gilt nach Definition von $d(v)$ für alle v die Ungleichung $\text{dist}(s, v) \leq d(v)$ und es gilt $\text{dist}(s, s) = d(s) = 0$. Angenommen, es gibt einen Knoten $v_0 \in V$, für den ein kürzester Weg P von s nach v_0 existiert mit $\text{dist}(s, v_0) < d(v_0)$. Sei nun v_1 der erste Knoten auf dem Weg mit der Eigenschaft $\text{dist}(s, v_1) < d(v_1)$. Dann gilt $v_1 \neq s$, d.h. v_1 besitzt auf dem Pfad P einen Vorgänger v_2 mit $\text{dist}(s, v_2) = d(v_2)$. Folglich enthält T einen kürzesten Weg P' von s nach v_2. Wäre $e = (v_2, v_1) \in E'$, so enthielte T auch einen kürzesten Weg von s nach v_1 und es würde $d(v_1) = \text{dist}(s, v_1)$ folgen. Daher muss $e \in E \setminus E'$ gelten. Hieraus folgt aber

$$\text{dist}(s, v_1) < d(v_1) \leq d(v_2) + w(e) = \text{dist}(s, v_2) + w(e) = \text{dist}(s, v_1),$$

offensichtlich ein Widerspruch. □

Damit haben wir aber immer noch nicht beantwortet, wie man kürzeste Wege findet. Dies werden wir in den folgenden Abschnitten untersuchen. Im Gegensatz zum letzten Kapitel werden wir in diesem nur gerichtete Graphen betrachten. Die meisten Ideen lassen sich jedoch auch mit geeigneten Modifikationen auf ungerichtete Graphen übertragen.

5.1 Grundbausteine

Zur Erinnerung: Bei der Breitensuche setze man die Abstände am Anfang auf $d(s) = 0$ für den Startknoten s und $d(v) = \infty$ für alle anderen Knoten $v \neq s$. Die Vorgänger wurden auf $\pi(v) = 0$ für alle Knoten $v \in V$ gesetzt. Dann begann man im Startknoten s und setze bei allen Nachbarn v den Abstand auf $d(v) = 1$ und den Vorgänger auf $\pi(v) = s$. In der Reihenfolge ihres Auffindens untersuchte man anschließend alle anderen Knoten und korrigierte bei deren Nachbarn, falls nötig, den Abstand und den Vorgänger.

Wir werden das Verfahren nun so abändern, dass es in gerichteten Graphen mit beliebigen Gewichten funktioniert. Dazu betrachten wir zunächst die zwei zentralen Bestandteile des Verfahrens:

Ein wichtiger Bestandteil ist die Initialisierung des Verfahrens mit den bekannten Daten, vergleiche Algorithmus 8.

Algorithmus 8 Kürzeste Wege: Initialisierung

1 Es sei $G = (V, E)$ ein gerichteter Graph mit Kantengewichten $w(e)$ für alle $e \in E$ und einer Wurzel $s \in V$.
2 Setze die Abstände auf $d(s) = 0$ und $d(v) = \infty$ für alle $v \in V \setminus \{s\}$.
3 Setze die Vorgänger auf $\pi(v) = 0$ für alle $v \in V$.

Die zweite zentrale Zutat ist die Regel für die Aufdatierung von Abständen und Vorgängern, die wir ein kleines bisschen abwandeln wollen. Bei der Breitensuche wurde für jede Kante $e = (v, u)$ mit bekanntem Anfangsknoten v geprüft, ob zu dem Endknoten u schon ein Weg existierte oder nicht. Wir werden zusätzlich noch überprüfen, ob der schon bekannte Weg, so er existiert, länger ist, als einer über die betrachtete Kante, vergleiche Algorithmus 9.

Algorithmus 9 Kürzeste Wege: Test(e)

1 Es sei $G = (V, E)$ ein gerichteter Graph mit Kantengewichten $w(e)$ für alle $e \in E$, einer Wurzel $s \in V$, Abständen $d(v)$ und Vorgängern $\pi(v)$ für alle $v \in V$ und $e = (v, u) \in E$.
2 **if** $d(u) > d(v) + w(e)$ **then**
3 Setze $d(u) = d(v) + w(e)$.
4 Setze $\pi(u) = v$.
5 **end if**

Bei diesem Test verwenden wir die folgenden Rechenregeln für ∞:

$$\infty + w(e) = \infty \quad \text{und} \quad \infty = \infty.$$

Um ein Verfahren zur Bestimmung kürzester Wege zu entwickeln, müssen wir noch festlegen, in welcher Reihenfolge die Kanten $e = (v, u)$ mit Algorithmus 9 getestet werden sollen. Doch auch ohne hierfür eine Regel festzulegen, können wir schon einige Aussagen treffen. Dafür verwenden wir wieder den bei der Breitensuche eingeführten Vorgängergraphen $G_\pi = (V_\pi, E_\pi)$

Satz 5.6 *Es sei $G = (V, E)$ ein gerichteter Graph mit Kantengewichten $w(e)$ für alle $e \in E$ und einer Wurzel $s \in V$. Initialisiert man die Abstände $d(v)$ und die Vorgänger $\pi(v)$ mit Algorithmus 8 und führt eine beliebige Anzahl von Testschritten (Algorithmus 9) aus, so gelten:*

(a) $d(v) \geq dist(s, v)$ für alle $v \in V$.

(b) Jeder Kreis im Vorgängergraphen $G_\pi = (V_\pi, E_\pi)$ hat negative Länge.

(c) Falls G keinen Kreis negativer Länge enthält, so ist G_π ein Baum mit Wurzel s, der für jedes $v \in V_\pi \setminus \{s\}$ einen Weg der Länge höchstens $d(v)$ enthält, der sich durch Rückverfolgen der Vorgänger $\pi(v)$ konstruieren lässt.

Beweis Zunächst sollten wir zeigen, dass der Vorgängergraph $G_\pi := (V_\pi, E_\pi)$ mit

$$\begin{aligned} V_\pi &:= \{v \in V \mid \pi(v) \neq 0\} \cup \{s\}, \\ E_\pi &:= \{(\pi(v), v) \mid v \in V_\pi \setminus \{s\}\} \end{aligned}$$

wohldefiniert ist. Dazu müssen wir zeigen, dass für jede Kante $e = (\pi(v), v) \in E_\pi$ auch der Anfangsknoten $\pi(v) \in V_\pi$ ist. Damit $\pi(v)$ der Vorgänger werden kann, muss aber $d(\pi(v)) < \infty$ gelten. Daraus folgt, dass entweder $\pi(v) = s$ gilt, oder $\pi(v)$ selbst einen Vorgänger besitzt. In beiden Fällen folgt $\pi(v) \in V_\pi$.

Wir zeigen die Behauptungen (a) und (b) durch Induktion über die Anzahl der Testschritte: Vor dem ersten Testschritt gilt $V_\pi = \{s\}$, $E_\pi = \emptyset$, $\pi(v) = 0$ für alle $v \in V$, $d(s) = 0$ und $d(v) = \infty$ für alle $v \in V \setminus \{s\}$. Daher sind die Aussagen (a) und (b) wahr.

Seien sie nun für eine gewisse Anzahl von Testschritten wahr und es werde ein weiterer mit der Kante $e = (v, u)$ durchgeführt. Gilt $d(u) \leq d(v) + w(e)$, so ändert sich nichts und wir müssen nichts zeigen. Sei daher nun $d(u) > d(v) + w(e)$. Dann wird $d(u) = d(v) + w(e)$ und $\pi(u) = v$ gesetzt. Außerdem wird die Kante (v, u) zu G_π hinzugefügt und gegebenenfalls die Kante vom alten Vorgänger von u nach u entfernt.

(a) Mit der Induktionsvoraussetzung folgt

$$d(u) = d(v) + w(e) \geq \text{dist}(s, v) + w(e) \geq \text{dist}(s, u),$$

da $\text{dist}(s, v) + w(e)$ die Länge eines (nicht notwendig kürzesten) Weges von s nach u ist.

(b) Sei $P = (v_0 = v, e_1 = e, v_1 = u, \ldots, e_l, v_l = v)$ ein Kreis in G_π, der durch die neue Kante (v, u) geschlossen wurde. Alle Kreise in G_π, die diese Kante nicht enthalten, haben nach Induktionsvoraussetzung negative Länge. Dann galt vor dem gerade betrachteten Testschritt $d(u) > d(v) + w(e)$ und nach Konstruktion des Vorgängergraphen für alle anderen Kanten auf dem Kreis $d(v_i) = d(v_{i-1}) + w(e_i) < \infty$, $i = 2, \ldots, l$. Folglich ist

$$\begin{aligned} w(P) &= w(e) + \sum_{i=2}^{l} w(e_i) = w(e) + \sum_{i=2}^{l} (d(v_i) - d(v_{i-1})) \\ &= w(e) + d(v_l) - d(v_1) = w(e) + d(v) - d(u) < 0. \end{aligned}$$

(c) Sei $v_0 \in V_\pi \setminus \{s\}$ beliebig. Dann betrachten wir die Folge $v_0, v_1, v_2, \ldots$ mit $v_{i+1} = \pi(v_i)$ für alle $i = 0, 1, \ldots$. Wenn diese Folge irgendwann mit $v_l = s$ abbricht, so liefert sie uns in umgekehrter Reihenfolge einen Weg in G_π von s nach v_0. Bricht die Folge nicht ab, so müssen sich irgendwann Knoten wiederholen, da V_π nur endlich viele Elemente hat. Dies bedeutet aber, dass G_π mindestens einen Kreis enthält, der nach (b) negative Länge haben muss. Da G aber nach Voraussetzung keine Kreise negativer Länge enthält, kann der Teilgraph G_π keine Kreise negativer Länge enthalten. Folglich

bricht die Folge $v_0, v_1, v_2, \ldots$ irgendwann in einem Knoten v_l ab, der keinen Vorgänger hat. Da $v_l = \pi(v_{l-1})$ ist, ist $d(v_l) < \infty$. Folglich gilt entweder $v_l = s$ oder v_l hat einen Vorgänger $\pi(v_l)$. In diesem Fall würde die Folge aber nicht abbrechen, also muss $v_l = s$ gelten.

Folglich ist s eine Wurzel in G_π. Da nach Konstruktion außerdem $g^-(s) = 0$ und $g^-(v) = 1$ für alle $v \in V_\pi \setminus \{s\}$ gilt, ist G_π nach Satz 3.16 ein Baum mit Wurzel s.

□

5.2 Algorithmus von Dijkstra

Eine Möglichkeit unsere Überlegungen aus dem letzen Abschnitt zu nutzen, ist der Algorithmus von Dijkstra, vergleiche Algorithmus 10, der in gerichteten gewichteten Graphen kürzeste Wege von einem Startknoten zu allen anderen Knoten bestimmt. Sucht man nur einen kürzesten Weg zu einem bestimmten anderen Knoten, so kann man das Verfahren abbrechen, sobald dieser Knoten aus der Wartemenge S entfernt wurde.

Algorithmus 10 Algorithmus von Dijkstra

Input: Ein gerichteter gewichteter Graph $G = (V, E)$ mit Wurzel s, $n = |V|$ Knoten und nichtnegativen Kantengewichten $w(e) \geq 0$ für alle $e \in E$.

1 Setze die Wartemenge $S = V$, die Abstände auf $d(s) = 0$, $d(v) = \infty$ für alle $v \in V \setminus \{s\}$ und die Vorgänger auf $\pi(v) = 0$ für alle $v \in V$.
2 **for** $k = 1$ **to** n **do**
3 Entferne ein Element v mit $d(v) = \min\{d(u) \mid u \in S\}$ aus der Wartemenge S.
4 **for all** $u \in S$ mit $e = (v, u) \in E$ **do**
5 **if** $d(u) > d(v) + w(e)$ **then**
6 Setze $d(u) = d(v) + w(e)$ und $\pi(u) = v$.
7 **end if**
8 **end for**
9 **end for**

Output: Die Vorgänger $\pi(v)$ und die Abstände $d(v)$ für alle $v \in V$.

Bevor wir die Korrektheit des Verfahrens beweisen, wollen wir an einem kleinen Beispiel nachvollziehen, wie der Algorithmus funktioniert. Dazu betrachten wir den Graphen in Abbildung 5.4 mit der Wurzel 1 und notieren alle relevanten Daten in der Tabelle 5.1.
Nachdem wir an einem Beispiel nachvollzogen haben, dass der Algorithmus funktioniert, wollen wir es noch allgemein beweisen.

Satz 5.7 *Es sei $G = (V, E)$ ein gerichteter gewichteter Graph mit Wurzel s, $n = |V|$ Knoten und nichtnegativen Kantengewichten $w(e) \geq 0$ für alle $e \in E$. Dann ist der Algorithmus 10 wohldefiniert. Für alle $v \in V$ ist $d(v) = dist(s, v)$.*

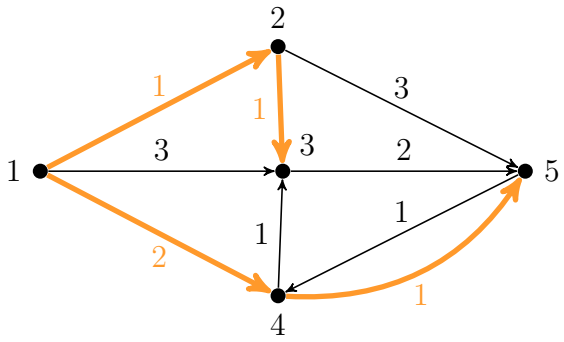


Abbildung 5.4: Illustration des Algorithmus von Dijkstra

k	v	S	d	π
0	-	$\{1,2,3,4,5\}$	$(0,\infty,\infty,\infty,\infty,\infty)$	$(0,0,0,0,0)$
1	1	$\{2,3,4,5\}$	$(0,1,3,2,\infty)$	$(0,1,1,1,0)$
2	2	$\{3,4,5\}$	$(0,1,2,2,4)$	$(0,1,2,1,2)$
3	4	$\{3,5\}$	$(0,1,2,2,3)$	$(0,1,2,1,4)$
4	3	$\{5\}$	$(0,1,2,2,3)$	$(0,1,2,1,4)$
5	5	$\emptyset$	$(0,1,2,2,3)$	$(0,1,2,1,4)$

Tabelle 5.1: Illustration des Algorithmus von Dijkstra

Beweis Zunächst müssen wir zeigen, dass der Algorithmus wohldefiniert ist, d.h. dass die Wartemenge S nicht vorzeitig leer ist und dass die Berechnung der $d(v)$ wohldefiniert ist. Da die Wartemenge S am Anfang genau n Elemente enthält und in jeder Iteration ein Element entfernt und keines neu hinzugefügt wird, ist die Wartemenge erst nach n Iterationen leer. Solange die Wartemenge S nicht leer ist, enthält sie in jeder Iteration mindestens einen Knoten v mit $d(v) < \infty$. In der ersten Iteration folgt dies aus der Definition von d. In allen weiteren Iterationen würde sonst folgen, dass es keine Kante zwischen den schon entfernten Knoten und den Knoten in S gibt. Dies wäre ein Widerspruch dazu, dass s eine Wurzel ist. Daher ist die Berechnung der $d(v)$ wohldefiniert und nach Abbruch des Algorithmus gilt $d(v) < \infty$ für alle Knoten $v \in V$.

Es bleibt zu zeigen, dass für alle Knoten $v \in V$ gilt $d(v) = \text{dist}(s, v)$. Nach Satz 5.6 (a) gilt für alle $v \in V$ zumindest $d(v) \geq \text{dist}(s, v)$. Wir müssen also nur noch $d(v) \leq \text{dist}(s, v)$ zeigen.

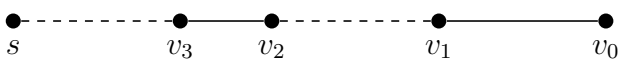


Abbildung 5.5: Illustration des Weges P im Beweis von Satz 5.7

Angenommen, es gäbe mindestens einen Knoten $v \neq s$ mit $d(v) > \text{dist}(s, v)$. Dann betrachten wir den ersten Knoten v_0 mit dieser Eigenschaft, der aus der Wartemenge entfernt wurde. Sei P ein kürzester Weg von s nach v_0, d.h. ein Weg mit Länge $\text{dist}(s, v_0)$,

vergleiche auch Abbildung 5.5. Dann muss der Vorgänger v_1 von v_0 auf diesem Weg nach v_0 aus der Wartemenge entfernt worden sein. Wäre er vor v_0 entfernt worden so wäre nach Voraussetzung $d(v_1) = \text{dist}(s, v_1)$ und daher

$$d(v_0) \leq d(v_1) + w((v_1, v_0)) = \text{dist}(s, v_1) + w((v_1, v_0)) = \text{dist}(s, v_0),$$

ein Widerspruch zur Annahme $d(v_0) > \text{dist}(s, v_0)$. Hierbei haben wir bei der letzten Gleichung verwendet, dass der Teilweg von P vom Knoten s zum Knoten v_1 ein kürzester Weg zwischen diesen beiden Knoten ist. Es gibt also auf dem Weg P mindestens einen Knoten, der nach v_0 aus der Wartemenge entfernt wurde. Sei v_2 der erste Knoten auf dem Weg P, der erst nach v_0 aus der Wartemenge entfernt wurde. Da der Startknoten des Weges P der Knoten s ist und dieser immer als allererster aus der Wartemenge entfernt wird, kann nicht $v_2 = s$ gelten, d.h. der Knoten v_2 hat auf dem Weg P einen Vorgänger v_3, der vor v_0 aus der Warteschlange entfernt wurde und für den daher $d(v_3) = \text{dist}(s, v_3)$ gilt. Nachdem v_3 aus der Wartemenge entfernt wurde, gilt aber

$$d(v_2) \leq d(v_3) + w((v_3, v_2)) = \text{dist}(s, v_3) + w((v_3, v_2)) = \text{dist}(s, v_2) \leq \text{dist}(s, v_0) < d(v_0).$$

Für die letzte Gleichung haben wir wieder verwendet, dass Teilwege von kürzesten Wegen selbst kürzeste Wege sind. Bei der Ungleichung danach geht ein, dass die Kantengewichte nichtnegativ sind und daher Knoten, die später auf einem kürzesten Weg liegen, einen größeren Abstand zum Startknoten haben. Dies bedeutet aber, dass v_2 vor v_0 aus der Wartemengen hätte entfernt werden müssen, ein Widerspruch zur Wahl von v_2. □

Tatsächlich bestimmt der Algorithmus von Dijkstra nicht nur kürzeste Wege vom Startknoten zu allen anderen Knoten sondern, ähnlich wie die Breitensuche, einen Baum kürzester Wege.

Korollar 5.8 *Es sei $G = (V, E)$ ein gerichteter gewichteter Graph mit Wurzel s, $n = |V|$ Knoten und nichtnegativen Kantengewichten $w(e) \geq 0$ für alle $e \in E$. Dann erzeugt Algorithmus 10 einen spannenden Baum kürzester Wege mit Wurzel s.*

Beweis Es sei $G_\pi = (V_\pi, E_\pi)$ der von Algorithmus 10 erzeugte Vorgängergraph. Dieser ist nach Satz 5.6 ein Baum mit Wurzel s, der für alle $v \in V_\pi \setminus \{s\}$ einen Weg von s nach v der Länge $d(v)$ enthält. Nach Satz 5.7 gilt außerdem $V_\pi = V$ und $d(v) = \text{dist}(s, v)$ für alle $v \in V$. Damit folgt die Behauptung. □

5.3 Algorithmus von Bellman-Ford

Treten in einem gewichteten Graphen auch negative Kantengewichte auf, so funktioniert der Algorithmus von Dijkstra nicht mehr korrekt. Betrachten wir den Graphen aus Abbildung 5.6, so bestimmt der Algorithmus von Dijkstra mit Startknoten $s = 1$ die Abstände $d = (0, 2, 1)$, richtig wäre aber offensichtlich $d = (0, 2, 0)$.

Andererseits gibt es Anwendungen, in denen negative Kantengewichte auftreten.

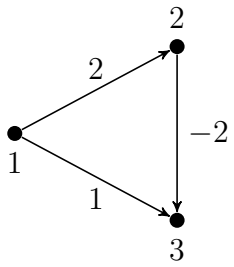


Abbildung 5.6: Beispiel für fehlerhafte Ergebnisse von Algorithmus 10

Beispiel 5.9 (Wechselkurse) Angenommen, wir haben Euro und wollen bei der Bank möglichst günstig 100 US-Dollar kaufen. Wir können nun direkt Dollar kaufen und müssen der Bank für jeden Dollar, den wir kaufen wollen, einen gewissen Preis bezahlen. Wir könnten stattdessen aber auch erst Britische Pfund für Euros kaufen, mit den Pfund dann Schweizer Franken kaufen und diese schließlich verwenden um die Dollar zu kaufen. Um eine Einheit der neuen Währung zu bekommen, müssen wir immer einen gewissen Preis in der alten Währung bezahlen. Die Frage ist nun, in welcher Reihenfolge wir am besten wechseln um die Kosten möglichst gering zu halten?

Dieses Problem können wir als gerichteten Graphen modellieren, indem wir jeder Währung einen Knoten zuordnen und die Kanten mit dem Preis $w(e)$ gewichten, den wir in der Startwährung bezahlen müssen um eine Einheit der Zielwährung zu bekommen. Die Gesamtkosten eines Tauschvorganges entlang eines Weges P sind dann durch $w(P) = \prod_{e \in P} w(e)$ gegeben und wir suchen einen möglichst günstigen Weg vom Euro-Knoten zum Dollar-Knoten. Dies lässt sich in ein kürzeste Wege-Problem umwandeln, wenn wir die Tauschkosten logarithmieren, denn es gilt

$$\log w(P) = \log \prod_{e \in P} w(e) = \sum_{e \in P} \log w(e).$$

Abhängig davon, ob $w(e) \geq 1$ ist oder $1 > w(e) > 0$, ist $\log w(e) \geq 0$ oder $\log w(e) < 0$. $\Diamond$

In solchen Fällen verwendet man den Algorithmus von Bellman-Ford, vergleiche Algorithmus 11. Dieser kann nicht nur negative Kantengewichte korrekt verarbeiten, sondern erkennt auch, wenn Kreise negativer Länge existieren.
Bevor wir zeigen, dass der Algorithmus von Bellman-Ford auch in Graphen mit möglicherweise negativen Kantengewichten kürzeste Wege bestimmt, wollen wir uns an dem kleinen Beispiel vom Anfang des Abschnitts anschauen, was der Algorithmus tut, vergleiche Abbildung 5.7 und Tabelle 5.2. Der hier dargestelle Iterationsverlauf hängt natürlich von der Reihenfolge ab, in der man die Kanten betrachtet.
Nun wollen wir aber noch nachweisen, dass der Algorithmus nicht nur in unserem Minibeispiel korrekt funktioniert, sondern immer.

Satz 5.10 *Es sei $G = (V, E)$ ein gerichteter gewichteter Graph mit Wurzel s, $n = |V|$ Knoten und Kantengewichten w_e für alle $e \in E$.*

Algorithmus 11 Algorithmus von Bellman-Ford

Input: Ein gerichteter, zusammenhängender und gewichteter Graph $G = (V, E)$ mit Wurzel s, $n = |V|$ Knoten und Kantengewichten $w(e)$ für alle $e \in E$.

```
1  Setze die Abstände auf d(s) = 0, d(v) = ∞ für alle v ∈ V \ {s} und die Vorgänger auf
   π(v) = 0 für alle v ∈ V.
2  for k = 1 to n − 1 do
3    for all e = (u, v) ∈ E do
4      if d(v) > d(u) + w(e) then
5        Setze d(v) = d(u) + w(e) und π(v) = u.
6      end if
7    end for
8  end for
9  for all e = (u, v) ∈ E do
10   if d(v) > d(u) + w(e) then
11     print Es gibt einen Kreis negativer Länge
12   end if
13 end for
```

Output: Die Vorgänger $\pi(v)$ und die Abstände $d(v)$ für alle $v \in V$.

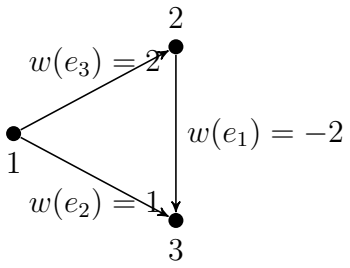


Abbildung 5.7: Illustration des Algorithmus 11 mit Wurzel 1

k	d	π
0	$(0, \infty, \infty)$	$(0, 0, 0)$
1	$(0, 2, 1)$	$(0, 1, 1)$
2	$(0, 2, 0)$	$(0, 1, 2)$

Tabelle 5.2: Illustration des Algorithmus 11 mit Wurzel 1

(a) Dann ist der Algorithmus 11 wohldefiniert.

(b) Enthält G keinen Kreis negativer Länge, so gilt $d(v) = dist(s, v)$ für alle $v \in V$.

(c) Enthält G einen Kreis negativer Länge, so wird dies am Ende des Algorithmus korrekt festgestellt.

Beweis Wir zeigen zunächst per Induktion die Hilfsaussage, dass am Ende jeder Iteration $k = 1, \ldots, n-1$ für alle $v \in V$ gilt

$$d(v) \leq \min\{w(P) \mid P \text{ ist Weg von } s \text{ nach } v \text{ mit höchstens } k \text{ Kanten }\}.$$

Hierbei setzen wir $\min \emptyset = \infty$. Für $k = 0$, d.h. zu Beginn des Verfahrens ist die Aussage wahr, denn es gilt $d(s) = 0$ und $d(v) = \infty$ für alle $v \neq s$. Betrachten wir nun eine Iteration k und gehen davon aus, dass die Aussage für alle vorherigen Iterationen wahr ist. Dann müssen wir nur

$$d(v) \leq \min\{w(P) \mid P \text{ ist Weg von } s \text{ nach } v \text{ mit genau } k \text{ Kanten }\}$$

für alle $v \neq s$ zeigen, denn $d(v)$ wird in der k-ten Iteration höchstens verkleinert und für Wege mit weniger als k Kanten wissen wir schon aus der Induktionsvoraussetzung, dass die Abschätzung mit dem alten $d(v)$ gilt. Sei also P ein Weg von s nach v mit genau k Kanten und u der Vorgänger von v auf diesem Weg, also der vorletzte besuchte Knoten. Dann ist der um die letzte Kante verkürzte Weg P' ein Weg von s nach u mit $k-1$ Kanten und mit der Induktionsvoraussetzung folgt, dass zu Beginn der k-ten Iteration gilt $d(u) \leq w(P')$. Daher ist am Ende der Iteration $d(v) \leq d(u) + w((u, v)) \leq w(P') + w((u, v)) = w(P)$.

(a) Mit der Konvention für das Rechnen mit ∞ ist der Vergleich der Abstände $d(v)$ auch im Fall $d(v) = \infty$ wohldefiniert. Daher ist der Algorithmus wohldefiniert und bricht nach endlich vielen Iterationen ab.

(b) Enthält G keine Kreise negativer Länge, so gibt es von s zu jedem Knoten v einen einfachen kürzesten Weg, denn das Entfernen von Kreisen aus einem kürzesten Weg verlängert diesen nach Voraussetzung nicht. Folglich hat ein kürzester Weg P immer höchstens $n-1$ Kanten und mit der obigen Hilfsaussage folgt $d(v) \leq w(P) = \text{dist}(s, v)$ für alle $v \in V$. Andererseits gilt nach Satz 5.6 (a) auch $d(v) \geq \text{dist}(s, v)$ für alle $v \in V$, es muss also Gleichheit gelten.

(c) Enthält G keinen Kreis negativer Länge, so ist der vom Algorithmus bestimmte Vorgängergraph $G_\pi = (V_\pi, E_\pi)$ nach Teil (b) und Satz 5.6 (c) ein spannender Baum kürzester Wege mit Wurzel s und es gilt $d(v) = \text{dist}(s, v)$ für alle $v \in V$. Daher gilt für alle Kanten $e = (u, v) \in E_\pi$ die Gleichung $d(v) = d(u) + w(e)$ und für alle Kanten $e = (u, v) \in E \setminus E_\pi$ nach Satz 5.5 die Ungleichung $d(v) \leq d(u) + w(e)$. Folglich gibt der Algorithmus keine Fehlermeldung zurück.

Gibt der Algorithmus umgekehrt keine Fehlermeldung zurück, so müssen wir zeigen, dass G keinen Kreis negativer Länge enthält. Gibt es keine Fehlermeldung, so gilt für

alle $e = (u, v) \in E$ am Ende des Algorithmus $d(v) \leq d(u) + w(e)$. Da s eine Wurzel ist, gibt es außerdem für alle $v \in V$ einen Weg von s nach v. Da es dann auch einen einfachen Weg und folglich auch einen Weg mit höchstens $n - 1$ Kanten gibt, folgt mit der obigen Hilfsaussage außerdem $d(v) < \infty$ für alle $v \in V$. Folglich gilt für alle Kreise $P = (v_0, e_1, v_1, \ldots, e_l, v_l = v_0)$ in G

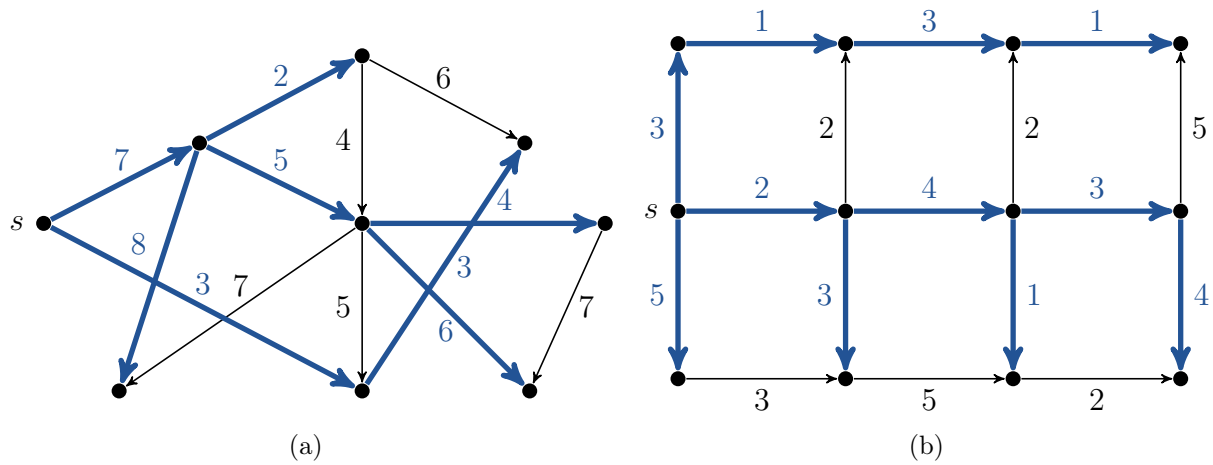
$$w(P) = \sum_{i=1}^{l} w(e_i) \geq \sum_{i=1}^{l} (d(v_i) - d(v_{i-1})) = d(v_l) - d(v_0) = 0,$$

d.h. G enthält keine Kreise negativer Länge.

□

5.4 Aufgaben

Aufgabe 5.1 Welche der spannenden Bäume in Abbildung 5.8 sind Bäume kürzester Wege mit Wurzel s? Wenn nicht, wie lassen Sie sich zu Bäumen kürzester Wege mit Wurzel s abändern?



(a) (b)

Abbildung 5.8: Bäume kürzeste Wege mit Wurzel s?

Aufgabe 5.2 Es sei $G = (V, E)$ ein gerichteter Graph mit Kantengewichten $w(e)$ für alle $e \in E$ und $P = (v_0 = s, e_1, v_1 \ldots, e_l, v_l = v)$ ein kürzester Weg von s nach v.
Beweisen oder widerlegen (Gegenbeispiel) Sie:

(a) Sei $\alpha > 0$. Dann definieren wir neue Kantengewichte $w_1(e) := \alpha \cdot w(e)$ für alle $e \in E$. Dann ist P auch ein kürzester Weg von s nach v bezüglich der Gewichte w_1.

(a) Sei $\beta > 0$. Dann definieren wir neue Kantengewichte $w_2(e) := w(e) + \beta$ für alle $e \in E$. Dann ist P auch ein kürzester Weg von s nach v bezüglich der Gewichte w_2.

Aufgabe 5.3 Es sei $G = (V, E)$ ein gerichteter stark zusammenhängender Baum mit Wurzel s und nichtnegativen Kantengewichten $w(e) \geq 0$ für alle $e \in E$. Weiter sei $T = (V, E')$ ein minimaler spannender Baum mit Wurzel s und $G_\pi = (V, E_\pi)$ ein Baum kürzester Wege mit Wurzel s.

(a) Zeigen Sie, dass dann $w(G_\pi) \geq w(T)$ gilt.

(b) Zeigen Sie, dass umgekehrt $w(G_\pi) \leq (|V| - 1)w(T)$ gilt.

Aufgabe 5.4 Betrachten Sie die gerichteten gewichteten Graphen aus Abbildung 5.9.

(a) Versuchen Sie Bäume kürzester Wege mit Wurzel s mit dem Algorithmus von Dijkstra zu bestimmen.

(b) Bestimmen Sie Bäume kürzester Wege mit Wurzel s mit dem Algorithmus von Bellman-Ford.

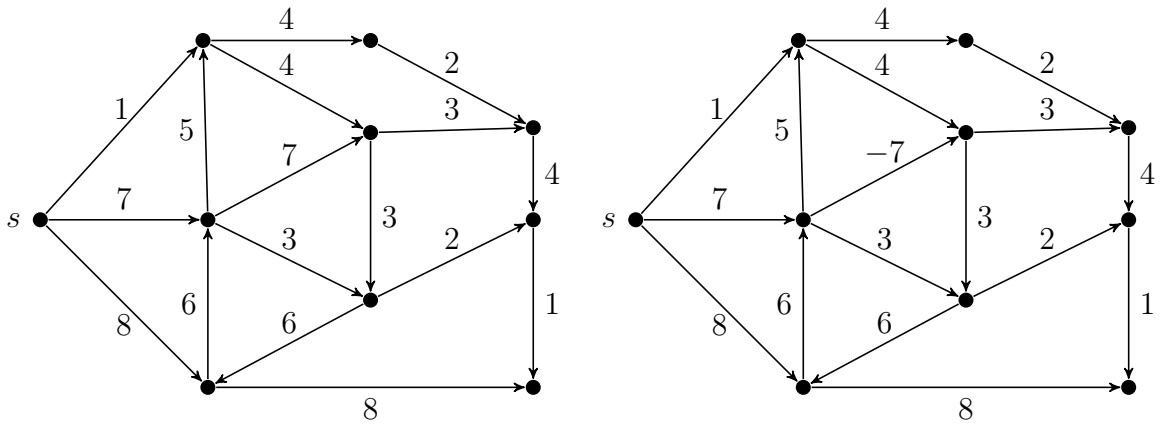


Abbildung 5.9: Wie sieht ein Baum kürzester Wege mit Wurzel s aus?

Aufgabe 5.5 Es sei $G = (V, E)$ ein gerichteter Graph mit Wurzel s, der ein Kommunikationsnetz modelliert: die Knoten stellen die Sender, Empfänger und Verteilerstellen dar und die Kanten die unidirektionalen Kommunikationsverbindungen. Für jede Kante $e \in E$ sei eine Wahrscheinlichkeit $p(e) \in (0, 1)$ bekannt, mit sie ausfällt. Hierbei nehmen wir an, dass diese Wahrscheinlichkeiten unabhängig von einander sind. Für einen Weg

$P = (v_0 = s, e_1, v_1, \ldots, e_l, v_l)$ bezeichnet man dann die Wahrscheinlichkeit, dass er nicht ausfällt, d.h die Wahrscheinlichkeit

$$z(P) := \prod_{i=1}^{l} (1 - p(e_i))$$

als die *Zuverlässigkeit* des Weges.

Geben Sie einen Algorithmus an, der für alle $v \in V$ einen zuverlässigsten Weg von s nach v bestimmt.

6 Matchings

Erinnern wir uns wieder an das Hochzeitsproblem vom Beginn der Vorlesung:

Beispiel 6.1 (Hochzeitsproblem) Graphentheoretisch können wir dies wie folgt modellieren. Wir ordnen jedem Kind und jedem Heiratskandidaten einen Knoten zu. Ist es möglich zwei Personen zu verheiraten, so fügen wir eine Kante zwischen den beiden zugehörigen Knoten ein. So erhalten wir einen ungerichteten Graphen $G = (V, E)$. Jede Kante steht also für eine mögliche Ehe. Daher suchen wir eine möglichst große Teilmenge der Kanten $M \subseteq E$ mit der folgenden Eigenschaft: Betrachten wir zwei beliebige Kanten in M, so dürfen diese keine gemeinsamen Endknoten haben. Dies würde nämlich bedeuten, dass die zugehörige Person mehr als einen Ehepartner hat, und unser Modell sieht Polygamie nicht vor. ◊

Ähnlich gelagerte Probleme mit etwas mehr Anwendungsbezug sind die folgenden:

Beispiel 6.2 (Stellenverteilung) Hat eine Firma mehrere Stellen neu zu besetzen, so steht sie vor einem ähnlichen Problem: Für die Stellen gibt es viele Bewerber, aber nicht jeder Bewerber kann jede Stelle ausfüllen. Außerdem ist zu beachten, dass jede Stelle nur mit einem Bewerber besetzt werden kann und umgekehrt jeder Bewerber höchstens eine Stelle besetzen kann. ◊

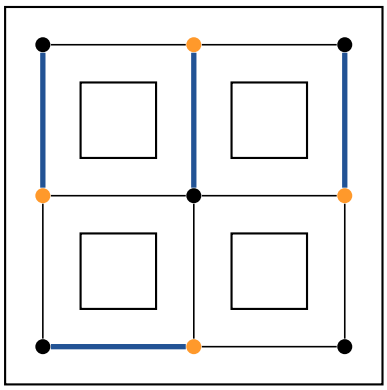


Abbildung 6.1: Ein Museumswärterproblem

Beispiel 6.3 (Museumswärter I) Ein Museum möchte Wärter aufstellen, so dass die ganze Ausstellungsfläche im Blickfeld von mindestens einem Wärter liegt. Hierbei gehen wir davon aus, dass die Wärter einen Rundumblick haben und beliebig weit sehen können.

Da das Museum nur aus engen Gängen besteht, können wir es als Graphen beschreiben, indem wir Kreuzungen, Ecken und Sackgassen jeweils einen Knoten zuordnen und für alle Gänge Kanten einfügen.

Wollen wir aus Kostengründen möglichst wenige Wärter aufstellen, so werden wir diese immer an Knoten aufstellen, da sie dann alle dazu inzidenten Kanten überwachen können. Wir suchen also eine möglichst kleine Teilmenge der Knoten, so dass jede Kante im Graphen zu mindestens einem dieser Knoten inzident ist.

Um eine untere Schranke für die Anzahl der nötigen Wärter zu bekommen, können wir versuchen eine möglichst große Menge an Kanten zu finden, die jeweils keine gemeinsamen Endknoten haben. Jede dieser Kanten muss dann von einem anderen Wärter überwacht werden, daher brauchen wir mindestens so viele Wärter wie wir solche Kanten finden können, vergleiche Abbildung 6.1. ◊

Das allen Beispielen gemeinsame Grundproblem lässt sich wie folgt formalisieren.

Definition 6.4 *Es sei $G = (V, E)$ ein gerichteter oder ungerichteter Graph. Eine Menge $M \subseteq E$ heißt* Matching*, wenn keine zwei Kanten $e, e' \in E$ adjazent sind. Ein Matching heißt* maximal*, wenn es kein Matching $M' \supsetneq M$ gibt.*

Da Adjazenz von Kanten nicht davon abhängt, ob diese gerichtet oder ungerichtet sind, werden wir im folgenden nur ungerichtete Graphen betrachten.

Wir interessieren uns offensichtlich für Matchings mit möglichst vielen Kanten. Dies gibt Anlass zu den folgenden Definitionen.

Definition 6.5 *Es sei $G = (V, E)$ ein ungerichteter Graph und $M \subseteq E$ ein Matching. Alle zu M inzidenten Knoten heißen M-*überdeckt*, alle nicht zu M inzidenten Knoten heißen M-*frei*.*

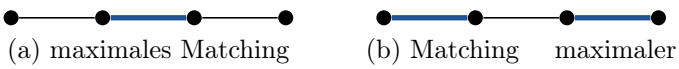
Ein Matching heißt perfekt*, wenn es alle Knoten $v \in V$ überdeckt.*

Die Matching-Größe *von G ist definiert als*

$$\nu(G) := \max\{|M| \mid M \text{ ist ein Matching in } G\}.$$

Offensichtlich besitzt ein Graph G genau dann ein perfektes Matching, wenn $\nu(G) = \frac{|V|}{2}$ gilt.

Achtung: Unser Ziel ist es, Matchings mit möglichst vielen, also $\nu(G)$ vielen Kanten zu finden. Diese sind dann auch immer maximale Matchings, umgekehrt hat aber nicht jedes maximale Matching $\nu(G)$-viele Kanten, vergleiche Abbildung 6.2.



(a) maximales Matching (b) Matching maximaler Kardinalität

Abbildung 6.2: Der Unterschied zwischen maximalen Matchings und Matchings maximaler Kardinalität

6.1 Charakterisierung von Matchings maximaler Kardinalität

Bevor wir uns überlegen, wir wir ein Matching mit $\nu(G)$ vielen Kanten finden, wollen wir uns überlegen, wie man einem gegebenen Matching ansehen kann, ob es diese Eigenschaft hat. Dafür führen wir die folgenden Wege ein.

Definition 6.6 *Es sei $G = (V, E)$ ein ungerichteter Graph und $M \subseteq E$ ein Matching. Ein einfacher Weg P in G heißt* M-alternierend*, wenn seine Kanten abwechselnd in M und nicht in M liegen. Ein M-alternierender Weg, der kein Kreis ist und dessen beide Endknoten M-frei sind, heißt* M-vergrößernder Weg*.*

(a) M (b) M-vergrößernder Weg

Abbildung 6.3: M-alternierende Wege

Existiert ein solcher Weg, vergleiche Abbildung 6.3, so können wir das betrachtete Matching so abändern, dass es mehr Kanten hat.

Lemma 6.7 *Es sei $G = (V, E)$ ein ungerichteter Graph, $M \subseteq E$ ein Matching und P ein M-vergrößernder Weg. Dann ist*

$$M' := (M \cup P) \setminus (M \cap P) = (M \setminus P) \cup (P \setminus M)$$

ein Matching mit $|M'| = |M| + 1 > |M|$.

Beweis Wir zeigen zunächst, dass M' auch ein Matching ist. Angenommen, es gäbe in M' zwei adjazente Kanten. Diese können nicht beide in $M \setminus P$ liegen, da M ein Matching ist. Sie können auch nicht beide in $P \setminus M$ liegen, da P ein M-alternierener Weg ist. Also muss eine der beiden Kanten in $M \setminus P$ liegen und die andere in $P \setminus M$. Der gemeinsame Knoten liegt also auch in P und kann daher keiner der Endknoten sein, da diese M-frei sind. Es kann aber auch keiner der Zwischenknoten sein, da an diesen immer auch eine Kante in $P \cap M$ endet, folglich zwei Kanten in M adjazent wären.

Da P ein M-vergrößernder Weg ist, liegen die erste und letzte Kante von P nicht in M. Folglich ist

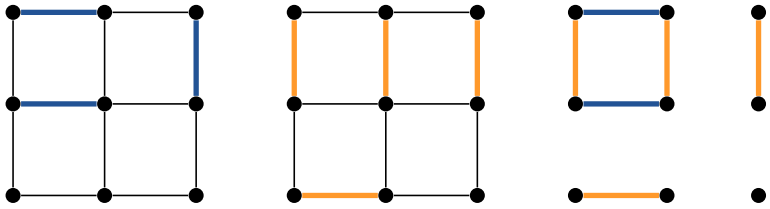
$$|M'| = |M| \underbrace{-|M \cap P| + |P \setminus M|}_{=1} = |M| + 1 > |M|.$$

□

Tatsächlich ist die Bedingung, dass es keinen M-vergrößernden Weg geben darf, nicht nur notwendig sondern auch hinreichend dafür, dass ein Matching M genau $\nu(G)$ Kanten enthält.

Satz 6.8 (Satz von Berge) *Es sei $G = (V, E)$ ein ungerichteter Graph. Für ein Matching $M \subseteq E$ gilt $|M| = \nu(G)$ genau dann, wenn es keinen M-vergrößernden Weg gibt.*

Beweis $\Longrightarrow$ Es sei $M \subseteq E$ ein Matching mit $|M| = \nu(G)$. Dann kann es nach Lemma 6.7 keinen M-vergrößernden Weg geben.



(a) M (b) M' (c) M'

Abbildung 6.4: Illustration des Beweises von Satz 6.8

$\Longleftarrow$ Sei nun umgekehrt $|M| < \nu(G)$. Dann gibt es ein anderes Matching M' mit $|M'| = \nu(G) > |M|$. Setze nun $E' := (M \cup M') \setminus (M \cap M') = (M \setminus M') \cup (M' \setminus M)$ und $G' = (V, E')$. Da jeder Knoten höchstens zu einer Kante aus M und einer aus M' inzident sein kann, hat dann jeder Knoten in G' höchstens Grad 2, d.h. alle Zusammenhangskomponenten von G' sind entweder einzelne Knoten, einfache Wege oder einfache Kreise gerader Länge. Da die Kreise gleich viele Kanten aus M und M' enthalten, E' aber mehr Kanten aus M' als aus M enthält, muss es mindestens eine Zusammenhangskomponente in G' geben, die kein Kreis ist und deren beide Endkanten zu M' gehören. Diese Zusammenhangskomponente ist dann ein M-vergrößernder Weg. □

Dieses Resultat liefert eine erste Idee für einen Algorithmus zur Bestimmung von Matchings maximaler Kardinalität: Man beginnt mit einem leeren Matching und vergrößert dieses so lange wie in Lemma 6.7 angegeben, bis kein M-vergrößernder Pfad mehr existiert. Dies ist nach maximal $\frac{|V|}{2}$ Iterationen der Fall und liefert nach den obigen Satz ein Matching größtmöglicher Kardinalität.

Die einzige verbleibende Frage ist: Wie findet man M-vergrößernde Pfade? In beliebigen Graphen ist dies zwar möglich, das dafür nötige Verfahren aber sehr kompliziert. Häufig haben die Graphen, in denen man Matchings sucht, jedoch spezielle Eigenschaften, die das Problem vereinfachen. Solche Graphen wollen wir im nächsten Abschnitt genauer untersuchen.

6.2 Bipartite Graphen

Erinnern wir uns an das Hochzeitsproblem oder das Stellenverteilungsproblem, so kann man in dem zugehörigen Graphen die Knoten in zwei Mengen aufteilen (Männer und Frauen beziehungsweise Stellen und Bewerber), so dass Kanten nur von einer Knotenmenge in die andere, aber nicht innerhalb einer Knotenmenge verlaufen.

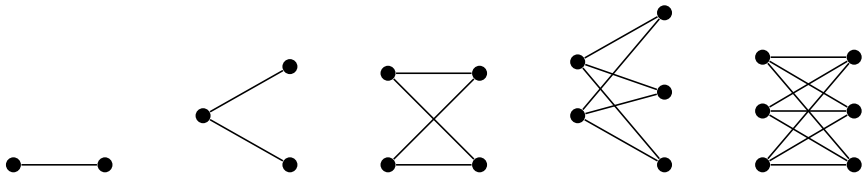
Definition 6.9 *Es sei $G = (V, E)$ ein gerichteter oder ungerichteter Graph. G heißt* bipartit*, wenn es eine Partition der Knotenmenge $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$ gibt, so dass jede Kante $e \in E$ sowohl mit einem Knoten in V_1 als auch mit einem Knoten in V_2 inzidiert.*

Ob ein Graph bipartit ist, oder nicht, hängt offensichtlich nicht davon ab, ob seine Kanten orientiert sind. Daher betrachten wir im folgenden nur ungerichtete Graphen.

In zusammenhängenden Graphen ist die Partition der Knoten eindeutig, in nicht zusammenhängenden Graphen ist sie nicht eindeutig.

Beispiele für bipartite Graphen sind die folgenden. Einer davon wird uns später noch einmal beschäftigen.

Definition 6.10 *Mit $K_{n,m}$ bezeichnen wir den vollständigen bipartiten Graphen mit $|V| = n + m$, der eine Partition mit $|V_1| = n$ und $|V_2| = m$ besitzt, vergleiche Abbildung 6.5.*



(a) $K_{1,1}$ (b) $K_{1,2}$ (c) $K_{2,2}$ (d) $K_{2,3}$ (e) $K_{3,3}$

Abbildung 6.5: Einige bipartite Graphen

Wie erkennt man nun, ob ein Graph bipartit ist? Wie wir sehen werden, lassen sich bipartite Graphen einfach mit Hilfe von Kreisen charakterisieren.

Satz 6.11 *Ein ungerichteter Graph $G = (V, E)$ ist genau dann bipartit, wenn er keine Kreise ungerader Länge enthält.*

Beweis $\Longrightarrow$ Es sei G bipartit mit der Knotenpartition $V = V_1 \cup V_2$. Angenommen, es gibt einen Kreis $P = (v_0, e_1, v_1, \ldots, e_l, v_l = v_0)$ ungerader Länge. Dann können wir ohne Beschränkung der Allgemeinheit $v_0 \in V_1$ annehmen. Da G bipartit ist, folgt $v_1 \in V_2$, $v_3 \in V_1$ und so weiter. Da P ungerade Länge hat, folgt $v_l \in V_2$. Andererseits gilt $v_l = v_0 \in V_1$, ein Widerspruch.

$\Longleftarrow$ G ist genau dann bipartit, wenn alle Zusammenhangskomponenten bipartit sind. Es genügt daher zu zeigen dass für zusammenhängende Graphen daraus, dass keine Kreise ungerader Länge existieren, folgt, dass sie bipartit sind. Sei also G zusammenhängend und $s \in V$ beliebig. Dann gibt es für alle $v \in V$ einen Weg zwischen s und v in G und wir bezeichnen mit $\text{dist}(s, v)$ die Länge (=Kantenzahl) eines kürzesten solchen Weges. Damit partitionieren wir V wie folgt:

$$\begin{aligned} V_1 &:= \{v \in V \mid \text{dist}(s, v) \text{ ist ungerade}\}, \\ V_2 &:= \{v \in V \mid \text{dist}(s, v) \text{ ist gerade}\}. \end{aligned}$$

Diese beiden Mengen bilden offensichtlich eine Partition von V und wir behaupten, dass G mit dieser Partition bipartit ist. Angenommen, die ist nicht der Fall. Dann gibt es eine Kante $e = \{u, v\} \in E$, deren Endknoten u, v beide in X mit $X = V_1$ oder $X = V_2$ liegen. Dann enthält G aber einen Kreis P von s über u und v nach s der Länge $\text{dist}(s, u) + \text{dist}(s, v) + 1$. Da $\text{dist}(s, u)$ und $\text{dist}(s, v)$ entweder beide gerade oder ungerade sind, ist ihre Summe immer gerade. Der Kreis P hat also ungerade Länge, ein Widerspruch zur Voraussetzung. □

Damit folgt sofort, dass eine wichtige Klasse von Graphen, die wir bereits ausführlich betrachtet haben, bipartit ist.

Korollar 6.12 *Es sei $G = (V, E)$ ein ungerichteter Graph. Ist G ein Baum, so ist G bipartit.*

Der Beweis des obigen Satzes ist die Grundlage des folgenden Algorithmus 12, der für einen zusammenhängenden Graphen überprüft, ob er bipartit ist und gegebenenfalls eine Partition der Knotenmenge bestimmt. Nichtzusammenhängende Graphen sind genau dann bipartit, wenn alle ihre Zusammenhangskomponenten bipartit sind, hier kann der Algorithmus also auf die Zusammenhangskomponenten angewendet werden.

Algorithmus 12 Algorithmus zur Bestimmung der Knotenpartition in bipartiten Graphen

Input: Ein ungerichteter zusammenhängender Graph $G = (V, E)$.

1 Wähle einen beliebigen Startknoten s und bestimme mit der Breitensuche Wege mit minimaler Kantenzahl zu allen anderen Knoten $v \in V$.

2 Für alle $v \in V$ definiere $d(v)$ als den von der Breitensuche bestimmten Abstand des Knotens v zum Startknoten s.

3 Setze

$$\begin{aligned} V_1 &:= \{v \in V \mid d(v) \text{ ist ungerade}\}, \\ V_2 &:= \{v \in V \mid d(v) \text{ ist gerade}\}. \end{aligned}$$

4 **for all** $e = \{u, v\} \in E$ **do**

5 **if** $u, v \in V_1$ **or** $u, v \in V_2$ **then**

6 **Fehlermeldung:** G ist nicht bipartit

7 **end if**

8 **end for**

Output: Für bipartite Graphen eine Knotenpartition $V = V_1 \cup V_2$.

Dass dieser Algorithmus wohldefiniert ist und das Gewünschte leistet, vergleiche zum Beispiel Abbildung 6.6, folgt sofort aus dem Beweis von Satz 6.11.

Satz 6.13 *Es sei $G = (V, E)$ ein zusammenhängender ungerichteter Graph. Dann bestimmt Algorithmus 12 korrekt, ob G bipartit ist und gibt in diesem Fall eine zugehörige Knotenpartition $V = V_1 \cup V_2$ zurück.*

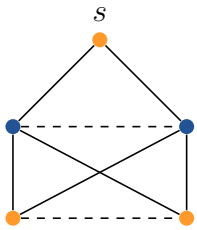


Abbildung 6.6: Illustration von Algorithmus 12

6.3 Matchings in bipartiten Graphen

Nun zurück zu unserem ursprünglichen Problem: Wie findet man ein Matching maximaler Kardinalität in bipartiten Graphen? Ein Verfahren zur Lösung dieses Problems ist der sogenannte *ungarische Algorithmus*, vergleiche Algorithmus 13.

Algorithmus 13 Der ungarische Algorithmus

Input: Ein ungerichteter bipartiter Graph $G = (V, E)$ mit zugehöriger Knotenpartition $V = V_1 \cup V_2$.

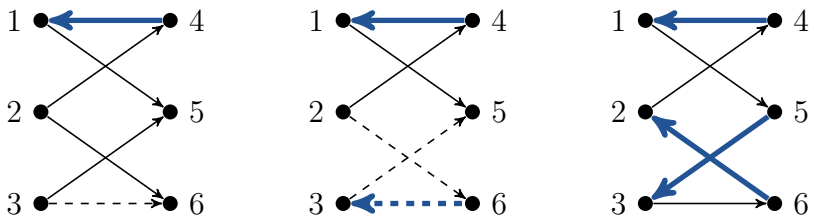
1 Wähle ein beliebiges Matching M, z.B. $M = \emptyset$ oder $M = \{e\}$ mit einem $e \in E$. Orientiere alle Kanten $e \in M$, so dass sie von V_2 nach V_1 zeigen, und alle Kanten $e \in E \setminus M$, so dass sie von V_1 nach V_2 zeigen.
2 Bestimme die Menge F_1 aller M-freien Knoten in V_1 und die Menge F_2 aller M-freien Knoten in V_2.
3 Bestimme die Menge R_2 aller Knoten $v_2 \in V_2$, die durch einen gerichteten Weg mit Startknoten in $v_1 \in F_1$ erreichbar sind. (Dies geht z.B. durch wiederholtes Anwenden der Breitensuche mit Startknoten in $v_1 \in F_1$.)
4 **while** $R_2 \cap F_2 \neq \emptyset$ **do**
5 Wähle einen Knoten $v_2 \in R_2 \cap F_2$ und kehre auf dem zugehörigen gerichteten Weg die Orientierung aller Kanten um.
6 Definiere M als die Menge aller Kanten e mit Startknoten in V_2.
7 Bestimme F_1, F_2 und R_2 neu.
8 **end while**

Output: Ein Matching M maximaler Kardinalität.

Bevor wir beweisen, dass der Algorithmus die gewünschten Eigenschaften hat, wollen wir an einem kleinen Beispiel nachvollziehen, wie er funktioniert. Betrachte dazu Abbildung 6.7 und Tabelle 6.1.

Satz 6.14 *Es sei $G = (V, E)$ ein ungerichteter bipartiter Graph mit Knotenpartition $v = V_1 \cup V_2$. Dann ist Algorithmus 13 wohldefiniert und bestimmt ein Matching M maximaler Kardinalität.*

Beweis Wir zeigen zunächst, dass die Menge $M \subseteq E$ in jeder Iteration ein Matching ist und $|M|$ in jeder Iteration um 1 wächst. Zu Beginn des Algorithmus ist M nach Voraus-



(a) 1. Iteration (b) 2. Iteration (c) 3. Iteration

Abbildung 6.7: Illustration des Ungarischen Algorithmus

M	F_1	F_2	R_2	$F_2 \cap R_2$
$\{\{1,4\}\}$	$\{2,3\}$	$\{5,6\}$	$\{4,5,6\}$	$\{5,6\}$
$\{\{1,4\},\{3,6\}\}$	$\{2\}$	$\{5\}$	$\{4,5,6\}$	$\{5\}$
$\{\{1,4\},\{2,6\},\{3,5\}\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Tabelle 6.1: Illustration des Ungarischen Algorithmus

setzung ein Matching. Sei nun M zu Beginn einer Iteration ein Matching und P der Weg, dessen Orientierung in dieser Iteration umgekehrt wird. Dann ist P ein M-vergrößernder Weg, denn seine Endknoten sind nach Konstruktion M-frei und er ist M-alternierend, da alle Kanten von V_1 nach V_2 in $E \setminus M$ liegen, alle Kanten von V_2 nach V_1 zu M gehören und innerhalb von V_1 und V_2 keine Kanten existieren. Nach Lemma 6.7 ist daher auch die in dieser Iteration neu erzeugte Menge M ein Matching und hat eine um 1 größere Kardinalität.

Folglich ist der Algorithmus wohldefiniert, da er nach höchstens $\frac{|V|}{2}$ Iterationen abbricht und er erzeugt ein Matching.

Es bleibt zu zeigen, dass dieses Matching M maximale Kardinalität hat. Nach Satz 6.8 genügt es dafür zu zeigen, dass es keinen M-vergrößernden Weg mehr gibt. Gäbe es einen M-vergrößernden Weg P mit Endknoten v_1, v_2, so müssten dieser eine ungerade Kantenzahl haben. Für seine M-freien Endknoten müsste daher (bei geeigneter Bezeichnung) $v_1 \in F_1$ und $v_2 \in F_2$ gelten. Da v_2 von $v_1 \in F_1$ aus durch einen M-alternierenden Weg erreichbar ist, würde aber auch $v_2 \in R_2$ gelten, d.h. es wäre $F_2 \cap R_2 \neq \emptyset$, ein Widerspruch dazu, dass der Algorithmus terminiert hat. □

Da wir uns für Matchings maximaler Kardinalität interessieren, liegt die Frage nahe, wann ein Graph ein perfektes Matching besitzt. Bei bipartiten Graphen mit Knotenpartition $V = V_1 \cup V_2$ interessiert man sich außerdem dafür, ob es ein Matching M gibt, so dass zumindest alle Knoten in einer der beiden Mengen V_1, V_2 M-überdeckt sind.

Satz 6.15 (Satz von Hall) *Es sei $G = (V, E)$ ein ungerichteter bipartiter Graph mit Knotenpartition $V = V_1 \cup V_2$. Dann gibt es genau dann ein Matching M, so dass alle Knoten $v_1 \in V_1$ M-überdeckt sind, wenn gilt: Für alle $S \subseteq V_1$ ist*

$$|N(S)| := |\{v_2 \in V_2 \mid \exists_{v_1 \in S} \{v_1, v_2\} \in E\}| \geq |S|.$$

Man nennt die Menge $N(S)$ die Nachbarschaft *der Menge S.*

Beweis $\Longrightarrow$ Gibt es ein Matching, dass alle $v_1 \in V_1$ versorgt, so gilt für jedes $S \subseteq V_1$: Für alle $v_1 \in S$ existiert ein $v_2 \in V_2$ mit $\{v_1, v_2\} \in M$, d.h. es gilt $v_2 \in N(S)$. Da M ein Matching ist, müssen alle diese v_2 verschieden sein. Folglich gilt $|N(S)| \geq |S|$.

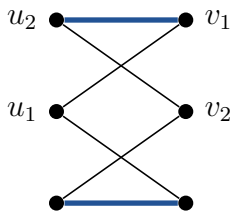


Abbildung 6.8: Illustration des Beweises von Satz 6.15

$\Longleftarrow$ Sei nun umgekehrt $|N(S)| \geq |S|$ für alle $S \subseteq V_1$. Dann konstruieren wir ein Matching, das ganz V_1 versorgt. Sei hierfür M ein beliebiges Matching, so dass es noch M-freie Knoten in V_1 gibt. Wir zeigen, dass es dann auch einen M-vergrößernden Weg gibt: Sei $u_1 \in V_1$ ein M-freier Knoten. Nach Voraussetzung ist $|N(\{u_1\})| \geq |\{u_1\}| = 1$, d.h. der Knoten u_1 hat mindestens einen Nachbarn $v_1 \in V_2$ und es gilt $\{u_1, v_1\} \in E \setminus M$. Ist dieser auch M-frei, so haben wir einen M-vergrößernden Weg gefunden. Ist er hingegen M-versorgt, so gibt es eine Kante $\{u_2, v_1\} \in M$. Nun betrachten wir den Knoten u_2. Es gilt $u_2 \neq u_1$, da u_2 M-versorgt ist. Wegen $|N(\{u_1, u_2\})| \geq |\{u_1, u_2\}| = 2$ gibt es in V_2 einen weiteren, zu einem $u_i \in \{u_1, u_2\}$ benachbarten Knoten $v_2 \neq v_1$. Es ist dann wieder $\{u_i, v_2\} \in E \setminus M$ für $i = 1, 2$, da u_1 M-frei ist und in u_2 bereits die Matchingkante $\{u_2, v_1\}$ endet. Ist dieser M-frei, so können wir einen M-vergrößernden Weg konstruieren, indem wir von w_2 zu seinem Vorgänger in $\{u_1, u_2\}$ zurück gehen und so weiter, bis wir wieder beim Knoten u_1 sind. Dies liefert einen M-alternierenden Weg, da wir uns (auf dem Rückweg!) von V_2 nach V_1 immer auf Kanten in $E \setminus M$ bewegen und von V_1 nach V_2 auf Kanten in M. Ist v_2 M-versorgt, so gibt es eine Kante $\{u_3, v_2\} \in M$. Da u_3 M-versorgt ist und $v_2 \neq v_1$ gilt, folgt $u_3 \notin \{u_1, u_2\}$. Von u_3 können wir wieder weitergehen wie von u_2 aus. Da die Mengen $\{u_1, u_2, \ldots\} \subseteq V_1$ und $\{v_1, v_2, \ldots\} \subseteq V_2$ immer größer werden, muss dieses Verfahren irgendwann abbrechen. Ein Abbruch ist aber nur möglich, wenn ein M-freies v_i und damit ein M-vergrößernder Weg gefunden wurde. □

Ein bipartiter Graph kann offensichtlich nur dann ein perfektes Matching besitzen, wenn $|V_1| = |V_2|$ gilt. In diesem Fall gibt Satz 6.15 eine notwendige und hinreichende Bedingung für die Existenz eines perfekten Matchings an und ist auch unter dem Namen *Heiratssatz* bekannt.

6.4 Aufgaben

Aufgabe 6.1

(a) Zeigen Sie, dass ein Baum $T = (V, E)$ höchstens ein perfektes Matching besitzen kann.

(b) Finden Sie ein Beipiel für einen Baum, der kein perfektes Matching besitzt.

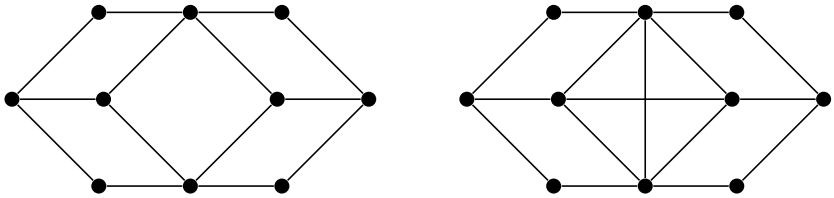


Abbildung 6.9: Bipartite Graphen?

Aufgabe 6.2 Verwenden Sie Algorithmus 12 um zu entscheiden, ob die Graphen in Abbildung 6.9 bipartit sind.

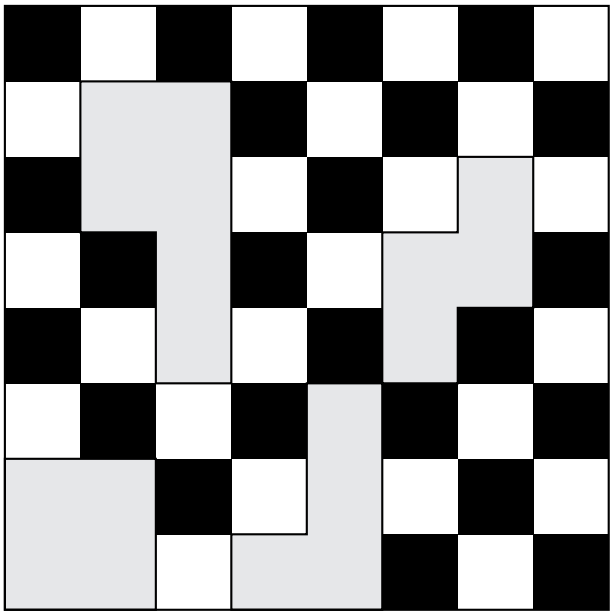


Abbildung 6.10: Ein löchriges Schachbrett

Aufgabe 6.3 Betrachten Sie das löchrige Schachbrett aus Abbildung 6.10. Stellen Sie sich vor, Sie haben Dominosteine, die genau so groß wie zwei benachbarte Felder des Schachbretts sind, also ein schwarzes und ein weißes Feld. Mit diesen Steinen wollen Sie das gesamte Spielbrett (ohne Löcher) überdecken, so dass an keiner Stelle Dominosteine mehrfach liegen oder über das Feld hinausragen.

(a) Formulieren Sie das Problem graphentheoretisch.

(b) Können Sie Ihr Ziel erreichen?

Aufgabe 6.4 Es sei $G = (V, E)$ ein ungerichteter Graph. Dann betrachten wir das folgende Spiel für zwei Personen: Spieler 1 wählt einen beliebigen Knoten $v_1 \in V$. Spieler 2 wählt einen dazu benachbarten Knoten $v_2 \in V$. Dann ist wieder Spieler 1 an der Reihe und wählt einen zu v_2 benachbarten, noch unbesuchten Knoten $v_3 \in V$, und so weiter. Wer als letzes einen Knoten wählen kann, hat gewonnen.

Zeigen Sie: Falls G ein perfektes Matching besitzt, so kann Spieler 2 immer gewinnen. Geben Sie die Gewinnstrategie an.

Aufgabe 6.5 Die Heiratsvermittlung *Online-Matching* sucht für n unverheiratete Herren $H = \{h_1, h_2, \ldots, h_n\}$ die passende Partnerin und hat dazu ebenfalls n unverheiratete Damen $D = \{d_1, d_2, \ldots, d_n\}$ zu einer Tanzveranstaltung eingeladen. Der Einfachheit halber wollen wir voraussetzen, dass n gerade ist. Jeder der Damen bekam eine Einladung mit Portraits aller Herren zugesandt und wurde gebeten, darauf diejenigen anzukreuzen, die ihr gefallen und die Einladung zum Tanz mitzubringen. Wir nehmen nun an, dass alle eingeladenen Damen erscheinen und ihre ausgefüllte Liste mitbringen. Der Portier bekommt nun die Aufgabe, jeder ankommenden Dame einem Herren vorzustellen, den sie angekreuzt hat. Ist keiner dieser Herren mehr frei, so muss sie wieder heimgehen. Die Heiratsagentur hat natürlich ein Interesse daran, dass möglichst vielen Herren eine interessierte Dame vorgestellt wird.

Das Problem lässt sich durch einen bipartiten Graphen $G = (H \cup D, E)$ mit $2n$ Knoten modellieren, in dem ein möglichst großes Matching gesucht wird. Wir wollen voraussetzen, dass die Interessen der Damen derart sind, dass ein perfektes Matching existiert.

(a) Geben Sie Entscheidungsregeln für den Portier an, so dass am Ende der Veranstaltung mindestens $n/2$ Herren einer interessierten Dame vorgestellt wurden.

(b) Zeigen Sie, dass es, egal welchen Regeln der Portier folgt, immer eine Menge von n Damen mit gewissen Interessen und eine Ankunftsreihenfolge gibt, so dass höchstens $n/2$ Herren einer interessierten Dame vorgestellt werden können, obwohl ein perfektes Matching existiert.

7 Eulerwege und -kreise

Erinnern wir und noch einmal an das Haus vom Nikolaus.

Beispiel 7.1 (Haus vom Nikolaus) Mit Hilfe der Graphentheorie können wir die Frage, ob sich das Haus vom Nikolaus ohne Absetzen zeichnen lässt, wie folgt modellieren: Wir suchen in dem Graphen, der das Haus vom Nikolaus darstellt, vergleiche Aufgabe 2, einen Weg, auf den jede Kante des Graphen genau einmal durchlaufen wird. ◊

Es gibt weitere bekannte Probleme, die auf eine ähnliche Fragestellung hinauslaufen:

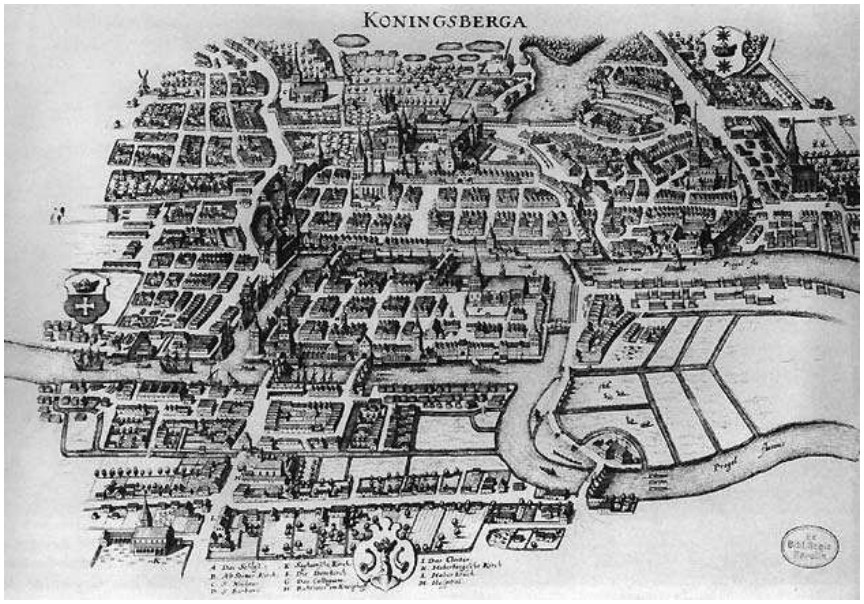


Abbildung 7.1: Karte von Königsberg 1652 (www.preussen-chronik.de)

Beispiel 7.2 (Königsberger Brückenproblem) Das Königsberger Brückenproblem aus dem Jahr 1736 kann als die Geburtsstunde der Graphentheorie betrachtet werden. In Königsberg umfließen die beiden Arme der Pregel eine Insel, den sogenannte Kneiphof. Über die Pregel führten in Königsberg insgesamt sieben Brücken und man stellte sich die Frage, ob es einen Rundweg durch die Stadt gibt, der über jede dieser Brücken genau einmal führt. ◊

Beispiel 7.3 (Der chinesische Postbote) Ein Postbote bekommt eine Menge von Straßen zugeteilt, in denen er Briefe austeilen muss. Um Zeit und Wege zu sparen, versuchet

er die Briefe möglichst effizient auszuteilen. Geht man davon aus, dass er in einer Straße gleichzeitig Briefe auf beiden Straßenseiten einwirft und am Ende seiner Runde wieder seine Posttasche an seiner Dienststelle abliefern muss, so sucht er einen möglichst kurzen Rundweg, der jede Straße in seinem Revier mindestens einmal durchläuft. Woher weiß der Postbote, wie er laufen muss?

Dieses Problem wird als chinesisches Postbotenproblem bezeichnet, weil es 1960 erstmals von dem chinesischen Mathematiker Mei Ko Kwan untersucht wurde. ◊

Wer mehr über die Geschichte dieser beiden Probleme wissen möchte, dem sei der Artikel [4] ans Herz gelegt.

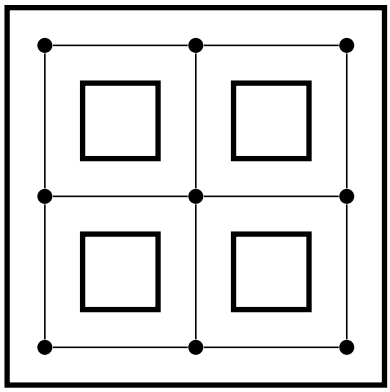


Abbildung 7.2: Wie könnte ein Rundgang aussehen?

Beispiel 7.4 (Museumstour) Das im letzten Kapitel schon betrachtete Museum möchte seinen Besuchern einen Rundgang vorschlagen, bei dem sie an allen Exponaten möglichst genau einmal vorbeilaufen. Dabei wird davon ausgegangen, dass ein Besucher die Ausstellungsstücke auf beiden Seiten eines Ganges gleichzeitig betrachtet. Ist ein Rundgang, bei dem jeder Gang genau einmal durchlaufen wird, möglich, wenn Eingang und Ausgang des Museums an der gleiche Stelle sind? Ist ein solcher Rundgang möglich, wenn Ein- und Ausgang an verschiedenen Stellen liegen? ◊

In allen obigen Beispielen suchen wir einen Weg mit den folgenden Eigenschaften:

Definition 7.5 *Es sei $G = (V, E)$ ein gerichteter oder ungerichteter Graph. Ein Weg P in G heißt* Euler-Weg*, wenn er jede Kante in E genau einmal durchläuft. Ist P zusätzlich ein Kreis, so nennt man P einen* Euler-Kreis*. Der Graph G heißt* eulersch*, wenn er einen Euler-Kreis enthält.*

7.1 Der Satz von Euler

Wir betrachten zunächst nur gerichtete Graphen und wollen uns überlegen, wann diese eulersch sind:

Satz 7.6 (Satz von Euler (I)) *Es sei $G = (V, E)$ ein gerichteter und schwach zusammenhängender Graph. Der Graph G ist genau dann eulersch, wenn für alle Knoten $v \in V$ gilt*

$$g^+(v) = g^-(v).$$

Beweis $\Longrightarrow$ Ist $E = \emptyset$, so sind alle Knotengrade 0, also gerade. Sei nun $E \neq \emptyset$ und $P = (v_0, e_1, v_1 \ldots, e_l, v_l = v_0)$ mit $l \geq 2$ ein Euler-Kreis in G. Da G schwach zusammenhängend ist, ist $g(v) \neq 0$ für alle $v \in V$ und folglich liegen alle Knoten $v \in V$ auf dem Kreis P. Da der Start-/Endknoten eines Kreises beliebig gewählt werden kann, genügt es deshalb zu zeigen, dass $g^+(v_0) = g^-(v_0)$ ist. Wenn wir dem Kreis P folgen, so verlassen wir den Knoten v_0 genau so oft, wie wir ihn wieder betreten. Da auf einem Euler-Kreis jede Kante genau einmal durchlaufen wird, müssen wir bei jedem Betreten und Verlassen des Knotens v_0 eine neue Kante durchlaufen und haben am Ende von P alle Kanten durchlaufen. Daher muss für den Knoten v_0 gelten $g^+(v_0) = g^-(v_0)$.

$\Longleftarrow$ Sei nun umgekehrt $g^+(v) = g^-(v)$ für alle $v \in V$. Wir müssen nun zeigen, dass es einen Euler-Kreis in G gibt. Ist $E = \emptyset$, so ist wiederum nichts zu zeigen. Sei daher nun $E \neq \emptyset$. Wir konstruieren nun einen Euler-Kreis auf eine Art, die später auch die Grundlage eines Algorithmus zur Bestimmung von Euler-Kreisen sein wird. Dazu wählen wir einen beliebigen Startknoten v_0. Da G schwach zusammenhängend ist, ist $g(v_0) \neq 0$ und daher auch $g^+(v_0) \neq 0$. Folglich gibt es mindestens eine Kante mit Anfangsknoten v_0. Wir wählen also eine bisher unbenutzte Kante e_1 mit Anfangsknoten v_0 und bezeichnen den Endknoten von e_1 mit v_1. Wegen $g^+(v_1) = g^-(v_1)$ muss es nun auch eine (unbenutzte) Kante e_2 geben, die v_1 verlässt. Wir setzen dieses Verfahren solange fort, bis wir in einem Knoten landen, den keine unbenutzte Kante mehr verlässt. Dieser Knoten muss nun aber wieder v_0 sein, da nach Voraussetzung jeden Knoten genau so viele Kanten verlassen wie ihn betreten. Wir haben also einen einfachen Kreis erzeugt, den wir mit P_1 bezeichnen wollen.

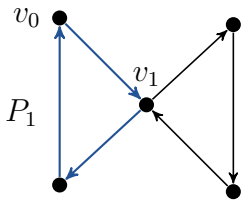


Abbildung 7.3: Illustration des Beweises von Satz 7.6

Enthält P_1 schon alle Kanten in E, so haben wir einen Euler-Kreis konstruiert. Enthält P_1 noch nicht alle Kanten, so gibt es wegen des schwachen Zusammenhangs von G einen Knoten v_1 auf P, von dem noch unbenutzte Kanten ausgehen. Von diesem Knoten ausgehend beginnen wir wieder das gleiche Verfahren wie zuvor und erzeugen so einen weiteren Kreis P_2. Diesen können wir mit dem Kreis P_1 verbinden, indem wir ihn in P_1 an einer Stelle, an der der Knoten v_1 betreten wird, einfügen.

Enthalten $P_1 \cup P_2$ immer noch nicht alle Kanten in E, so wiederholen wir diesen Schritt solange, bis alle Kanten enthalten sind. Da G nur endlich viele Kanten enthält, ist dies nach endlich vielen Schritten der Fall und wir haben einen Euler-Kreis konstruiert. □

In Satz 7.6 haben wir statt starkem Zusammenhang an G nur die vermeintlich schwächere Forderung des schwachen Zusammenhangs gestellt. Das folgende Resultat zeigt jedoch, dass im Fall von eulerschen Graphen beide Bedingungen zusammenfallen.

Korollar 7.7 *Es sei $G = (V, E)$ ein gerichteter und schwach zusammenhängender Graph mit $g^+(v) = g^-(v)$ für alle $v \in V$. Dann ist G stark zusammenhängend.*

Beweis Nach Satz 7.6 besitzt G dann einen Euler-Kreis. Über diesen Kreis sind alle Knoten in V paarweise durch einen (gerichteten!) Weg verbunden, folglich ist G stark zusammenhängend. □

Als nächstes wollen wir untersuchen, wenn in einem gerichteten Graphen ein Euler-Weg, der kein Kreis ist, existiert.

Satz 7.8 (Satz von Euler (II)) *Es sei $G = (V, E)$ ein gerichteter und schwach zusammenhängender Graph. Der Graph G besitzt genau dann einen Euler-Weg, der kein Kreis ist, wenn es zwei Knoten $s, t \in V$ gibt mit*

$$g^+(s) = g^-(s) + 1, \quad g^+(t) = g^-(t) - 1$$

und

$$g^+(v) = g^-(v)$$

für alle $v \in V \setminus \{s, t\}$. Die Knoten s und t sind dann Start- beziehungsweise Endknoten jedes Euler-Weges in G.

Beweis $\Longrightarrow$ Ist P ein Euler-Weg in G, der kein Kreis ist, so folgt analog zum Beweis von Satz 7.6 $g^+(v) = g^-(v)$ für alle Knoten außer dem Start- und Endknoten. Für den Startknoten gilt $g^+(s) = g^-(s) + 1$ und für den Endknoten $g^+(t) = g^-(t) - 1$.

$\Longleftarrow$ Sei G ein schwach zusammenhängender Graph, dessen Knoten die angegebenen Gradeigenschaften besitzen. Betrachte dann den Graphen $G' = (V, E \cup \{(t, s)\})$, d.h füge G eine zusätzliche Kante von t nach s hinzu. (Existiert diese Kante schon, so füge eine Parallele hinzu.) Dann ist G' ein schwach zusammenhängender Graph mit $g^+(v) = g^-(v)$ für alle $v \in V$. Nach Satz 7.6 besitzt G' daher einen Euler-Kreis P. Entfernt man aus P die zusätzliche Kante (t, s), so entsteht ein Euler-Weg mit Startknoten s und Endknoten t. □

Für ungerichtete Graphen lassen sich analoge Resultate zur Existenz von Euler-Kreisen und -wegen zeigen.

Satz 7.9 *Es sei $G = (V, E)$ ein ungerichteter zusammenhängender Graph.*

(a) Der Graph G ist genau dann eulersch, wenn alle Knoten geraden Grad $g(v)$ haben.

(b) Der Graph G enthält genau dann einen Euler-Weg, der kein Kreis ist, wenn genau zwei Knoten s und t ungeraden Grad haben. (Diese Knoten sind dann Start- und Endknoten jedes Euler-Wegs.)

Beweis Wir könnten die beiden Aussagen analog zu den Beweisen von Satz 7.6 und 7.8 zeigen. Statt die Beweise noch einmal nachzuvollziehen, wollen wollen wir hier aber den ungerichteten Fall auf den gerichteten zurückführen. Dazu orientieren wir die ungerichteten Kanten in G auf geeignete Art und versuchen so einen gerichteten Graphen zu erzeugen, der den Bedingungen aus den beiden eben genannten Sätzen genügt. Die dann existierenden gerichteten Euler-Kreise bzw. -wege sind natürlich auch Euler-Kreise bzw. -wege im ungerichteten Graphen G.

Sei also $G' = (V, E')$ der Graph G mit beliebig orientierten Kanten. Dann definieren wir die *Ladung* einen Knotens $v \in V$ als $l(v) = g^+(v) - g^-(v)$ und den *Defekt* des gerichteten Graphen G' als $\sum_{v \in V} |l(v)|$. Können wir die Kanten von G so orientieren, dass G' den Defekt 0 hat, so besitzt G' und damit auch G nach Satz 7.6 einen Euler-Kreis. Ist der minimale Defekt von G stattdessen 2, so folgt mit Lemma 1.13 und Satz 7.8, dass G' und damit G einen Euler-Weg besitzt, der kein Kreis ist.

Wir betrachten ab jetzt nur noch Fall (a), Aussage (b) zeigt man genauso. Da in G alle Knoten geraden Grad haben, sind auch alle Ladungen gerade. Ist P ein Weg in G' von v nach u mit $l(v) > 0$ (also $l(v) \geq 2$) und $l(u) < 0$ (also $l(u) \leq -2$), so führt ein Umdrehen der Orientierung aller Kanten in P dazu, dass die Ladungen der inneren Knoten von P unverändert bleiben, die Ladung von v sich um 2 verringert und die von u sich um 2 erhöht. Der Defekt von G' würde sich also um 4 verringern.

Sei nun G' eine Orientierung von G mit minimalem Defekt. Ist der Defekt von G' gleich 0, so ist nichts mehr zu zeigen. Ist der Defekt von G' positiv, so gibt es nach Lemma 1.13 mindestens einen Knoten mit positiver Ladung. Sei $U \subseteq V$ die Menge aller Knoten, die in G' von einem Knoten v mit $l(v) > 0$ aus über einen (gerichteten) Weg erreichbar sind. Dann darf U aufgrund der obigen Überlegung nur Knoten mit nichtnegativer Ladung enthalten, da sich sonst der Defekt von G' verringern ließe. Folglich ist

$$\sum_{v \in U} g^+(v) - \sum_{v \in U} g^-(v) = \sum_{v \in U} l(v) > 0.$$

Nach Lemma 1.13 ist daher $U \neq V$ und es gibt mindestens eine Kante, die aus U herausführt. Dies widerspricht aber der Definition von U. □

Mit diesen Resultaten können wir schon einige der anfangs aufgeworfenen Fragen beantworten:

Das Haus vom Nikolaus besitzt einen Euler-Weg, weil es genau zwei Knoten mit ungeradem Grad gibt. Beim Zeichnen muss man also immer in einem dieser beiden Knoten beginnen. Zieht der Weihnachtsmann nebendran ein, so gibt es vier Knoten mit ungeraden Grad und daher keinen Euler-Weg mehr.
Das Königsberger Brückenproblem lässt sich durch den Graphen aus Abbildung 7.4 darstellen. Wie man sofort sieht, haben hier alle vier Knoten ungeraden Grad, es gibt also keinen Euler-Weg und folglich auch keinen Euler-Kreis.

Auch das anfangs betrachtete Museum hat zu viele (fünf) Knoten mit ungeradem Grad. Es besitzt also keinen Euler-Kreis und auch keinen Euler-Weg.

Das Problem des Postboten werden wir uns später noch einmal genauer anschauen.

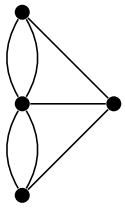


Abbildung 7.4: Graphendarstellung des Königsberger Brückenproblems

7.2 Algorithmus von Hierholzer

Algorithmus 14 Algorithmus von Hierholzer

Input: Ein gerichteter eulerscher Graph $G = (V, E)$.

1 Wähle einen Startknoten $s \in V$.
2 Initialisiere den Eulerkreis $P := (s)$, die Wartemenge $S = V$ und setze die Farbe auf $c(e) = 0$ für alle $e \in E$.
3 Setze $v := s$.
4 **while** $S \neq \emptyset$ **do**
5 Entferne v aus der Wartemenge S.
6 **while** es gibt $e = (v, u) \in E$ mit $c(e) = 0$ **do**
7 Konstruiere einen Kreis aus unbenutzen Kanten:
8 Setze $c(e) = 1$ und $K = (v, e, u)$.
9 **while** $u \neq v$ **do**
10 Wähle $e' = (u, u') \in E$ mit $c(e') = 0$.
11 Setze $K := (K, e', u')$, $c(e') = 1$, $u := u'$.
12 **end while**
13 Füge den Kreis K an der Stelle v in P ein
14 **end while**
15 Sei v' der erste Knoten, der aktuell in P nach v kommt mit $v' \in S$. Setze $v := v'$.
16 **end while**

Output: Ein Eulerkreis P.

Wir stehen nun vor dem folgenden Problem: der Satz von Euler charakterisiert, wann es einen Eulerkreis gibt, aber er gibt keinen Weg an, wie man ihn findet. Im Beweis haben wir aber schon einen Algorithmus verwendet, den wir jetzt formalisieren wollen.

Der Algorithmus bestimmt Eulerkreise in gerichteten Graphen. Er funktioniert aber, mit leichten Modifikationen in der Implementierung, genauso für ungerichtete Graphen.

Für Eulerwege funktioniert der Algorithmus mit leichten Modifikationen auch. Man muss hierbei nur beachten, dass man als Startknoten s den Knoten mit $g^+(s) = g^-(s)+1$ wählt, und zwischen Zeile 3 und 4 des Algorithmus noch einen Block einfügen, in dem von s ausgehend ein Weg aus unbenutzten Kanten gesucht wird. Dieser ist dann kein Kreis, sondern ein Weg von s nach t. Danach werden auch hier nur noch Kreise eingefügt.

Die Korrektheit dieses Verfahren haben wir schon im Beweis von Satz 7.6 gezeigt. Daher beschränken wir uns hier auf den Satz und verzichten auf einen erneuten Beweis und illustrieren den Algorithmus statdessen an einen Beispiel, vergleiche Abbildung 7.5 und Tabelle 7.1. Im Beispiel lassen wir bei der Darstellung von P die Kanten weg, da es keine Parallelen gibt und deuten die Färbung der Kanten im Graphen an.

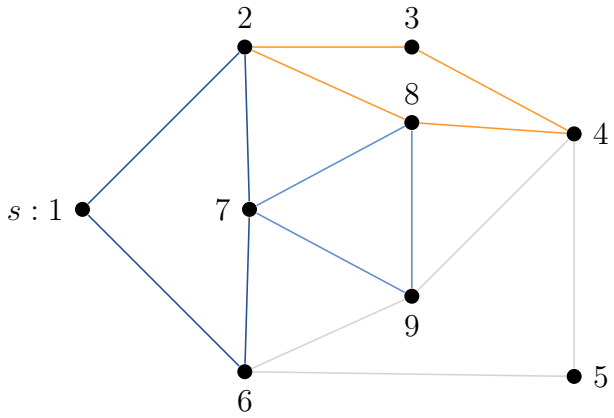


Abbildung 7.5: Illustration des Algorithmus von Hierholzer

Iter.	S	P	v
0	$\{1,2,3,4,5,6,7,8,9\}$	(1)	1
1	$\{2,3,4,5,6,7,8,9\}$	$(1,2,7,6,1)$	2
2	$\{3,4,5,6,7,8,9\}$	$(1,2,3,4,8,2,7,6,1)$	3
3	$\{4,5,6,7,8,9\}$	$(1,2,3,4,8,2,7,6,1)$	4
3	$\{5,6,7,8,9\}$	$(1,2,3,4,5,6,9,4,8,2,7,6,1)$	5
3	$\{6,7,8,9\}$	$(1,2,3,4,5,6,9,4,8,2,7,6,1)$	6
3	$\{7,8,9\}$	$(1,2,3,4,5,6,9,4,8,2,7,6,1)$	9
3	$\{7,8\}$	$(1,2,3,4,5,6,9,7,8,9,4,8,2,7,6,1)$	7
3	$\{8\}$	$(1,2,3,4,5,6,9,7,8,9,4,8,2,7,6,1)$	8
3	$\emptyset$	$(1,2,3,4,5,6,9,7,8,9,4,8,2,7,6,1)$	–

Tabelle 7.1: Illustration des Algorithmus von Hierholzer

Satz 7.10 *Es sei $G=(V,E)$ ein gerichteter eulerscher Graph. Dann ist Algorithmus 14 wohldefiniert und erzeugt einen Eulerkreis.*

7.3 Ausblick: Das Postboten-Problem

In diesem Abschnitt wollen wir uns noch einmal dem Problem des Postboten widmen. Wir stellen sein Revier als ungerichteten Graphen $G=(V,E)$ da, d.h wir betrachten die Straßen

als Kanten und die Kreuzungen als Knoten. Dann wissen wir, dass es genau dann einen Euler-Kreis gibt, wenn G zusammenhängend ist und alle Knoten geraden Grad haben. Wir wollen davon ausgehen, dass der Graph G zusammenhängend ist. Aber was soll der Postbote tun, wenn der Graph G nicht eulersch ist?

Um diese Frage zu beantworten, erweitern wir unser Modell: In der Realität sind nicht alle Straßen gleich lang. Daher sollten wir einen gewichteten Graphen betrachten, in dem alle Kanten mit der Länge der zugehörigen Straße gewichtet sind. In diesem ungerichteten gewichteten Graphen sucht der Postbote dann einen möglichst kurzen Kreis, der alle Kanten mindestens einmal enthält.

Gibt es einen Euler-Kreis, so sind die Kantengewichte unwichtig, da alle Euler-Kreise die Länge $\sum_{e \in E} w(e)$ haben. Müssen wir hingegen manche Kanten mehrfach durchlaufen, so sollten wir versuchen hierfür möglichst kurze Kanten zu wählen.

Probleme bei der Konstruktion einer Tour für den Postboten machen offensichtlich die Knoten mit ungeradem Grad. Um diese zu verlassen, muss der Postbote irgendwann eine schon benutzte Kante erneut verwenden. Ein möglicher Ansatz zur Lösung des Problems ist daher der folgende: Wir fügen zwischen Knoten mit ungeradem Grad künstlich neue Kanten ein.

Dazu betrachten wir den vollständigen Hilfsgraphen $G' = (V', E')$ mit $V' := \{v \in V \mid g(v) \text{ ist ungerade}\}$ und $E' = \{\{u, v\} \mid u, v \in V'\}$. In G' gewichten wir jede Kante $e = \{u, v\} \in E'$ mit der Länge des kürzesten Weges von u nach v im Graphen G. Da G zusammenhängend ist, existiert immer mindestens ein solcher Weg.

In dem Graphen G' können wir nun ein gewichtsminimales perfektes Matching M bestimmen. Da G' nach Korollar 1.14 eine gerade Anzahl von Knoten hat und vollständig ist, gibt es perfekte Matchings und folglich auch ein gewichtsminimales perfektes Matching. Wir haben im letzten Abschnitt nur Matchings in ungewichteten bipartiten Graphen bestimmt. Es gibt jedoch auch (deutlich kompliziertere) Algorithmen, die gewichtsminimale perfekte Matchings in allgemeinen Graphen bestimmen.

Für jede Kante $e = \{u, v\} \in M$ verdoppeln wir nun in G den kürzesten Weg von u nach v, d.h. wir fügen entlang dieses Weges Parallelen ein. Der so entstehende Graph $G'' = (V, E'')$ enthält den Graphen G als Teilgraphen. Außerdem haben alle Knoten geraden Grad, d.h. G'' ist eulersch. Jeder Euler-Kreis P in G'' ist dann eine optimale Tour für den Postboten.

7.4 Algorithmus von Fleury

Für ungerichtete Graphen wollen wir uns schließlich noch den Algorithmus von Fleury anschauen. Dieser hat gegenüber dem Algorithmus von Hierholzer den Vorteil, dass er den Euler-Kreis direkt findet und keine „kleineren“ Kreise mehr einbauen muss. Allerdings ist er im Allgemeinen langsamer als der Algorithmus von Hierholzer und wird daher inzwischen selten verwendet. Wie der Algorithmus von Hierholzer lässt sich auch der Algorithmus von Fleury modifizieren um Euler-Wege zu finden.

Für den Algorithmus von Fleury benötigen wir den Begriff einer Brücke, den wir in der folgenden Definition einführen wollen (siehe auch Abbildung 7.6).

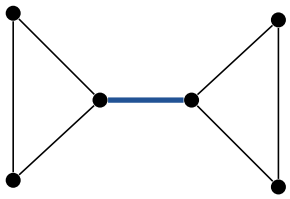


Abbildung 7.6: Eine Brücke

Definition 7.11 *Es sei $G = (V, E)$ ungerichteter zusammenhängender Graph. Eine Kante $e \in E$ heißt* Brücke*, wenn $G' := (V, E \setminus \{e\})$ nicht mehr zusammenhängend ist.*

Ob eine Kante $e \in E$ eine Brücke ist, kann man bestimmen, indem man e aus dem Graphen G entfernt und anschließend die Breiten- oder Tiefensuche mit einem beliebigen Startknoten durchführt. Findet diese Wege zu allen anderen Knoten, so ist e keine Brücke. Gibt es hingegen Knoten, zu denen kein Weg mehr gefunden wird, so hat man eine Brücke entfernt.

Unter Verwendung dieser Definition können wir jetzt den Algorithmus von Fleury definieren und zeigen, dass er einen Euler-Kreis bestimmt.

Algorithmus 15 Algorithmus von Fleury

Input: Ein ungerichteter eulerscher Graph $G = (V, E)$.

1 Wähle einen Startknoten $v_0 \in V$.
2 Initialisiere den Euler-Kreis $P := (v_0)$.
3 Setze $v := v_0$ und den Restgraphen $G_R = (V_R, E_R) := G$.
4 **while** $E_R \neq \emptyset$ **do**
5 Wähle eine Kante $e = \{v, w\} \in E$ mit Anfangsknoten v. Wähle hierbei nur dann eine Brücke, wenn es keine andere Kanten mit Anfangsknoten v mehr in E gibt.
6 Füge e und w an P an.
7 Entferne e und alle isolierten Knoten aus G_R.
8 Setze $v := w$.
9 **end while**

Output: Der Euler-Kreis P.

Satz 7.12 *Es sei $G = (V, E)$ ein ungerichteter eulerscher Graph. Dann ist Algorithmus 15 wohldefiniert und erzeugt einen Euler-Kreis.*

Beweis Nach Konstruktion erzeugt Algorithmus 15 einen Weg, der, wenn der Algorithmus korrekt terminiert, alle Kanten $e \in E$ genau einmal durchläuft und daher in einem eulerschen Graphen ein Euler-Kreis ist.

Wir müssen also nur zeigen, dass der Algorithmus terminiert. Dazu zeigen wir per Induktion über die Iterationen, dass in jeder Iteration der Restgraph G_R zusammenhängend ist und $v, v_0 \in V_R$ gilt. Dann ist es, solange $E_R \neq \emptyset$ gilt, immer möglich eine Kante mit Anfangsknoten v zu wählen. Folglich verringert sich die Zahl der Elemente von E_R in jeder Iteration um 1 und das Verfahren bricht korrekt ab.

In der ersten Iteration gilt $G_R = G$ und folglich $v = v_0 \in V_R$. Da G nach Voraussetzung eulersch ist, ist G und damit auch G_R zusammenhängend. Sei nun in einer Iteration G_R zusammenhängend mit $v \in V_R$. Dann ist $E_R \neq \emptyset$ und wir zeigen, dass dann auch in der in dieser Iteration erzeugte neue Restgraph $G'_R = (V'_R, E'_R)$ mit $E'_R = E_R \setminus \{e\}$, $e = \{v, w\}$ zusammenhängend ist und v_0 sowie w (das ja das neue v ist) enthält.

Fall 1: Entfernen wir in der Iteration eine Kante e, die keine Brücke ist, so ist der Graph $(V_R, E_R \setminus \{e\})$ nach Definition immer noch zusammenhängend und daher ist der neue Restgraph G'_R mit $V'_R = V_R$ und $E'_R = E_R \setminus \{e\}$ zusammenhängend und enthält v_0, w.

Fall 2: Entfernen wir hingegen in der Iteration eine Kante e, die Brücke ist, so sind alle Kanten in E_R, die von v ausgehen, Brücken.

Da G eulersch ist, haben alle Knoten in G geraden Grad. Dadurch, dass aus G ein zusammenhängender Weg entfernt wird, haben im Restgraphen G_R auch immer alle Knoten, außer möglicherweise v_0 und v, geraden Grad. Gilt $v = v_0$, so haben im Restgraphen G_R alle alle Knoten geraden Grad. Da der Restgraph zusammenhängend ist, enthält er folglich einen Euler-Kreis und deswegen keine Brücken. Daher muss $v \neq v_0$ gelten und beide Knoten haben im Restgraphen G_R ungeraden Grad.

Ist $g(v) = 1$, so ist $e = \{v, w\}$ die einzige Kante in G_R, die mit v inzidiert. Ist $E_R = \{e\}$, so folgt $w = v_0$ und mit dem Entfernen der Kante e terminiert der Algorithmus. Enthält E_R mehr als eine Kante, so zerfällt G_R durch das Entfernen der Kante e in den neuen zusammenhängenden Restgraphen $G'_R = (V_R \setminus \{v\}, E_R \setminus \{e\})$, der v_0 und w enthält, und den isolierten Knoten v, der entfernt wird.

Ist $g(v) > 1$, so folgt $g(v) \geq 3$. Wir zeigen, dass es dann auch eine Kante in E_R gibt, die mit v inzidiert und keine Brücke ist. Angenommen, dies ist nicht der Fall. Dann können wir zwei Brücken $e_1 = \{v, w_1\}$ und $e_2 = \{v, w_2\}$ aus G_R entfernen. Der entstehende Graph muss mindestens 3 Zusammenhangskomponenten haben, denn v und w_1 bzw. v und w_2 liegen jeweils in verschiedenen Zusammenhangskomponenten, da e_1 und e_2 Brücken sind. Daraus folgt auch, dass w_1 und w_2 nicht in der gleichen Zusammenhangskomponente liegen können. Wir bezeichnen die Zusammenhangskomponente, in der v liegt mit Z_0 und mit Z_1 bzw. Z_2 die Zusammenhangskomponenten, die w_1 bzw. w_2 enthalten. Jede der Zusammenhangskomponenten für sich ist ein Graph und muss daher eine gerade Anzahl von Knoten mit ungeradem Grad enthalten. In Z_1 hat der Knoten w_1 ungeraden Grad, es sei denn es gilt $w_1 = v_0$. Daher muss Z_1 noch mindestens einen anderen Knoten mit ungeradem Grad enthalten. Da v und w_2 nicht in Z_1 liegen, kommt nur noch v_0 in Frage, d.h. es muss $v_0 \in Z_1$ gelten. Analog folgt aber $v_0 \in Z_2$, ein Widerspruch, da die Zusammenhangskomponenten Z_1 und Z_2 keine gemeinsamen Knoten haben. □

7.5 Aufgaben

Aufgabe 7.1 Welche der Graphen in Abbildung 7.7 sind eulersch? Welche enthalten zu-

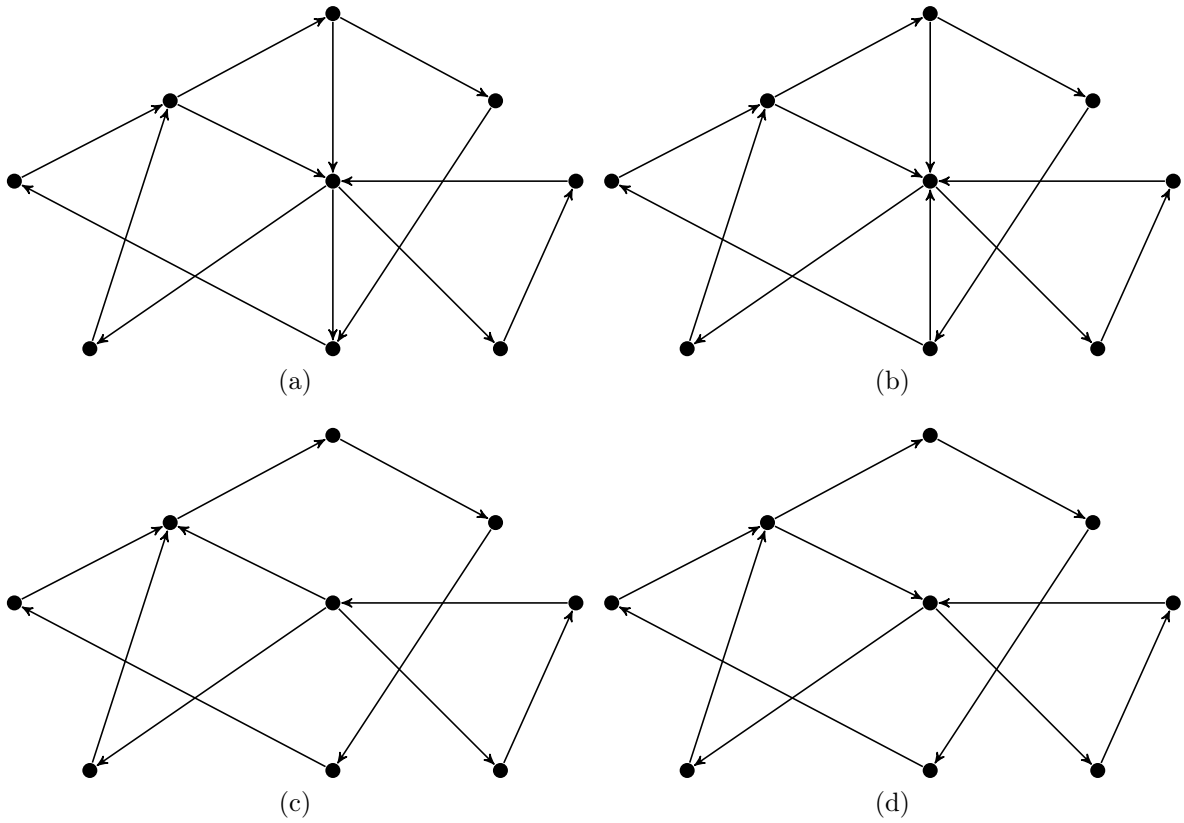


Abbildung 7.7: Sind die Graphen eulersch?

mindest einen Euler-Weg?

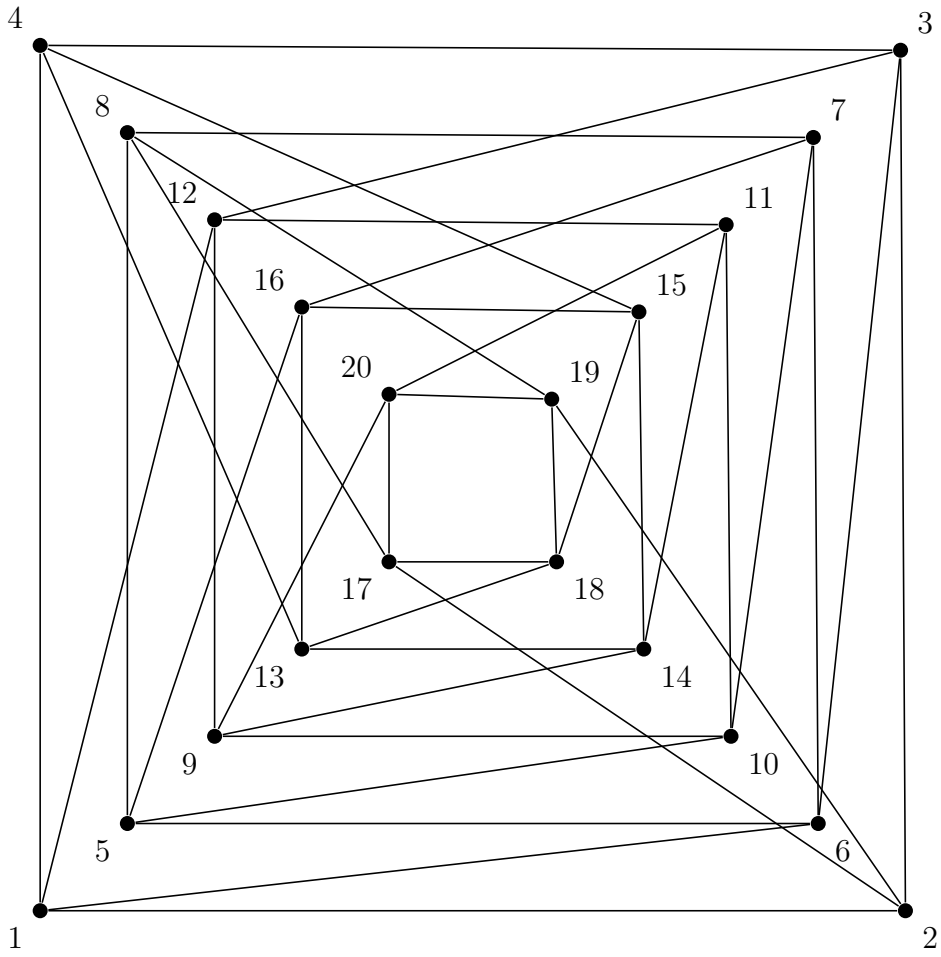


Abbildung 7.8: Wie sieht ein Euler-Kreis aus?

Aufgabe 7.2 Bestimmen Sie in dem Graph aus Abbildung 7.8 einen Euler-Kreis mit dem Algorithmus von Hierholzer.

Aufgabe 7.3

(a) Für welche $n, m \in \mathbb{N}$ ist der vollständige bipartite Graph $K_{n,m}$ eulersch?

(b) Zeigen Sie, dass ein eulerscher bipartiter Graph immer eine grade Anzahl von Kanten hat.

Aufgabe 7.4 Bestimmen Sie eine optimale Tour für einen Postboten, dessen Revier durch den Graphen in Abbildung 7.9 beschrieben wird.

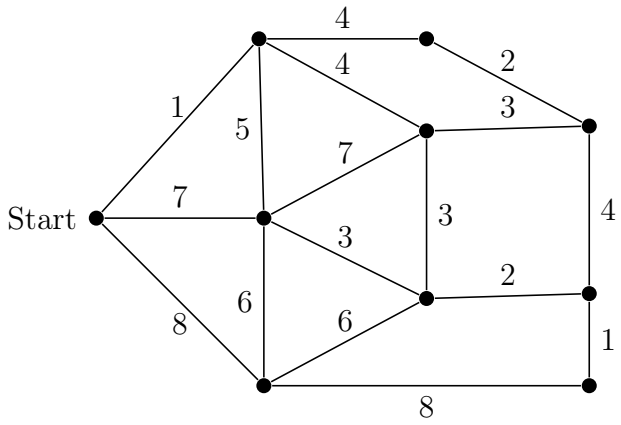


Abbildung 7.9: Welchen Weg sollte der Postbote nehmen?

Aufgabe 7.5 In einem Dominospiel gibt es zu jedem Zahlenpaar (a, b) mit $a, b \in \{0, 1, \ldots, 6\}$ einen Stein, insgesamt also 28 Steine. Man darf zwei Steine nur dann aneinander legen, wenn die sich berührenden Enden die gleiche Anzahl Augen haben.

(a) Ist es möglich so aus allen Steinen einen Kreis zu legen?

(b) Nun sortieren wir die Steine (a, b) aus, für die $a, b \in \{0, 1, 2, 3\}$ gilt. Ist es möglich aus diesen Steinen einen Kreis zu legen?

(c) Betrachten wir schließlich ein Dominospiel für Mathematiker: Es sei $k \geq 5$ eine beliebige ungerade Zahl und es gebe zu jedem Zahlenpaar (a, b) mit $a, b \in \{0, 1, 2, \ldots, k\}$ einen Stein. Wenn wir nun die Steine $(0, 1), (2, 3), \ldots, (k-1, k)$ aus dem Spiel entfernen, ist es dann möglich einen Kreis aus den verbleibenden Steinen zu legen?

8 Rundreisen

Zu Beginn hatten wir den Ägyptologen Petrie kennengelernt, vergleiche auch [3]. Sein Problem wollen wir in diesem Abschnitt genauer betrachten.

Beispiel 8.1 (Archäologie) Petrie hatte es sich zum Ziel gesetzt, die ca. 3000 in Naqada gefunden Gräber chronologisch anzuordnen. Sein Ansatz hierfür ist ein Vorläufer der heute noch in der Archäologie gebräuchlichen Seriation. Allerdings führte er diesen Vorgang damals von Hand durch, während heute Computer eingesetzt werden.

Petrie teilte die in den Gräbern gefundene Keramik in 9 Kategorien (teilweise mit Unterkategorien) auf. Die einzelnen Kategorien unterschieden sich durch die Form der Keramiken, die Glasur, die Existenz von Henkeln, die Farbe,.... Anschließend ordnete er jedem Grab einen Vektor (in seinem Fall ein Papierstreifen) zu, in dem komponentenweise jeweils eine 1 stand, falls das Grab Keramiken aus der zugehörigen Kategorie enthielt, und sonst eine 0. Basierend auf diesen Vektoren definierte er den Abstand zweier Gräber als die Anzahl der unterschiedlichen Einträge, d.h. für zwei Gräber mit zugehörigen Vektoren u, v betrachtete er

$$\sum_{i=1}^{9} |u_i - v_i|$$

als den Abstand[1].

Petrie nahm an, dass zwei Gräber mit kleiner Distanz, also ähnlichen Keramikbeigaben, auch zeitlich in kurzem Abstand entstanden sind. Daher versuchte er, eine Reihenfolge zu finden, in der für alle aufeinander folgenden Gräber der Abstand möglichst klein ist. ♢

Dies ist eine archäologische Fassung eines der bekanntesten Probleme in der Graphentheorie ist das Problem des Handlungsreisenden, auch Traveling Salesman Problem (TSP) genannt.

Beispiel 8.2 (Problem des Handlungsreisenden) Ein Vertreter möchte seine Kunden besuchen, die in verschiedenen Städten wohnen. Hierbei startet er von der Stadt, in der er wohnt und will auch dort wieder ankommen. Gesucht ist also eine Rundreise durch alle Städte, in der er Kunden hat, zuzüglich seiner Heimatstadt, auf der jede Stadt nur einmal besucht wird. Da Zeit Geld ist, soll die Rundreise außerdem möglichst kurz sein. ♢

Dieses Problem hat eine Vielzahl von Anwendungen, einige davon im Bereich der Biologie.

Beispiel 8.3 (Routenplanung bei Hummeln) Eine Hummel steht vor dem gleichen Problem wir der Handlungsreisende. Sie startet ihre Runde am Nest und will dort auch

[1] Ein ähnlicher Abstandsbegriff, die Hamming-Distanz, findet sich heute noch in der Informatik und wird zur Erkennung von Übertragungsfehlern verwendet.

wieder ankommen. Unterwegs sucht sie bekannte Stellen auf, an denen sie Pollen sammelt. Liegen diese Stellen nah beieinander, so wählt die Hummel meistens eine Route, auf der der Abstand zwischen zwei Sammelstellen möglichst klein ist. Liegen die Sammelstellen jedoch weiter auseinander, so beginnen Hummeln die Route so zu optimieren, dass die Gesamtstrecke möglichst klein wird, vergleiche [8]. ◊

Beispiel 8.4 (DNS-Sequenzierung) Bei der DNS-Sequenzierung möchte man die Abfolge der Nukleotide in einem DNS-Strang bestimmen. Eine Möglichkeit dies zu tun, ist den Strang in viele kleine Teile zu zerschneiden und diese getrennt zu analysieren. Dann stellt sich aber die Frage: Wie fügt man die vielen Teilsequenzen wieder zu dem ursprünglichen Strang zusammen? Hier kann man die Tatsache nutzen, dass man im Allgemeinen nicht einen einzigen DNS-Strang zerschneidet, sondern viele gleiche. Da diese an zufälligen Stellen aufgebrochen werden, hat man am Ende nicht die gleichen kurzen Sequenzen mehrfach, sondern viele verschiedene Teilsequenzen, die sich überlappen. Man kann deshalb versuchen, eine möglichst kurze Sequenz zu finden, die alle auftretenden Teilsequenzen enthält. Man versucht also eine Reihenfolge der Teilsequenzen zu finden, so dass der Teil einer Sequenz, der nicht in der vorherigen enthalten ist, möglichst klein wird, d.h. die Überschneidung zweier aufeinander folgender Sequenzen möglichst groß ist. ◊

8.1 Hamiltonwege und -kreise

Etwas vereinfacht betrachtet man hier die folgende Problemstellung.

Definition 8.5 *Es sei $G = (V, E)$ ein gerichteter oder ungerichteter Graph. Ein Weg P in G heißt* Hamilton-Weg*, wenn er jeden Knoten in V genau einmal durchläuft. Ist P zusätzlich ein Kreis, so nennt man P einen* Hamilton-Kreis*. Der Graph G heißt* hamiltonsch*, wenn er einen Hamilton-Kreis enthält.*

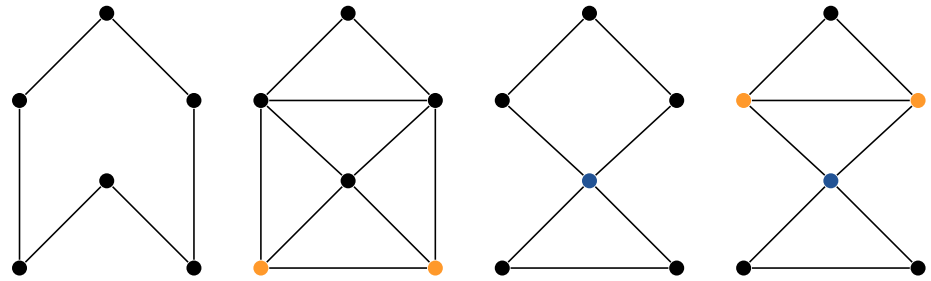
In gewichteten Graphen kann man zusätzlich die Frage nach einem kürzesten Hamilton-Kreis stellen.

Im Gegensatz zur Bestimmung von Euler-Kreisen und -Wegen ist die Bestimmung von Hamilton-Kreisen oder -Wegen ungleich schwerer und es gibt keine „schöne“ Charakterisierung von hamiltonschen Graphen.

Obwohl die Hamilton-Kreise und Euler-Kreise verwandt zu sein scheinen, gibt es hier keinen Zusammenhang, vergleiche Abbildung 8.1.
Wir werden uns daher in diesem Kapitel auf den einfacheren Fall der ungerichteten Graphen einschränken und uns damit begnügen, notwendige und hinreichende Bedingungen für die Existenz von Hamilton-Kreisen herzuleiten.

Zwei offensichtliche notwendige Bedingungen sind:

- Ist $G = (V, E)$ ungerichtet und hamiltonsch, so enthält V keinen Knoten v vom Grad $g(v) = 1$.



(a) eulersch und hamiltonsch (b) nicht eulersch, aber hamiltosch (c) eulersch, aber nicht hamiltonsch (d) weder hamiltonsch noch eulersch

Abbildung 8.1: Zusammenhang zwischen eulerschen und hamiltonschen Graphen?

- Ist $G = (V, E)$ ungerichtet und hamiltonsch, so ist G zusammenhängend.

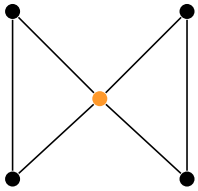


Abbildung 8.2: Ein zusammenhängender, nicht hamiltonscher Graph mit $g(v) \geq 1$ für alle Knoten $v \in V$

Diese beiden Bedingungen sind jedoch, wie Abbildung 8.2 zeigt, weit davon entfernt hinreichend zu sein. Eine stärkere notwendige Bedingung ist die folgende.

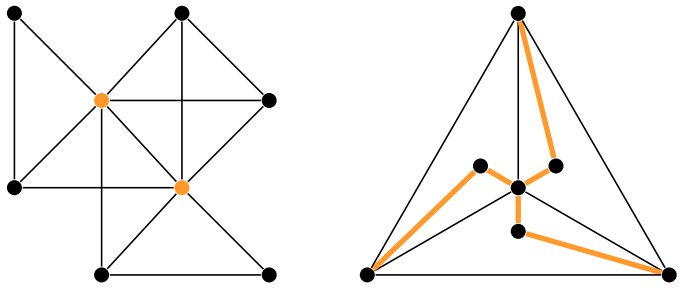
Satz 8.6 *Es sei $G = (V, E)$ ein ungerichteter hamiltonscher Graph. Dann gilt: Werden $k \leq |V|$ Knoten mit den zugehörigen Kanten aus dem Graphen G entfernt, so besteht der Restgraph aus höchstens k Zusammenhangskomponenten.*

Beweis Nach Voraussetzung gibt es in G einen Hamilton-Kreis

$$P = (v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n = v_0)$$

mit $n = |V|$, der alle Knoten $v \in V$ enthält. Nun betrachten wir den Graphen $G' = (V, E')$, der nur die Kanten aus dem Hamilton-Kreis enthält. Entfernen wir k Knoten mit den zugehörigen Kanten aus G', so zerfällt G' in höchstens k Zusammenhangskomponenten. Fügen wir anschließend wieder alle noch fehlenden Kanten in E hinzu, deren Endknoten nicht gelöscht wurden, so werden eventuell manche der Zusammenhangskomponenten wieder miteinander verbunden. Es entstehen aber keinesfalls neue Zusammenhangskomponenten, d.h die Zahl der Zusammenhangskomponenten wird nicht größer als k. □

Mit Satz 8.6 kann man leicht verifizieren, dass der Graph in Abbildung 8.3(a) nicht hamiltonsch ist. Entfernt man die 2 orange markierten Knoten und alle zugehörigen Kanten, so



(a) nach Satz 8.6 nicht hamiltonsch

(b) nicht hamiltonsch, obwohl die Bedingung aus Satz 8.6 erfüllt ist

Abbildung 8.3: Anwendungen von Satz 8.6

zerfällt der Restgraph in 3 Zusammenhangskomponenten. Betrachtet man andererseits den Graphen aus Abbildung 8.3(b), so kann man sich leicht überlegen (vergleiche Aufgabe 2), dass hier die Bedingung aus Satz 8.6 erfüllt ist. Dennoch ist der Graph nicht hamiltonsch, denn jeder Hamilton-Kreis müsste die orange markierten Kanten enthalten, würde dann aber den mittleren Knoten mindestens zweimal enthalten.

Da wir nun gesehen haben, dass die obigen Bedingungen nicht ausreichend sind um zu grantieren, dass ein ungerichteter Graph einen Hamilton-Kreis besitzt, wollen wir uns noch ein paar hinreichende Bedingungen anschauen. Ein hinreichende, aber auch extrem starke Bedingung basiert auf der Vollständigkeit des Graphen:

- Es sei $G = (V, E)$ ein ungerichteter und vollständiger Graph. dann ist G hamiltonsch.

Diese Bedingung ist zwar offensichtlich ausreichend für die Existenz eines Hamilton-Kreises, aber auch sehr einschränkend. Daher wollen wir uns im folgenden noch zwei hinreichende Bedingungen anschauen, die – grob gesagt – auch darauf basieren, dass der Graph G ausreichend viele Kanten haben muss.

In diesen Resultaten müssen wir voraussetzen, dass der betrachtete Graph einfach ist, da es sonst nicht möglich ist von eine Knotengrad Rückschlüsse auf die Anzahl der Nachbarn eines Knoten zu ziehen. In einen ungerichteten Graphen hängt die Existenz eines Hamilton-Kreises aber offensichtlich nicht davon ab, ob eine Kante Parallelen besitzt oder nicht. Wir können die folgenden beiden Resultate daher auch verwenden, um die Existenz eines Hamilton-Kreises in einem nicht einfachen Graphen zu zeigen, wenn wir sie, statt auf den Graphen selbst, auf den zugehörigen einfachen Graphen anwenden.

Satz 8.7 *Es sei $G = (V, E)$ ein einfacher ungerichteter Graph mit $|V| \geq 3$ und $u, v \in V$ zwei nichtadjazente Knoten mit $g(u) + g(v) \geq |V|$. Dann ist G hamiltonsch genau dann, wenn $G' := (V, E \cup \{u, v\})$ hamiltonsch ist.*

Beweis $\Longrightarrow$ Besitzt G einen Hamilton-Kreis P, so ist P auch ein Hamilton-Kreis in G'.

$\Longleftarrow$ Es sei $P = (v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n = v_0)$ mit $n = |V|$ ein Hamilton-Kreis in G', d.h. er enthält *alle* Knoten $v \in V$. Verwendet P die Kante $\{u, v\}$ nicht, so ist P auch

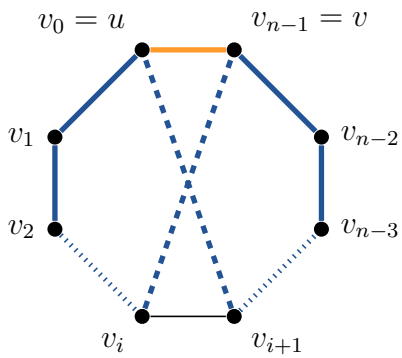


Abbildung 8.4: Illustration des Beweises von Satz 8.7

ein Hamilton-Kreis in G. Nehmen wir nun an, dass P die Kante $\{u, v\}$ enthält. Durch geeignetes „Drehen“ des Kreises können wir annehmen, dass $v_0 = u$ und $v_{n-1} = v$ gilt. Dann betrachten wir die Mengen

$$A := \{i \mid \{v, v_i\} \in E, i = 1, \ldots, n-3\} \quad \text{und} \quad B := \{i \mid \{u, v_{i+1}\} \in E, i = 1, \ldots, n-3\}.$$

Dann folgt $|A| \geq d(v) - 1$, da nach Voraussetzung $v = v_{n-1}$ und $u = v_0$ nicht adjazent sind und folglich alle Nachbarn von v in der Menge $\{v_1, \ldots, v_{n-2}\}$ enthalten sind. Genauso erhält man $|B| \geq d(u) - 1$. Folglich gilt $|A| + |B| \geq d(u) + d(v) - 2 \geq n - 2$. Andererseits ist $A, B \subseteq \{1, \ldots, n-3\}$. Daher gibt es mindestens ein $i \in A \cap B$, d.h. einen Knoten v_i mit $\{v, v_i\} \in E$ und $\{u, v_{i+1}\} \in E$. Dann ist

$$P' := (u = v_0, v_1, v_1, \ldots, e_{i-1}, v_i, \{v_i, v\}, v = v_{n-1}, e_{n-1}, v_{n-2}, e_{n-2}, \ldots, v_{i+1}, \{v_{i+1}, u\}, u)$$

ein Hamilton-Kreis in G. □

Etwas leichter zu überprüfen ist die folgende Bedingung.

Korollar 8.8 *Es sei $G = (V, E)$ ein einfacher ungerichteter Graph mit $|V| \geq 3$ und $g(v) \geq |V|/2$ für alle $v \in V$. Dann ist G hamiltonsch.*

Beweis Für ein beliebiges Paar nichtadjazenter Knoten $u, v \in V$ gilt $g(u) + g(v) \geq |V|$, d.h. der Graph G ist genau dann hamiltonsch, wenn der um die Kante $\{u, v\}$ erweiterte Graph hamiltonsch ist. In diesem gilt wieder für alle Paare nicht adjazenter Knoten obige Gradbedingung, so dass wir G solange erweitern können, bis wir bei dem vollständigen Graphen mit $|V|$ Knoten angelangt sind. Folglich folglich ist G genau dann hamiltonsch, wenn der vollständige Graph hamiltonsch ist, d.h. G ist hamiltonsch. □

8.2 Problem des Handlungsreisenden

Kommen wir nun wieder zum Problem des Handlungsreisenden zurück: Mit den Begriffen, die wir bisher eingeführt haben, lässt sich dieses Problem wie folgt beschreiben: Gegeben

sei ein ungerichteter, vollständiger und gewichteter Graph. Dann wird ein Hamilton-Kreis minimaler Länge gesucht. Da der Graph vollständig ist, gibt es einen solchen immer.

Wie wir aus dem letzten Abschnitt wissen, ist es schon sehr schwer Hamilton-Kreise zu finden. Noch schwerer ist es dann natürlich unter den $(|V|-1)!/2$ Hamilton-Kreis zu finden. Daher werden wir in diesem Abschnitt sogenannte *Heuristiken* zur Lösung des Problems des Handlungsreisenden betrachten. Unter einer Heuristik versteht man ein Verfahren zur Lösung eines Optimierungsproblems, dass versucht mit Hilfe von „Faustregeln“ und „intelligentem Raten“ eine gute Lösung zu finden, aber nicht garantieren kann eine optimale Lösung zu finden.

Eine der einfachsten Heuristiken zur Lösung des Problems des Handlungsreisenden ist die sogenannte Nächster-Nachbar-Heuristik, vergleiche Algorithmus 16.

Algorithmus 16 Nächster-Nachbar-Heuristik

Input: Ein ungerichteter, vollständiger und gewichteter Graph $G = (V, E)$ mit Kantengewichten $w(e)$ für alle $e \in E$.

1 Wähle einen beliebigen Startknoten $s \in V$ und setze $P = (s)$.
2 Setze $v = s$.
3 **for** $k = 1$ **to** $|V| - 1$ **do**
4 Wähle aus allen Kanten $e = \{v, u\} \in E$ diejenige Kante $e^* = \{v, u^*\}$ mit minimalem Gewicht $w(e^*)$, so dass $u^* \notin P$ gilt.
5 Füge (e^*, u^*) hinten an P an.
6 Setze $v = u^*$.
7 **end for**
8 Schließe den Kreis mit der Kante $e^* = \{v, s\} \in E$ minimalen Gewichts.

Output: Einen Hamiltonkreis P.

Satz 8.9 *Es sei $G = (V, E)$ ein ungerichteter, vollständiger und gewichteter Graph mit Kantengewichten $w(e)$ für alle $e \in E$. Dann findet Algorithmus 16 eine Hamilton-Kreis in G.*

Beweis Da G vollständig ist, gibt es in jeder Iteration mindestens eine Kante zu einem noch nicht besuchten Knoten und folglich auch eine mit minimalem Gewicht. Daher ist die Wahl von e^* wohldefiniert. Nach $|V| - 1$ Iterationen enthält P nach Konstruktion $|V|$ verschiedene Knoten, also alle Knoten in V. Da G vollständig ist, existiert eine Kante vom letzten Knoten zum Startknoten und der geschlossene Kreis P ist ein Hamilton-Kreis in G. □

Algorithmus 16 findest zwar immer einen Hamilton-Kreis, aber nicht immer den kürzesten. Tatsächlich kann die Länge des gefundenen Hamilton-Kreises sogar beliebig viel größer sein als die des kürzesten, vergleiche Abbildung 8.5. Wählt man hier $C > 1$, so ist der rot eingezeichnete Kreis der von der Nächster-Nachbar-Heuristik mit Startknoten $s = 1$ gefundene Hamilton-Kreis mit der Länge $5 + C$. Nutzt man die Diagonalen, so ist es jedoch auch möglich eine Hamilton-Kreis der Länge 8 zu finden. Für $C > 3$ ist der von

der Heuristik gefundene Kreis also nicht optimal, kann sogar beliebig viel länger das der optimale werden.

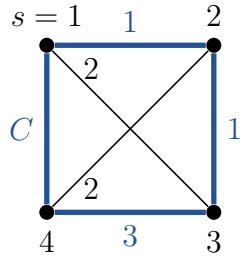


Abbildung 8.5: Beispiel für die Nächster-Nachbar-Heuristik

Eine weitere Heuristik, bei der man abschätzen kann, wieviel schlechter der gefundene Hamilton-Kreis im Vergleich zum optimalen höchstens sein kann, beruht auf der Konstruktion minimaler spannender Bäume. Diese funktioniert allerdings nur in sogenannten metrischen Graphen.

Definition 8.10 *Es sei $G = (V, E)$ ein ungerichteter gewichteter Graph mit positiven Kantengewichten $w(e) > 0$ für alle $e \in E$. Der Graph G heißt* metrisch*, wenn für alle $u, v, z \in V$ mit $\{u, v\}, \{u, z\}, \{z, v\} \in E$ die Dreiecksungleichung*

$$w(\{u, v\}) \leq w(\{u, z\}) + w(\{z, v\})$$

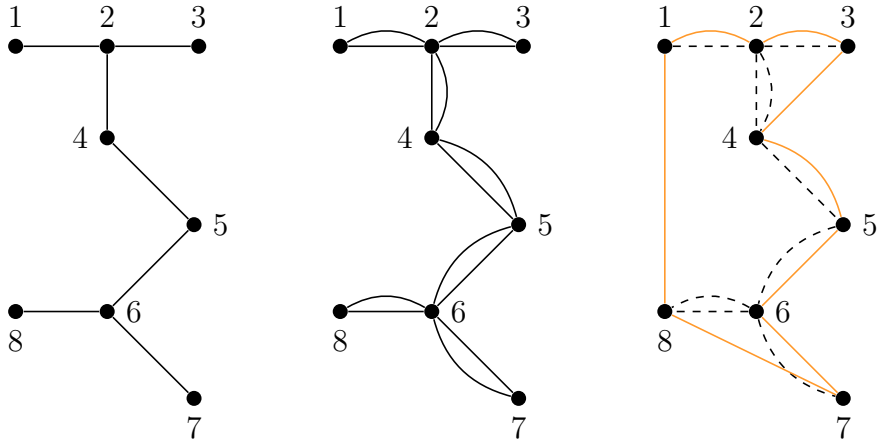
gilt.

Was hat nun in einem ungerichteten, vollständigen und metrischen Graphen die Länge des kürzesten Hamilton-Kreises P_H^* mit dem Gewicht eines minimalen spannenden Baums T^* zu tun? Da P_H^* ein Hamilton-Kreis ist, ist P_H^* ohne die schwerste Kante $e^* \in P_H^*$ ein (entarteter) spannender Baum. Es gilt also $w(P_H^*) - w(e^*) \geq w(T^*)$. Da P_H^* genau $|V|$ Kanten enthält und e^* die schwerste davon ist, folgt $w(e^*) \geq w(P_H^*)/|V|$. Damit erhalten wir

$$w(T^*) \leq w(P_H^*) - w(e^*) \leq w(P_H^*) - \tfrac{w(P_H^*)}{|V|} \leq w(P_H^*)\big(1 - \tfrac{1}{|V|}\big).$$

Umgekehrt lässt sich in einem vollständigen Graphen aus einem beliebigen spannenden Baum auch ein Hamilton-Kreis erzeugen. Wir verwenden hier T^*, siehe Abbildung 8.6(b). Dafür verdoppeln wir in T^* jede Kante, d.h wir fügen zu jeder Kante in T^* eine Parallele ein, vergleiche Abbildung 8.6(b). Durch das Verdoppeln der Kanten in dem zusammenhängenden Graph T^* entsteht ein ebenfalls zusammenhängender Graph, in dem jeder Knoten geraden Grad hat. Folglich besitzt dieser Graph einen Euler-Kreis P_E, den wir zum Beispiel mit dem Algorithmus von Hierholzer bestimmen können. Der Euler-Kreis hat dann die Länge $2w(T^*)$, da er jede Kante genau einmal durchläuft. Aus dem Euler-Kreis können wir einen Hamilton-Kreis P_H wie folgt erzeugen: Wähle einen beliebigen Startknoten auf dem Euler-Kreis und folge diesem so lange, bis ein Knoten zum zweiten Mal besucht würde. Da der Graph G vollständig ist, ist es möglich stattdessen direkt zu dem nächsten

unbesuchten Knoten auf dem Euler-Kreis weiterzugehen. Man kürzt also im Vergleich zu dem Euler-Kreis ab, vergleiche Abbildung 8.6(c). Dadurch werden Teile des Euler-Kreises durch eine einzelne neue Kante ersetzt, die auf Grund der Dreiecksungleichung aber nicht länger als der ersetzte Teil des Euler-Kreises ein kann. Schließlich erhält man so einen Hamilton-Kreis P_H mit $w(P_H) \leq w(P_E) = 2w(T^*)$.



(a) minimaler spannender Baum T^*

(b) Verdoppeln der Kanten liefert einen Euler-Kreis P_E

(c) Abkürzen liefert einen Hamilton-Kreis P_H

Abbildung 8.6: Illustration der Minimaler-spannender-Baum-Heuristik

Insgesamt besteht also der folgende Zusammenhang zwischen der Länge des kürzesten Hamilton-Kreises und dem Gewicht eines minimalen spannenden Baums.

Lemma 8.11 *Es sei $G = (V, E)$ ein ungerichteter, vollständiger und metrischer Graph, P_H^* ein minimaler Hamilton-Kreis und T^* ein minimaler spannender Baum. Dann gilt*

$$\frac{|V|}{|V|-1} \cdot w(T^*) \leq w(P_H^*) \leq 2 \cdot w(T^*).$$

Gleichzeitig haben wir die sogenannte Minimaler-spannender-Baum-Heuristik, siehe Algorithmus 17, hergeleitet und den folgenden Satz 8.12 bewiesen.

Satz 8.12 *Es sei $G = (V, E)$ ein ungerichteter, vollständiger und metrischer Graph mit Kantengewichten $w(e) > 0$ für alle $e \in E$. Dann bestimmt Algorithmus 17 einen Hamilton-Kreis, welcher höchstens $2(1 - \frac{1}{|V|})$ mal so lang wie der kürzeste ist.*

Dieses Verfahren lässt sich weiter verbessern, wenn wir Matchings nutzen und einen ähnlichen Trick wie bei der Bestimmung der optimalen Route für einen Postboten verwenden. Sei also $G = (V, E)$ wieder ein ungerichteter, vollständiger und metrischer Graph mit Kantengewichten $w(e) > 0$ für alle $e \in E$ und $T^* = (V, E^*)$ ein minimaler spannender Baum von G.

Algorithmus 17 Minimaler-spannender-Baum-Heuristik

Input: Ein ungerichteter, vollständiger und metrischer Graph $G = (V, E)$ mit Kantengewichten $w(e) > 0$ für alle $e \in E$.

1 Bestimme einen minimalen spannenden Baum T^* (zum Beispiel mit dem Algorithmus von Prim oder Kruskal).
2 Verdopple in T^* alle Kanten.
3 Bestimme in dem entstehenden Graphen einen Euler-Kreis P_E (zum Beispiel mit dem Algorithmus von Hierholzer).
4 Erzeuge aus dem Euler-Kreis einen Hamilton-Kreis P_H durch Überspringen bereits besuchter Knoten.

Output: Ein Hamilton-Kreis P_H.

Dann bezeichnen wir mit $U \subseteq V$ die Menge aller Knoten, die in T^* ungeraden Grad haben. Da die Menge U eine gerade Anzahl von Elementen haben muss, existiert in dem vollständigen Graphen mit Knotenmenge U ein perfektes Matching. Folglich gibt es auch ein perfektes Matching M^* mit minimalem Gewicht.

In dem Graphen $G^* = (V, E^* \cup M^*)$ haben dann alle Knoten geraden Grad und G^* ist zusammenhängend. Daher besitzt G^* einen Euler-Kreis P_E mit $w(P_E) = w(T^*) + w(M^*)$, den wir wie bei der Minimaler-spannender-Baum-Heuristik zu einem Hamilton-Kreis P_H mit $w(P_H) \leq w(P_E) = w(T^*) + w(M^*)$ abkürzen können.

Wir wissen schon, dass $w(T^*) \leq \frac{|V|-1}{|V|} w(P_H^*)$ gilt, wobei P_H^* wieder einen kürzesten Hamilton-Kreis bezeichne. Aber wie hängt $w(M^*)$ mit $w(P_H^*)$ zusammen? Durch Abkürzen können wir aus dem Hamilton-Kreis P_H^* in G einen Hamilton-Kreis P_U in dem vollständigen Graphen mit Knotenmenge U machen. Da U eine gerade Anzahl von Elementen hat, ist $P_U = (v_0, v_1, \ldots, v_{2k} = v_0)$ ein Kreis gerader Länge. Wir können also die Kanten in P_U wie folgt in zwei perfekte Matchings M_1, M_2 zerlegen:

$$\begin{aligned} M_1 &= \{\{v_0, v_1\}, \{v_2, v_3\}, \ldots, \{v_{2k-2}, v_{2_k-1}\}\}, \\ M_2 &= \{\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{2k-1}, v_{2k}\}\}. \end{aligned}$$

Es gilt dann

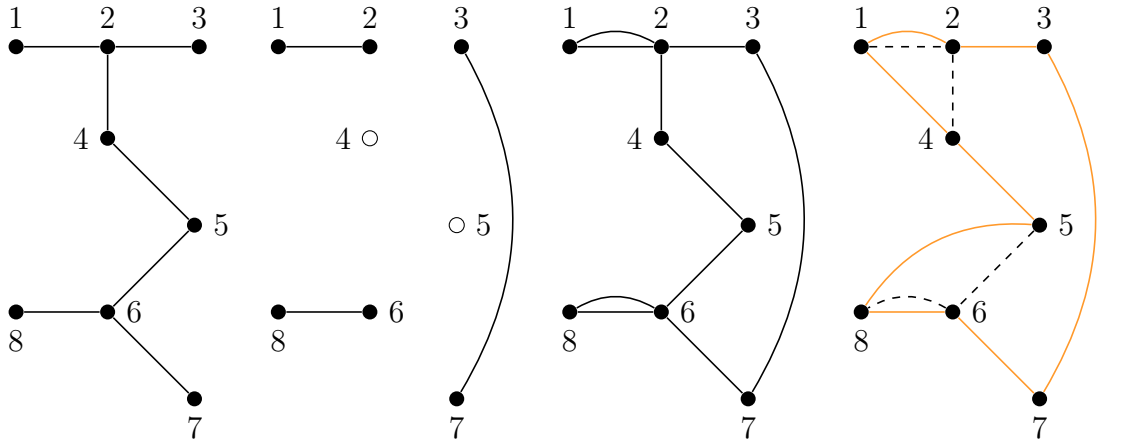
$$w(M_1) + w(M_2) = w(P_U) \leq w(P_H^*) \quad \Longrightarrow \quad \min\{w(M_1), w(M_2)\} \leq \tfrac{w(P_H^*)}{2}.$$

Da M^* ein gewichtsminimales perfektes Matching in dem vollständigen Graphen mit Knotenmenge U ist, gilt dann auch $w(M^*) \leq \frac{w(P_H^*)}{2}$ und folglich

$$w(P_H) \leq w(T^*) + w(M^*) \leq \tfrac{|V|-1}{|V|} w(P_H^*) + \tfrac{w(P_H^*)}{2} \leq \tfrac{3}{2} w(P_H^*).$$

Zur Erinnerung: Die Minimaler-spannender-Baum-Heuristik hatte einen Hamilton-Kreis P_H mit $w(P_H) \leq 2w(P_H^*)$ erzeugt.

Damit haben wir die, nach ihren Erfinder benannte, Christofides-Heuristik hergeleitet, vergleiche Algorithmus 18 und Satz 8.13 bewiesen.



(a) minimaler spannender Baum T^*

(b) gewichtsminimales perfektes Matching M^*

(c) Vereinigung von T^* und M^* liefert einen Euler-Kreis P_E

(d) Abkürzen liefert einen Hamilton-Kreis P_H

Abbildung 8.7: Illustration der Christofides-Heuristik

Algorithmus 18 Christofides-Heuristik

Input: Ein ungerichteter, vollständiger und metrischer Graph $G = (V, E)$ mit Kantengewichten $w(e) > 0$ für alle $e \in E$.

1 Bestimme einen minimalen spannenden Baum T^* (zum Beispiel mit dem Algorithmus von Prim oder Kruskal).

2 Bestimme die Menge $U \subseteq V$ aller Knoten mit ungeradem Grad in T^*.

3 Bestimme ein perfektes Matching M^* mit minimalem Gewicht in dem vollständigen Graphen mit Knotenmenge U.

4 Bestimme in dem Graphen $T^* \cup M^*$ einen Euler-Kreis P_E (zum Beispiel mit dem Algorithmus von Hierholzer).

5 Erzeuge aus dem Euler-Kreis einen Hamilton-Kreis P_H durch Überspringen bereits besuchter Knoten.

Output: Ein Hamilton-Kreis P_H.

Satz 8.13 *Es sei $G = (V, E)$ ein ungerichteter, vollständiger und metrischer Graph mit Kantengewichten $w(e) > 0$ für alle $e \in E$. Dann bestimmt Algorithmus 18 einen Hamilton-Kreis, welcher höchstens $\frac{3}{2}(1 - \frac{1}{|V|})$ mal so lang wie der kürzeste ist.*

8.3 Aufgaben

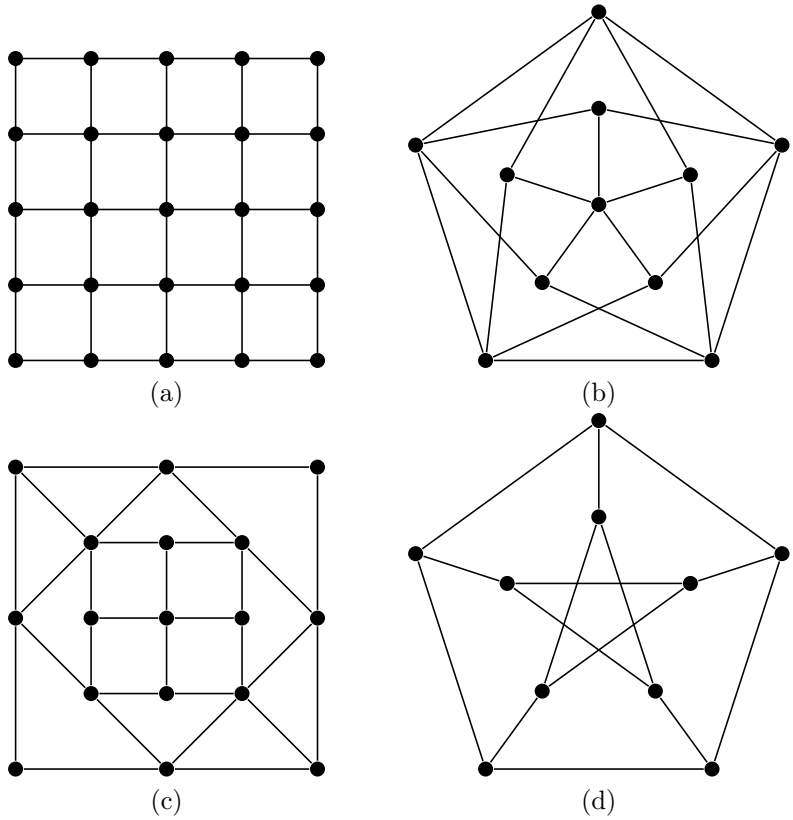


Abbildung 8.8: Welche der Graphen sind hamiltonsch?

Aufgabe 8.1 Welche der Graphen in Abbildung 8.8 sind hamiltonsch? Begründen Sie Ihre Antworten.

Aufgabe 8.2 Zeigen Sie, dass der Graph aus Abbildung 8.9 der Bedingung aus Satz 8.6 genügt.

Aufgabe 8.3 Finden Sie zwei ungerichtete, einfache und zusammenhängende Graphen mit gleicher Knotenzahl und gleichen Knotengraden, vor denen einer hamiltonsch ist und der andere nicht.

Aufgabe 8.4 Kann man auf einem 4×4–Schachbrett, vergleiche Abbildung 8.10, einen

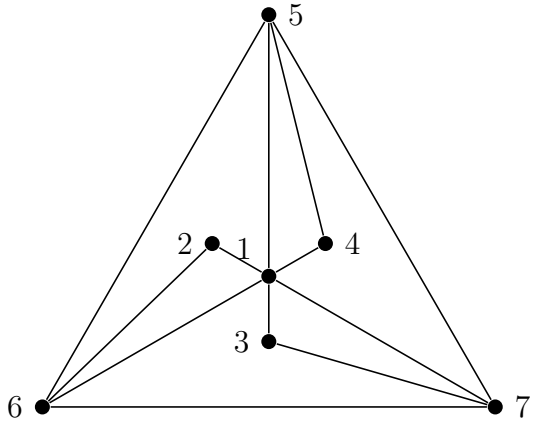


Abbildung 8.9: Genügt der Graph der Bedingung aus Satz 8.6?

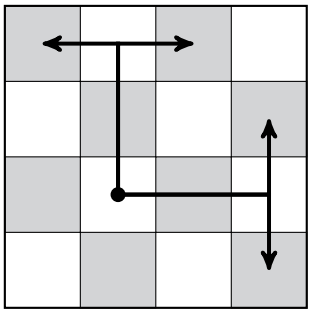


Abbildung 8.10: Bewegungsmöglichkeiten eines Springers

Springer so ziehen, dass er auf jedes Feld genau einmal gesetzt wird und am Ende wieder da ankommt, wo er gestartet ist? Wie sieht es bei einem 5×5– oder 6×6–Schachbrett aus?

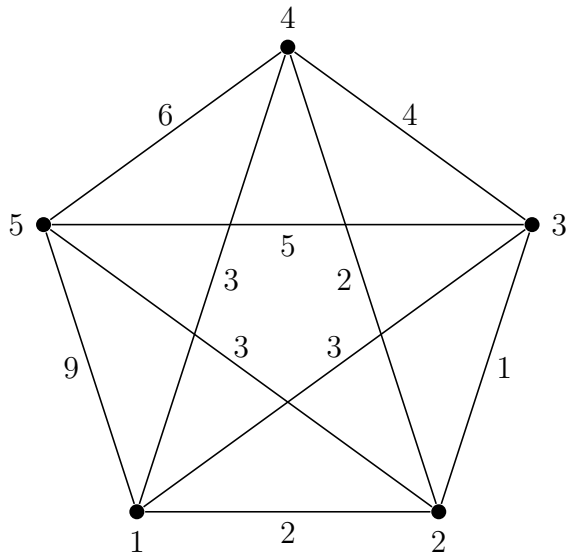


Abbildung 8.11: Wie sieht eine kürzeste Rundreise aus?

Aufgabe 8.5 Betrachten Sie den Graphen aus Abbildung 8.11 und bestimmen Sie möglichst kurze Hamilton-Kreise mit Startknoten 1

(a) mit der Nächster-Nachbar-Heuristik,

(b) mit der Minimaler-spannender-Baum-Heuristik,

(c) mit der Christofides-Heuristik,

(d) von Hand, indem Sie alle 12 Möglichkeiten ausprobieren.

9 Planarität

Ein Problem, bei dem nicht nur Eigenschaften eines Graphen wie Knotengrade und Zusammenhang wichtig sind, sondern auch seine Darstellung, ist die Frage, wie man Leiterbahnen auf Platinen verlegt.

Beispiel 9.1 (Leiterbahnen auf Platinen) Wie schon zu Beginn erwähnt, geht es hierbei darum, Bauteile auf einer Platine so anzuordnen und mit Leiterbahnen zu verbinden, dass diese sich nicht kreuzen. Will man dieses Problem graphentheoretisch formulieren, so betrachtet man alle Bauteile als Knoten und die dazwischen notwendigen Leiterbahnen als Kanten. Dann stellt sich die Frage. Kann man den Graphen so zeichnen, dass sich die Kanten nicht kreuzen? ◊

Ein ähnliches Problem gibt es auch einige Größenordnungen größer.

Beispiel 9.2 (Stom-/Gas-/Wasserleitungen) Wenn ein neues Haus gebaut wird, dann muss dieses an das Strom-/Wasser- und Gasnetz angeschlossen werden. Die hierfür nötigen Leitungen sollten aus Sicherheitsgründen so verlegt werden, dass sie sich nicht kreuzen. ◊

Bisher war es völlig unserem ästhetischen Empfinden überlassen, wie wir einen Graphen dargestellt haben. Dabei konnten wir einen Graphen auf sehr viele verschiedene Arten zeichnen. Bei manchen Darstellungen haben sich mehr Kanten überschnitten, bei anderen weniger. Im folgenden wollen wir daher definieren, was wir unter der Darstellung, insbesondere der planaren Darstellung eines Graphen verstehen.

Definition 9.3 *Ein Paar $\Gamma = (V, E)$ endlicher Mengen heißt* planare Graphendarstellung*, wenn gelten:*

(a) $V \subset \mathbb{R}^2$.

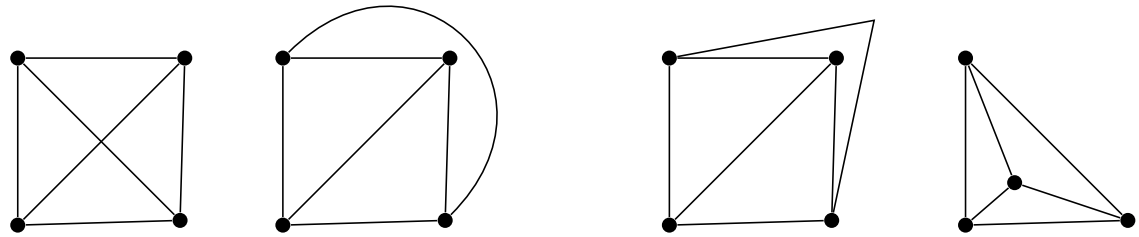
(b) Jede Kante $e \in E$ ist ein Polygonzug, ihre beiden Endpunkte liegen in V.

(c) Abgesehen von ihren Endpunkten hat jede Kante $e \in E$ keinen Punkt mit einer anderen Kante $e' \neq e$ oder einem Knoten $v \in V$ gemeinsam.

Ein gerichteter oder ungerichteter Graph $G = (V, E)$ heißt planar*, wenn er eine planare Graphendarstellung Γ besitzt.*

Ein Polygonzug ist eine Linie, die sich nur aus geraden Stücken zusammensetzt. Dass wir nur Polygonzüge als Kanten zulassen, ist keine wirkliche Einschränkung. Man kann zeigen, dass jeder Graph, der eine planare Graphendarstellung mit glatten Kurven als Kanten

hat, auch eine mit polygonalen Kanten hat. Tatsächlich kann man sogar zeigen, dass jeder einfache (und inversenfreie) planare Graph eine Graphendarstellung besitzt, in der alle Kanten Strecken, also gerade Linien, sind.



(a) klassische Darstellung von K_4

(b) planare Darstellung von K_4 mit glatten Kanten

(c) planare Darstellung von K_4 mit polygonalen Kanten

(d) planare Darstellung von K_4 mit geraden Kanten

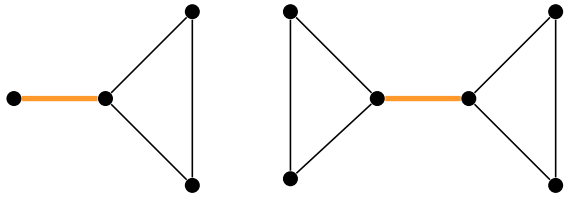
Abbildung 9.1: Ist K_4 planar?

Es stellt sich die folgende Frage: Ist jeder Graph planar? Wenn nicht, wie kann man erkennen, ob ein Graph planar ist? Dies ist einem Graphen nicht so ohne weiteres anzusehen, da auch ein planarer Graph Darstellungen haben kann, die nicht planar sind, vergleiche Abbildung 9.1.

Ob ein Graph planar ist, wird aber offensichtlich nicht davon beeinflusst, ob Kanten eine Richtung haben oder nicht. Daher können wir uns bei der Untersuchung von Planarität auf ungerichtete Graphen einschränken.

Weiter macht es keinen Unterschied, eine Kante nur einmal existiert oder ob es dazu noch parallele Kanten gibt. Im folgenden werden wir unsere Überlegungen daher auf einfache ungerichtete Graphen einschränken.

Aber auch Kanten, die nur einmal vorkommen, können manchmal entfernt werden ohne etwas an der Planarität oder Nichtplanarität eines Graphen zu ändern, vergleiche Abbildung 9.2.



(a) Graph mit Blatt

(b) Graph mit Brücke

Abbildung 9.2: Mögliche Kantenkontraktionen

Diese Beobachtung gibt Anlass zur folgenden Definition.

Definition 9.4 *Es sei $G = (V, E)$ ein einfacher ungerichteter Graph und $e = \{u, v\} \in E$. Die* Kontraktion der Kante *e ergibt den Graphen $G' = (V', E')$, indem die Kante e entfernt wird und die Knoten uv zu einem neuen Knoten $\hat{uv}$ verschmolzen werden, der zu allen Kanten inzident ist, die in G zu u oder v inzident waren. Möglicherweise entstehende Parallelen werden entfernt.*

Ein Graph H heißt Minor *von G, wenn er ein Teilgraph von G ist oder aus diesem durch (eventuell mehrere) Kantenkontraktionen hervorgeht.*

Warum Minoren interessant sind erläutert das folgende Resultat:

Lemma 9.5 *Es sei $G = (V, E)$ ein ungerichteter einfacher Graph. Dann ist G genau dann planar, wenn alle Minoren von G planar sind.*

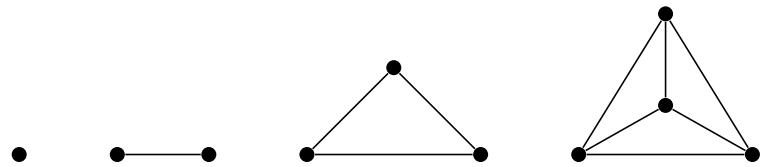
Beweis „$\Longrightarrow$“ Es sei G planar. Dann sind offensichtlich alle Teilgraphen von G ebenfalls planar. Da Kantenkontraktionen Planarität erhalten **klar?**, sind alle Minoren von G ebenfalls planar.

„$\Longleftarrow$“ Es sei G nicht planar. Da G ein Teilgraph von G ist, gibt es dann auch (mindestens) einen nichtplanaren Minor von G. □

Mit Hilfe dieses Resultats können wir nun recht einfach eine, zugegebenermaßen kleine, Menge von planaren Graphen bestimmen. Diese werden wir später an manchen Stellen wiedersehen.

Lemma 9.6 *Alle ungerichteten einfachen Graphen $G = (V, E)$ mit $|V| \leq 4$ sind planar.*

Beweis Die vollständigen Graphen K_1, K_2, K_3 und K_4 sind planar, vergleiche Abbildung 9.3. Alle anderen einfachen Graphen mit höchstens 4 Knoten sind Teilgraphen eines dieser vier planaren Graphen und daher auch planar. □



(a) K_1 (b) K_2 (c) K_3 (d) K_4

Abbildung 9.3: Planare Graphendarstellungen von K_1, K_2, K_3 und K_4

9.1 Die Eulersche Polyederformel

Um einige notwendige Bedingungen für die Planarität herzuleiten, wenden wir uns nun der *eulerschen Polyederformel*. Diese gibt einen Zusammenhang zwischen der Anzahl der Knoten n, der Kanten m, der von Knoten und Kanten eingeschlossenen Flächen f (wobei man die unbeschränkte Fläche mitzählt) und der Zusammenhangskomponenten k einer

planaren Graphendarstellung an. Für eine Illustration der auftretenden Größen vergleiche Abbildung 9.4. Die betrachtete planare Graphendarstellung hat $n = 8$ Knoten, $m = 12$ Kanten, $f = 6$ Flächen und $k = 1$ Zusammenhangskomponente.

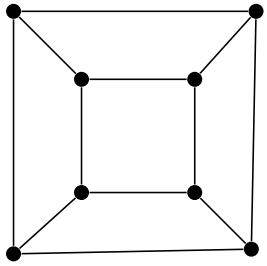


Abbildung 9.4: Planare Graphendarstellungen eines Würfelnetzes

Satz 9.7 *Es sei $\Gamma = (V, E)$ eine planare Graphendarstellung mit n Knoten, m Kanten, f Flächen (inklusive der unbeschränkten) und k Zusammenhangskomponenten. Dann gilt*

$$n - m + f = k + 1.$$

Beweis Wir zeigen die Aussage durch Induktion über die Anzahl der Kanten m. Gilt $m = 0$, so besteht Γ nur aus isolierten Knoten, d.h. die Anzahl der Zusammenhangskomponenten ist $k = n$. Es gibt nur eine Fläche, nämlich die unbeschränkte, also ist $f = 1$. Damit folgt wie gewünscht

$$n - m + f = n + 1 = k + 1.$$

Sei nun für ein $m \geq 1$ die Formel richtig für alle Graphen mit weniger als m Kanten. Wir unterscheiden jetzt zwei Fälle: Sind alle Zusammenhangskomponenten von Γ Bäume, so gilt nach Satz 3.9 $m = n - k$. Da Bäume keine Kreise enthalten gibt es auch hier wieder nur die unbeschränkte Fläche, d.h es gilt $f = 1$. Insgesamt folgt dann

$$n - m + f = n - (n - k) + 1 = k + 1.$$

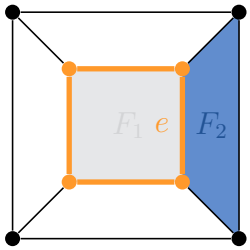


Abbildung 9.5: Illustration des Beweises von Satz 9.7

Gibt es eine Zusammenhangskomponente von Γ, die kein Baum ist, so enthält diese einen einfachen Kreis, vergleiche Abbildung 9.5. Sei e eine Kante auf diesem Kreis. Dann

liegt e auf dem Rand von zwei verschiedenen Flächen F_1 und F_2. Betrachten wir nun die Graphendarstellung $\Gamma' = (V, E \setminus \{e\})$, so hat diese ebenfalls n Knoten, $m-1$ Kanten, immer noch k Zusammenhangskomponenten, da e auf einem einfachen Kreis lag, und $f-1$ Flächen, da durch das Entfernen von e die Flächen F_1 und F_2 verschmelzen. Nach Induktionsannahme gilt daher für Γ'

$$n-(m-1)+(f-1) = k+1 \quad \Longleftrightarrow \quad n-m+f = k+1.$$

□

Hieraus folgt, dass, auch wenn die Graphendarstellung eines planaren Graphen nicht eindeutig ist, die Anzahl der Flächen bei allen planaren Graphendarstellungen gleich ist.

Ist der dargestellte Graph zusammenhängend, so vereinfacht sich die eulersche Polyederformel zu der bekannten Formel für dreidimensionale konvexe Polyeder.

Korollar 9.8 *Es sei $\Gamma = (V, E)$ eine planare Graphendarstellung eines einfachen zusammenhängenden Graphen $G = (V, E)$. Dann gilt*

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

Damit können wir nur einige notwendige Bedingungen für die Planarität von Graphen herleiten.

Satz 9.9 *Es sei $\Gamma = (V, E)$ eine planare Graphendarstellung eines einfachen zusammenhängenden Graphen $G = (V, E)$ mit $n \geq 3$. Dann gelten*

$$m \leq 3n-6 \quad \textit{und} \quad f \leq 2n-4.$$

Beweis Wir nehmen an, dass G zusammenhängend ist. Ist dies nicht der Fall, so können wir solange Kanten hinzufügen, bis G zusammenhängend ist. Dies erhöht die Zahl der Kanten m und der Flächen f und lässt die Zahl der Knoten n unverändert. Kanten zwischen verschiedenen Zusammenhangskomponenten ändern auch nichts an der Planarität von G.

Betrachten wir nun die Nachbarschaften zwischen Kanten und Flächen, so müssen wir für jede Kante beide Seiten betrachten. Da jede beschränkte Fläche von einem einfachen Kreis berandet wird und dieser aus mindestens 3 Kanten besteht, wird jede Fläche mindestens dreimal gezählt. Die unbeschränkte Fläche wird ebenfalls entweder von mindestens 3 verschiedenen Kanten berandet oder es gibt nur 2 Kanten (G ist zusammenhängend und hat mindestens 3 Knoten, also mindestens 2 Kanten), dann wird sie aber für jede Seite gezählt, insgesamt also 4 mal. Folglich gilt $3f \leq 2m$. Mit der eulerschen Polyederformel folgt dann

$$3f = 3(2-n+m) \leq 2m \quad \Longrightarrow \quad m \leq 3n-6.$$

Löst man zuerst nach m auf, folgt

$$2m = 2(n+f-2) \geq 3f \quad \Longrightarrow \quad f \leq 2n-4.$$

□

Aus dem Zusammenhang zwischen Knoten- und Kantenzahl können wir Aussagen über Knotengrade in planaren Graphen ableiten.

Korollar 9.10 *Jeder ungerichtete, einfache planare Graph $G = (V, E)$ mit $|V| \geq 3$ hat mindestens 3 Knoten vom Grad kleiner als 6.*

Beweis Wir nehmen wieder an, das G zusammenhängend ist. Sonst können wir, ohne die Planarität zu zerstören, solange Kanten einfügen, bis G zusammenhängend ist, und erhöhen dabei die Knotengrade nur.

Dann hat jeder Knoten wegen des Zusammenhangs mindestens Grad 1. Gäbe es höchstens 2 Knoten mit Grad kleiner als 6, so würde

$$2m = \sum_{v \in V} d(v) \geq (n-2) \cdot 6 + 2 = 6n - 10 \quad \Longrightarrow \quad m \geq 3n - 5$$

gelten. Andererseits gilt $m \leq 3n - 6 < 3n - 5$, ein Widerspruch. □

Ist ein Graph planar, so muss es nicht nur Knoten mit kleinem Grad geben, sondern es kann auch „nicht zu viele“ Knoten mit großem Grad geben.

Korollar 9.11 *In jedem ungerichteten, einfachen planaren Graphen $G = (V, E)$ ist der durchschnittliche Knotengrad ist kleiner als 6, d.h. es gilt gilt $\frac{1}{n} \sum_{v \in V} d(v) < 6$.*

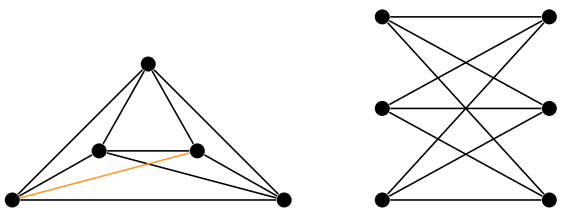
Beweis Angenommen, es würde $\frac{1}{n} \sum_{v \in V} d(v) \geq 6$ gelten. Dann folgte

$$6n \leq \sum_{v \in V} d(v) = 2m \leq 6n - 12 < 6n,$$

ein Widerspruch. □

9.2 Der Satz von Kuratowski

Mit den bisherigen notwendigen Bedingungen können wir zeigen, dass die beiden Graphen aus Abbildung 9.6 nicht planar sind.



(a) K_5 (b) $K_{3,3}$

Abbildung 9.6: Die Graphen K_5 und $K_{3,3}$

Satz 9.12 *Die vollständige Graph K_5 und der vollständige bipartite Graph $K_{3,3}$, vergleiche Abbildung 9.6, sind nicht planar.*

Beweis In K_5 gilt $n = 5$ und $m = 10$. Folglich ist die Bedingung $m \leq 3n - 6 = 9$ verletzt.

In $K_{3,3}$ gilt $n = 6$ und $m = 9$. Angenommen, es gäbe eine planare Graphendarstellung mit f Flächen. Da $K_{3,3}$ zusammenhängend ist, folgt $f = 2 - n + m = 5$. Andererseits wird jede der Flächen von einem Kreis umrandet und, da $K_{3,3}$ einfach und bipartit ist, hat jeder dieser Kreise gerade Länge, besteht also aus mindestens 4 Kanten. Es gilt also $4f \leq 2m = 18$ beziehungsweise $f \leq 4$. Daher kann $K_{3,3}$ keine planare Graphendarstellung besitzen. □

Dieser Satz erlaubt die folgende, auf den ersten Blick nicht sonderlich interessante, Schlussfolgerung:

Korollar 9.13 *Ist $G = (V, E)$ ein ungerichteter, einfacher und planarer Graph, so enthält er weder K_5 noch $K_{3,3}$ als Minoren.*

Dass wir uns genau diese beiden nicht planaren Graphen angeschaut haben, ist aber kein Zufall, denn sie liefern tatsächlich nicht nur ein notwendiges sondern sogar ein hinreichendes Kriterium für die Planarität von Graphen.

Satz 9.14 (Satz von Kuratowski) *Ein ungerichteter einfacher Graph ist genau dann planar, wenn er weder K_5 noch $K_{3,3}$ als Minoren enthält.*

Da der Beweis der noch fehlenden Richtung im Satz von Kuratowski recht länglich ist, wollen wir im Rahmen der Vorlesung darauf verzichten und verweisen auf [2].

9.3 Kreisplanare Graphen

Zum Abschluss dieses Kapitels wollen wir uns noch eine spezielle Klasse von planaren Graphen anschauen, die uns später auch bei der Graphenfärbung wieder begegnen werden.

Definition 9.15 *Es sei $G = (V, E)$ ein ungerichteter einfacher planarer Graph.*

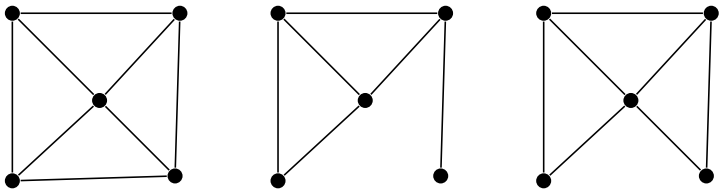
(a) G heißt trianguliert*, wenn jede beschränkte Fläche ein Dreieck ist.*

(b) G heißt kreisplanar*, wenn er eine planare Graphendarstellung besitzt, in der alle Knoten am Rand der gleichen (i.A. der unbeschränkten) Fläche liegen.*

(c) G heißt maximal kreisplanar*, wenn G kreisplanar ist, aber jede hinzugefügte Kante diese Eigenschaft verletzt.*

Kreisplanare Graphen haben also die schöne Eigenschaft, dass man zu ihnen noch einen Knoten hinzufügen kann und diesem mit allen anderen Knoten verbinden kann, ohne dass sich Kanten überschneiden. Genauer gilt sogar die folgende Aussage.

Satz 9.16 *Es sei $G = (V, E)$ ein ungerichteter einfacher Graph und $G^* = (V^*, E^*)$ der Graph, der entsteht, wenn man zu G einen Knoten v^* hinzufügt und diesen mit allen Knoten in V verbindet. Dann ist G^* genau dann planar, wenn G kreisplanar ist.*

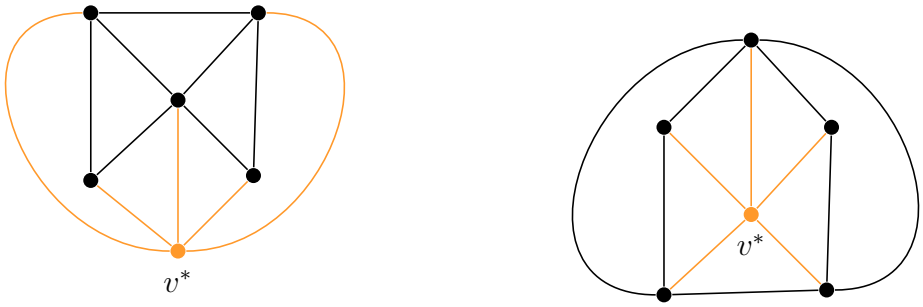


(a) trianguliert (b) kreisplanar (c) maximal kreisplanar

Abbildung 9.7: Illustration von Definition 9.15

Beweis „$\Longleftarrow$“ Sei zunächst G kreisplanar. Dann gibt es eine Graphendarstellung von G, in der alle Knoten am Rand der gleichen Fläche liegen. Platzieren wir nun den neuen Knoten v^* im Inneren dieser Fläche, so können wir ihn sukzessive mit allen Knoten in V verbinden, ohne dass die Kanten sich kreuzen. Dadurch erhalten wir eine planare Darstellung von G^*.

„$\Longrightarrow$“ Sei nun umgekehrt G^* planar. Wir betrachten nun eine planare Darstellung von G^* und entfernen alle Kanten aus E^*, die zu v^* inzident sind, und den Knoten v^*. Da v^* zu allen Knoten in V adjazent war, entsteht dadurch eine Fläche, an deren Rand alle Knoten aus V liegen. Folglich ist G kreisplanar. □



(a) G kreisplanar (b) G^* planar

Abbildung 9.8: Illustration des Beweises von Satz 9.16

Mit Hilfe dieses Resultats lässt sich leicht eine ähnliche Charakterisierung von kreisplanaren Graphen angeben, wie wir sie schon für planare Graphen aus dem Satz von Kuratowski kennen.

Satz 9.17 *Es sei $G = (V, E)$ ein ungerichteter einfacher Graph. Dann ist G genau dann kreisplanar, wenn er weder K_4 noch $K_{2,3}$ als Minoren enthält.*

Beweis „$\Longleftarrow$“ Sei zunächst G kreisplanar. Dann können wir einen neuen Knoten v^* hinzufügen und mit allen Knoten in v verbinden, so dass der entstehende Graph G^* planar ist. Daher kann G nicht K_4 oder $K_{2,3}$ als Minoren enthalten, denn G^* würde dann K_5 beziehungsweise $K_{3,3}$ als Minoren enthalten, wäre also nicht planar.

„$\Longrightarrow$“ Ist umgekehrt G nicht kreisplanar, so ist ist auch der oben konstruierte Graph G^* nicht planar, enthält also K_5 oder $K_{3,3}$ als Minoren. Daher muss G die Graphen K_4 oder $K_{2,3}$ als Minoren enthalten. □

Betrachtet man maximal kreisplanare Graphen, so wird der Ursprung des Namens dieser Graphen offensichtlich.

Satz 9.18 *Es sei $G = (V, E)$ ein ungerichteter einfacher maximal kreisplanarer Graph mid $|V| \geq 3$. Dann liegen alle Knoten von G auf einem einfachen Kreis und das (ohne Einschränkung) Innere des Kreises ist trianguliert.*

Beweis Wir nehmen ohne Einschränkung an, dass die Fläche, an deren Rand alle Knoten von G liegen, die unbeschränkte Fläche ist. Da G maximal kreisplanar ist, ist G zusammenhängend. Sonst könnten die einzelnen Zusammenhangskomponenten von G durch Kanten verbunden werden und der entstehende Graph wäre immer noch kreisplanar.

Wir betrachten nun den Rand der unbeschränkten Fläche. Da G zusammenhängend ist, wird dieser Rand durch *einen* Kreis beschrieben. Wir müssen also nur noch zeigen, dass dieser Kreis einfach ist. Angenommen, er wäre nicht einfach. Da G mindestens drei Knoten hat, bedeutet dies, dass ein Knoten, sagen wir v^*, mehrfach durchlaufen wird. Dann zerfällt G ohne den Knoten v^* in mehrere Zusammenhangskomponenten, vergleiche Abbildung 9.9. Seien nun u, w zwei zu v^* adjazente Knoten, die in G ohne den Knoten v^* in verschiedenen Zusammenhangskomponenten liegen. Dann enthält der Graph die Kante $\{u, w\}$ nicht. Wir können sie aber einfügen ohne die Kreisplanarität zu verletzen, denn u und w liegen danach immer noch am Rand der unbeschränkten Fläche. Würde v^* danach nicht mehr am Rand der unbeschränkte Fläche liegen, so würde die Kante $\{u, w\}$ einen Kreis mit v^* im Inneren schließen. Dies widerspricht aber der Tatsache, dass G ohne v^* zerfällt, d.h. u und w nur über v^* verbunden sind. Da G aber nach Voraussetzung maximal kreisplanar ist, gibt es keine Kante, die wir einfügen können ohne die Kreisplanarität zu verletzen. Daher muss der Kreis einfach sein.

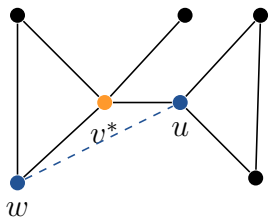
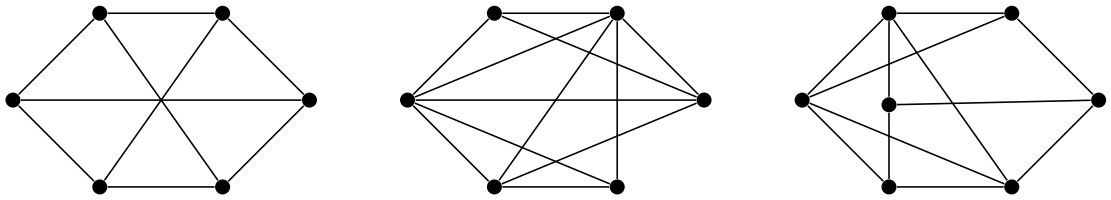


Abbildung 9.9: Illustration des Beweises von Satz 9.18

Angenommen, es gäbe eine beschränkte Fläche, die kein Dreieck ist. Dann könnten wir diese offensichtlich durch eine weitere Kante zerteilen ohne die Kreisplanarität zu verlieren, ein Widerspruch dazu, dass G maximal kreisplanar ist. Daher ist das Innere des Kreises trianguliert. □

Ein maximal kreisplanarer Graph enthält also immer einen (eindeutigen ?) Hamilton-Kreis.



(a) (b) (c)

Abbildung 9.10: Planare Graphen?

9.4 Aufgaben

Aufgabe 9.1 Welche der Graphen aus Abbildung 9.10 sind planar? Begründen Sie Ihre Antworten.

Aufgabe 9.2 Zeigen Sie, dass Bäume planar sind und ihre planare Graphendarstellung immer genau eine Fläche enthält. Gilt auch die Umkehrung, d.h. ist jeder planare Graph, dessen planare Graphendarstellung nur eine Fläche enthält, ein Baum?

Aufgabe 9.3 Es seien $n, m \in \mathbb{N}$ und $K_{n,m}$ der vollständige bipartite Graph wie in Kapitel 6 definiert.

(a) Für welche $m \in \mathbb{N}$ ist $K_{2,m}$ planar?

(b) Für welche $n, m \in \mathbb{N}$ ist $K_{n,m}$ planar?

Aufgabe 9.4 Gibt es Zahlen $n, m, f, k \geq 0$, die der eulerschen Polyederformel genügen, aber zu denen es keinen einfachen planaren Graphen gibt?

Aufgabe 9.5 Betrachte das folgende Spiel: Spieler A und B zeichnen zusammen wie folgt einen ungerichteten Graphen: Spieler A beginnt bei einem Graphen, der $n_0 \geq 2$ isolierte Knoten, also $m_0 = 0$ Kanten besitzt. Beide Spieler dürfen abwechselnd einen Zug der folgenden Art tätigen: Sie dürfen zwei Knoten u, v, die momentan beide einen Grad kleiner als 3 haben, durch einen Weg der Länge 2 verbinden, der über einen neu hinzugefügten Knoten z führt. Hierbei darf auch $u = v$ gelten. Der Graph wird dabei um den neuen Knoten z und die beiden Kanten $\{u, z\}, \{z, v\}$ ergänzt und muss planar bleiben. Der Spieler, der den letzten zulässigen Zug tätigt, gewinnt.

(a) Lässt sich dieses Spiel beliebig lange fortsetzen?

(b) Wie viele Züge müssen mindestens gespielt werden?

10 Knotenfärbung

Zu Beginn hatten wir uns überlegt, welchen Anforderungen ein Stundenplan genügen sollte. Mit Begriffen aus der Graphentheorie können wir dieses Problem jetzt präzisieren.

Beispiel 10.1 (Stundenplanerstellung) Zunächst ordnen wir allen Unterrichtsstunden einen Knoten zu. Alle Stunden, die gemeinsame Teilnehmer (Lehrer oder Schüler) haben, und deswegen nicht gleichzeitig stattfinden können, werden mit einer Kante verbunden. Außerdem ordnen wir jeder Zeit, zu der Veranstaltungen stattfinden können, eine eindeutige Farbe zu. Dann versuchen wir die Knoten in dem entstandenen Graphen so einzufärben, dass keine zwei benachbarten Knoten die gleiche Farbe haben, d.h. die zugehörigen Veranstaltungen nicht gleichzeitig stattfinden.

Alternativ können wir auch, statt die möglichen Unterrichtszeiten vorher festzulegen, versuchen den Graphen mit möglichst wenigen Farben zu färben. Die benötigte Farbenzahl gibt dann an, wieviele Unterrichtszeiten mindestens nötig sind. ◊

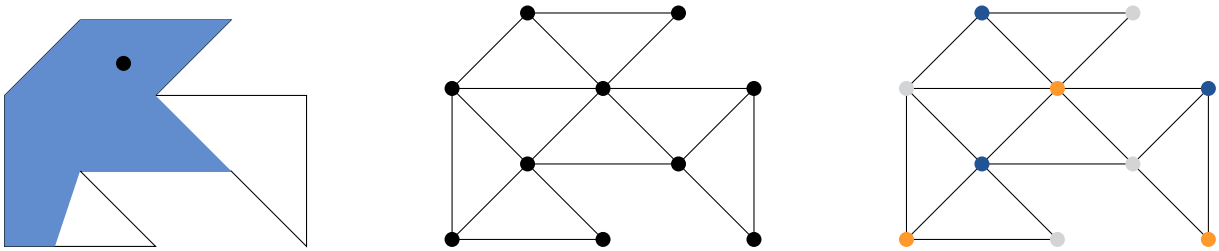
Probleme, die sich auf die gleiche Art beschreiben und lösen lassen, gibt es in vielen Bereichen.

Beispiel 10.2 (Frequenzplanung im Mobilfunk) Zur Übertragung von Daten steht im Mobilfunk jedem Anbieter eine beschränkte Anzahl von Frequenzbändern zur Verfügung. Betrachten wir nun einen Anbieter mit mehreren Mobilfunk-Antennen, so muss dieser festlegen, auf welcher Frequenz welche Antenne senden soll. Hierbei ist zu beachten, dass es, wenn zwei Antennen auf der gleichen Frequenz senden, zu Interferenz kommen kann, die, wenn sie einen gewissen Schwellenwert überschreitet, das gesendete Signal unbrauchbar machen kann. Daher dürfen bestimmte Antennen nicht auf der gleichen Frequenz senden. Andererseits stehen dem Anbieter im Normalfall weniger Frequenzbänder zur Verfügung, als er Antennen hat. Wie soll er entscheiden, welche Antenne auf welcher Frequenz sendet? ◊

Beispiel 10.3 (Landkartenfärbung) Eines der vielleicht bekanntesten Probleme, das sich mit Hilfe der Graphentheorie lösen lässt, ist das Problem der Landkartenfärbung. Hierbei möchte man eine Landkarte so einfärben, dass zwei Länder, die ein Stück Grenze gemeinsam haben, niemals die gleiche Farbe haben. Dies ist natürlich einfach möglich, indem man jedes Land mit einer anderen Farbe einfärbt. Bei sehr großen Karten führt dies jedoch dazu, dass viele Länder ähnliche Farben bekommen und deshalb, wenn sie benachbart sind, schwer zu unterscheiden sind. Daher möchte man eine Landkarte oft mit möglichst wenig Farben einfärben. ◊

Beispiel 10.4 (Museumswärter II) Ein Museum möchte seine Wärter so platzieren, dass sich die gesamte Ausstellungsfläche im Blickfeld von mindestens einem der Wächter befindet. Wir gehen davon aus, dass alle Wärter einen Rundumblick haben, vergleiche Abbildung 10.1(a), Wo sollte das Museum die Wärter aufstellen und wieviele reichen aus um die ganze Fläche zu überwachen?

Um diese Frage zu beantworten, können wir die Ausstellungsfläche in Dreiecke zerlegen, vergleiche Abbildung 10.1(b). Eine Wache, die in einem der Knoten des entstehenden Graphen steht, kann alle angrenzenden Dreiecke überblicken. Es genügt also, wenn es für jedes Dreieck eine Ecke gibt, an der ein Wächter steht. Wir können daher versuchen die Knoten mit drei Farben so einzufärben, dass benachbarte Knoten unterschiedliche Farben haben, d.h. jedes Dreieck drei verschiedenfarbige Ecken hat, vergleiche Abbildung 10.1(c). Wählen wir dann die Farbe, die am seltensten auftritt und platzieren in diesen Knoten einen Wächter, so können wir das gesamte Museum mit höchstens Anzahl der Ecken/3 Wächtern abdecken. ◊



(a) Sichtfeld (b) Triangulation (c) Färbung

Abbildung 10.1: Polygonales Museum

Um den Begriff der Knotenfärbung zu präzisieren, führen wir die folgende Notation ein.

Definition 10.5 *Es sei $G = (V, E)$ ein ungerichteter Graph. Eine Abbildung $c : V \to \{1, 2, \ldots, k\}$ heißt* Knotenfärbung (mit k Farben)*, wenn gilt: Sind $v, w \in V$ adjazent, so ist $c(v) \neq c(w)$. Jedes Element $i \in \{1, \ldots, k\}$ heißt* Farbe *und die Menge $c^{-1}(i) := \{v \in V \mid c(v) = i\}$ die i-te* Farbklasse *von c. Die Zahl*

$$\chi(G) := \min\{k \mid \text{ es gibt eine Knotenfärbung mit } k \text{ Farben}\}$$

heißt chromatische Zahl *von G.*

Die obigen Probleme lassen sich also auf die folgende Fragen zurückführen: Wie findet man die Chromatische Zahl? Wie findet man überhaupt eine Knotenfärbung?

Die chromatische Zahl und Knotenfärbungen hängen offensichtlich nicht davon ab, ob es zu einer Kante noch Parallelen gibt oder nicht, daher werden wir in diesem Kapitel nur einfache Graphen betrachten.

10.1 Ein sequentieller Färbungsalgorithmus

Da wir die folgene Kennzahl in diesem Kapitel öfter verwenden werden, führen wir eine neue Bezeichnung ein.

Definition 10.6 *Es sei $G = (V, E)$ ein gerichteter oder ungerichteter Graph. Dann heißt*

$$\Delta(G) := \max\{g(v) \mid v \in V\}$$

der Maximalgrad *von G.*

Ein einfaches Verfahren, mit dem man eine Knotenfärbung bestimmen kann, ist der sequentielle Färbungsalgorithmus 19. Dieser verwendet aber nicht notwendigerweise die minimale Zahl nötiger Farben.

Algorithmus 19 Sequentieller Färbungsalgorithmus

Input: Ein ungerichteter einfacher Graph $G = (V, E)$ mit $n := |V|$ Knoten.

1 Lege eine beliebige Reihenfolge $v_1, v_2, \ldots, v_n$ der Knoten fest.
2 Setze $c(v_1) = 1$.
3 **for** $i = 2, \ldots, n$ **do**
4 Setze $c(v_i) = \min\{q \in \mathbb{N} \mid \forall_{j<i \text{ mit } \{v_j,v_i\}\in E}\ c(v_j) \neq q\}$.
5 **end for**

Output: Die Knotenfärbung c.

Der folgende Satz belegt, dass dieser Algorithmus immer eine zulässige, wenn auch nicht notwendig optimale, Knotenfärbung bestimmt und gibt eine obere Schranke für die Anzahl der verwendeten Farben an.

Satz 10.7 *Es sei $G = (V, E)$ ein ungerichterer einfacher Graph. Dann ist Algorithmus 19 wohldefiniert und erzeugt eine Knotenfärbung mit höchstens $\Delta(G) + 1$ Farben.*

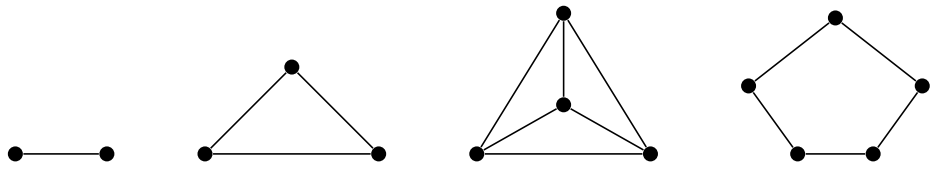
Beweis Da G nur endlich viele Knoten besitzt, gibt es in jeder Iteration eine mindestens eine natürliche Zahl q, die der Bedingung aus Zeile 4 genügt, nämlich die Zahl i. Folglich ist der Algorithmus wohldefiniert.

Die erzeugte Abbildung ist eine Knotenfärbung: Angenommen, es gäbe zwei adjazente Knoten $v_i \neq v_j$ mit $c(v_i) = c(v_j)$. Sei $i > j$. Dann wird in der i-ten Iteration die Farbe $c(v_i)$ so gewählt, dass sie von der Farbe aller schon betrachteten, zu v_i adjazenten, Knoten verschieden ist. Dies schließt aber den Knoten v_j ein, daher kann nicht $c(v_i) = c(v_j)$ gelten.

Es sei $v_i \in V$ ein beliebiger Knoten. Dann gibt es zu v_i genau $g(v_i)$ Nachbarn, folglich gilt $c(v_i) \leq g(v_i) + 1$. Daher ist $\max\{c(v_i) \mid v_i \in V\} \leq \Delta(G) + 1$. □

Aus diesem Resultat folgt sofort eine obere Schranke für die chromatische Zahl eines ungerichteten einfachen Graphen.

Korollar 10.8 *Es sei $G = (V, E)$ ein ungerichteter einfacher Graph. Dann gilt $\chi(G) \leq \Delta(G) + 1$.*



(a) K_2 (b) K_3, Kreis der Länge 3 (c) K_4 (d) Kreis der Länge 5

Abbildung 10.2: Graphen G mit der Eigenschaft $\chi(G) = \Delta(G) + 1$

Diese Aussage wirft die folgende Frage auf: Ist die Abschätzung scharf oder gibt es eine für alle Graphen gültige kleinere obere Schranke für die chromatische Zahl?

Man findet recht leicht Beispiele, für die gilt $\chi(G) = \Delta(G)+1$, vergleiche Abbildung 10.2. Etwas allgemeiner gilt sogar das folgende Resultat.

Lemma 10.9 *Es sei $G = (V, E)$ ein ungerichteter einfacher Graph. Ist G vollständig oder ein einfacher Kreis ungerader Länge, so gilt $\chi(G) = \Delta(G) + 1$.*

Beweis Es sei G ein vollständiger Graph mit n Knoten. Dann gilt für alle $v \in V$ wegen der Vollständigkeit $g(v) = n-1$. In G kann es keine zwei Knoten der gleichen Farbe geben, da alle Knoten benachbart sind. Folglich ist $\chi(G) = n = \max\{g(v) \mid v \in V\} + 1$.

Sei nun G ein einfacher Kreis ungerader Länge mit $n := |V|$. Dann haben alle Knoten in $v \in V$ den Grad $g(v) = 2$. Wir müssen also zeigen, dass es keine Knotenfärbung mit 1 oder 2 Farben gibt. Da jeder Knoten 2 Nachbarn hat, ist eine Knotenfärbung mit einer Farbe nicht möglich. Angenommen, es gibt eine Knotenfärbung mit 2 Farben. Sei $v_1, v_2, v_3, \ldots, v_n$ eine Nummerierung der Knoten in Kreisreihenfolge. Hat Knoten v_1 die Farbe 1, so muss Knoten v_2 die Farbe 2 haben. Knoten v_3 muss dann wieder die Farbe 1 haben und so weiter. Also haben alle Knoten mit ungeradem Index die Farbe 1 und alle Knoten mit geradem Index die Farbe 2. Dies bedeutet aber, dass die benachbarten Knoten v_1 und v_n die gleiche Farbe haben, ein Widerspruch. □

Wir wollen nun zeigen, dass dies aber auch die einzigen zusammenhängenden Graphen sind, für die die obere Schranke wirklich angenommen wird. Dafür betrachten wir zunächst den folgenden, verbesserten Färbungsalgorithmus 20.
Dieser Algorithmus unterscheidet sich also von dem vorherigen nur dadurch, dass wir die Reihenfolge der Knoten nicht mehr beliebig festlegen sondern durch einen spannenden Baum bestimmen.

Satz 10.10 *Es sei $G = (V, E)$ ein ungerichterer, einfacher und zusammenhängender Graph mit (mindestens) einem Knoten $s \in V$ mit $g(s) < \Delta(G)$. Dann ist Algorithmus 20 wohldefiniert und erzeugt eine Knotenfärbung mit höchstens $\Delta(G)$ Farben.*

Beweis Da G zusammenhängend ist, sind Breiten- und Tiefensuche wohldefiniert und finden Wege zu allen Knoten $v \in V$. Abgesehen von der dadurch erzeugten Nummerierung der Knoten stimmt der Algorithmus mit dem sequentiellen Färbungsalgorithmus 19 überein und findet daher eine Knotenfärbung c.

Algorithmus 20 Verbesserter Färbungsalgorithmus

Input: Ein ungerichteter, einfacher und zusammenhängender Graph $G = (V, E)$ mit $n := |V|$ Knoten und (mindestens) einem Knoten $s \in V$ mit $g(s) < \Delta(G)$.

1 Führe eine Breitensuche oder Tiefensuche mit Startknoten s durch und nummeriere die Knoten $v_1, v_2, \ldots, v_n$ in der Reihenfolge ihres Auftretens.
2 Setze $c(v_n) = 1$.
3 **for** $i = n-1, \ldots, 1$ **do**
4 Setze $c(v_i) = \min\{q \in \mathbb{N} \mid \forall_{j>i \text{ mit } \{v_j,v_i\}\in E}\ c(v_j) \neq q\}$.
5 **end for**

Output: Die Knotenfärbung c.

Es sei nun $v_i \in V$ ein beliebiger Knoten mit $i > 1$. Dann gibt es zu v_i genau $g(v_i)$ Nachbarn. Da zwischen $v_1(= s)$ und v_i ein Weg im Suchbaum existiert und alle Knoten auf diesem Weg einen kleineren Index als i haben, können höchstens $g(v_i) - 1 \leq \Delta(G) - 1$ der Nachbarn von v_i in der Menge $\{v_{i+1}, \ldots, v_n\}$ enthalten sein und damit vor v_i eine Farbe erhalten haben. Folglich gilt $c(v_i) \leq (g(v_i) - 1) + 1 \leq \Delta(G)$. Da $g(v_1) \leq \Delta(G) - 1$ gilt, ist auch $c(v_1) \leq \Delta(G)$. Insgesamt folgt $\max\{c(v_i) \mid v_i \in V\} \leq \Delta(G)$. □

Achtung: In dieses und die folgenden Resultate gelten nur für zusammenhängende Graphen. Hätte der Graph mehrere Zusammenhangskomponenten, könnte einen davon ein vollständiger Graph oder ein Kreis ungerader Länge sein. Dann ist nicht mehr gesichert, dass die Aussagen gelten.

Satz 10.11 (Satz von Brooks) *Es sei $G = (V, E)$ ein ungerichteter, einfacher und zusammenhängender Graph. Dann gilt $\chi(G) \leq \Delta(G)$, außer G ist vollständig oder ein einfacher Kreis ungerader Länge. In diesen beiden Fällen gilt $\chi(G) = \Delta(G) + 1$.*

Beweis Die zweite Aussage haben wir schon bewiesen. Wir müssen also nur zeigen, dass für zusammenhängende Graphen, die nicht vollständig oder Kreise ungerader Länge sind, gilt $\chi(G) \leq \Delta(G)$. Dazu folgen wir einem Beweis von [5].

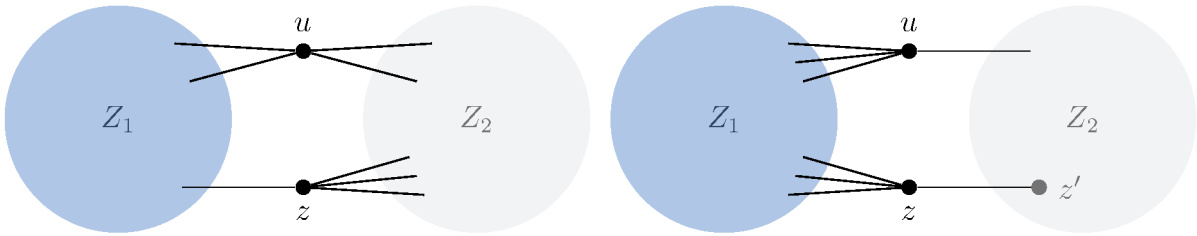
Wir schließen zunächst einige kleine Spezialfälle aus: Die Maximalgrade $\Delta(G) = 0$ und $\Delta(G) = 1$ können für zusammenhängenden Graphen nur in den vollständigen Graphen K_1 und K_2 auftreten. Ist $\Delta(G) = 2$, so ist entweder G ein Weg oder ein Kreis gerader Länge, woraus $\chi(G) = 2 \leq \Delta(G)$ folgt, oder ein Kreis ungerader Länge. Wir müssen also nur noch nicht vollständige Graphen mit $g(v) = \Delta(G) \geq 3$ für alle $v \in V$ betrachten. Denn hat ein Knoten einen kleineren Grad, so folgt die Behauptung aus Satz 10.10.

Da G nicht vollständig ist, muss es einen Knoten s mit Nachbarn u, z geben, die ihrerseits nicht adjazent sind **(klar?)**. Bezeichne nun G' den Graphen G ohne die Knoten u, w und alle zu diesen Knoten inzidenten Kanten.

Fall 1: Ist G' zusammenhängend, so können wir mittels Breiten- oder Tiefensuche von s aus Wege zu allen anderen Knoten in G' finden und bezeichnen die Knoten wieder in der Reihenfolge ihres Auftretens mit $v_1(= s), v_2, \ldots, v_{n-2}$, wobei $n = |V|$ sei. Weiter setzen wir $v_{n-1} = u$ und $v_n = z$. Nun färben wir wieder, bei v_n beginnend, alle Knoten mit

der kleinstmöglichen Farbe. Da u und z nicht adjazent sind, erhalten sie die gleiche Farbe $c(u) = c(z) = 1$. Für alle $i = n-2, \ldots, 2$ gibt es einen Weg zwischen v_i und $s = v_1$ in G', d.h. der Weg enthält die Knoten u, z nicht. Folglich haben alle Knoten auf diesem Weg einen kleineren Index als i und wie im Beweis von Satz 10.10 folgt $c(v_i) \leq (g(v_i)-1)+1 \leq \Delta(G)$. Der Knoten s hat höchstens $\Delta(G)$ Nachbarn, von denen zwei die gleiche Farbe haben, es folgt also auch $c(s) \leq \Delta(G)$, womit die Aussage für diesen Fall gezeigt wären.

Fall 2: Nun müssen wir noch den Fall betrachten, dass G' nicht zusammenhängend ist. Betrachten wir zunächst den Fall, dass es schon genügt, einen der beiden Knoten, sagen wir u, zu entfernen um einen unzusammenhängenden Graphen G'' zu erhalten. Seien $Z''_1, \ldots, Z''_k$ mit $k \geq 2$ die Zusammenhangskomponenten von G''. Jede von diesen Zusammenhangskomponenten muss dann in G über (mindestens) eine Kante mit u verbunden sein. Fügen wir zu jeder Zusammenhangskomponente Z''_i also wieder den Knoten u und die zu u und Z''_i inzidenten Kanten hinzu, so lassen sich die entstehenden Teilgraphen Z_i nach Satz 10.10 mit höchstens $\Delta(G)$ Farben färben, da der Knoten u in allen Z_i einen Grad kleiner als $\Delta(G)$ hat. Wählt man die Farben so, dass u in allen Z_i die gleiche Farbe hat, so hat man eine Knotenfärbung von G mit höchstens $\Delta(G)$ Farben gefunden.



(a) Fall 3(a) (b) Fall 3(b)

Abbildung 10.3: Illustration des Beweises von Satz 10.11

Fall 3: Bleibt noch der Fall, dass G erst durch das Entfernen *beider* Knoten u, z einen unzusammenhängenden Graphen G' ergibt. Seien $Z'_1, \ldots, Z'_k$ mit $k \geq 2$ die Zusammenhangskomponenten von G'. Jede von diesen Zusammenhangskomponenten muss dann in G über jeweils mindestens eine Kante mit u und z verbunden sein. Wir fügen daher zu jeder Zusammenhangskomponente Z_i wieder die Knoten u, z und die zu u bzw. z und Z_i inzidenten Kanten hinzu und erhalten so die Teilgraphen $Z_1, \ldots, Z_k$. Ist $k > 2$, so fügen wir $Z_2, \ldots, Z_k$ zu einem zusammenhängenden Teilgraphen zusammen, den wir im Weiteren trotzdem mit Z_2 bezeichnen wollen. Die entstehenden Teilgraphen Z_1, Z_2 lassen sich nach Satz 10.10 mit höchstens $\Delta(G)$ Farben färben, da u und z in beiden Z_i einen Grad kleiner als $\Delta(G)$ haben. (Sonst hätte es genügt einen von beiden zu entfernen.)

Fall 3(a): Hat in beiden Z_i mindestens einer der beiden Knoten u, z einen Grad von höchstens $\Delta(G) - 2$, so lassen sich beide Z_i so färben, dass u, z in beiden Z_i verschiedene Farben haben, d.h $c(u) \neq c(z)$ gilt. (Idee: Starte in beiden Z_i die Breiten- oder Tiefensuche im dem Knoten mit Grad von höchstens $\Delta(G)-2$ und färbe die Knoten wie üblich. Für den Startknoten, der als letzter gefärbt wird, gibt es dann mindestens zwei mögliche Farben.)

Diese Farben können dann in beiden Z_i gleich gewählt werden, so dass eine Färbung von G mit höchstens $\Delta(G)$ Farben entsteht.

Fall 3(b): Gilt in einem der beiden Teilgraphen, sagen wir in Z_1, $d(u) = d(z) = \Delta(G)-1$, so muss in Z_2 gelten $d(u) = d(z) = 1$. Es gibt also (genau) einen Knoten z' in Z_2, so dass die Kante $\{z, z'\}$ in Z_2 liegt. Wir entfernen nun z und die einzige dazu in Z_2 inzidente Kante $\{z, z'\}$ aus Z_2 und fügen diese Kante sowie den Knoten z' zu Z_1 hinzu. Die gemeinsamen Punkte der beiden Teilgraphen Z_1 und Z_2 sind dann nicht mehr u, z, sondern u, z', wobei in Z_1 gilt $g(z') = 1 \leq \Delta(G) - 2$ und in Z_2 immer noch $d(u) = 1 \leq \Delta(G) - 2$. Wir können also wie in Fall 3(a) fortfahren. □

Die Bestimmung der chromatischen Zahl beliebiger Graphen und die Bestimmung von Knotenfärbungen mit $\chi(G)$ Farben sind sehr schwere Probleme, für die keine schnellen Verfahren existieren. Betrachtet man jedoch spezielle Klassen von Graphen, so wird das Problem einfacher.

Satz 10.12 *Es sei $G = (V, E)$ ein ungerichteter einfacher Graph mit $E \neq \emptyset$. Dann ist $\chi(G) = 2$ genau dann, wenn G bipartit ist. In diesem Fall bestimmt Algorithmus 19 eine Knotenfärbung mit zwei Farben.*

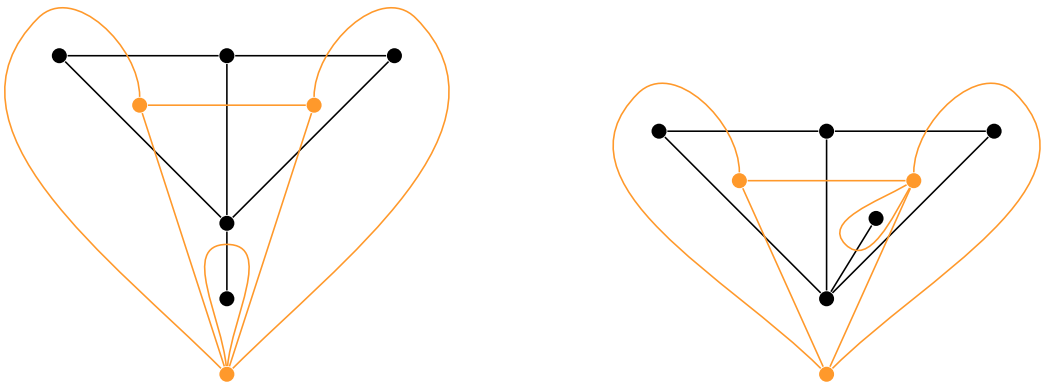
Beweis Es gilt $\chi(G) = 2$ genau dann, wenn V sich in zwei disjunkte, nichtleere Mengen V_1, V_2 aufteilen lässt, so dass keine Kanten zwischen zwei Knoten aus der gleichen Menge existieren und mindestens eine Kante zwischen V_1 und V_2. Dies ist aber genau die Definition dafür, dass ein Graph mit $E \neq \emptyset$ bipartit ist. Da alle Nachbarn eines Knotens aus V_1 in V_2 liegen und umgekehrt, kommt der Algorithmus 19 bei solchen Graphen mit 2 Farben aus. □

10.2 Färbung planarer Graphen

Die Färbung von planaren Graphen ist besonders interessant, da ein enger Zusammenhang zur Färbung von Landkarten besteht. Betrachten wir eine (nicht allzu wilde) Landkarte, so können wir die Grenzen von Ländern als Kanten eines Graphen G auffassen, dessen Knoten die Stellen sind, wo sich 3 oder mehr Länder berühren. Dieser Graph ist planar. Für die Färbung der Landkarte ist allerdings ein anderer Graph G^* wichtiger, der als Knoten gerade die Länder hat und in dem Kanten zwischen zwei Knoten bestehen, wenn die Länder eine gemeinsame Grenze haben. Zwischen den beiden Graphen G und G^* besteht ein enger Zusammenhang.

Definition 10.13 *Es sei $\Gamma = (V, E)$ die Darstellung eines planaren Graphen G. Dann heißt $\Gamma^* = (V^*, E^*)$* dual *zu Γ, wenn gelten: In jeder Fläche F von Γ liegt genau ein $v^* \in V^*$, in jeder Fläche F^* von Γ^* liegt genau ein $v \in V$, jede Kante $e \in E$ schneidet genau eine Kante $e^* \in E^*$ und jede Kante $e^* \in E^*$ schneidet genau eine Kante $e \in E$.*

Achtung: Zwar besitzt jeder planare Graph einen, ebenfalls planaren, dualen Graph G^*, aber dieser hängt von der gewählten Graphendarstellung Γ ab, ist also unter Umstän-



(a) (b)

Abbildung 10.4: Ein planarer Graph und zwei verschiedene zugehörige duale Graphen

den nicht eindeutig. Vergleiche hierfür auch Abbildung 10.4, wo zwei Darstellungen eines planaren Graphen gegeben sind und die entstehenden, verschiedenen, dualen Graphen.

Auch in planaren Graphen ist es möglich eine, vom Maximalgrad unabhängige obere Schranke für die chromatische Zahl anzugeben.

Satz 10.14 *Es sei $G = (V, E)$ ein ungerichteter, einfacher und planarer Graph. Dann gilt $\chi(G) \leq 6$.*

Beweis Sei $n := |V|$. Nach Korollar 9.10 gibt es in $G_0 := G$ mindestens einen Knoten v_1 mit $g(v_1) \leq 5$. Entfernen wir diesen Knoten und alle inzidenten Kanten aus G, so erhalten wir den ebenfalls planaren Graphen G_1. Dieser besitzt folglich auch einen Knoten v_2 mit $g(v_2) \leq 5$. Wir entfernen nun solange immer einen Knoten vom Grad höchstens 5, bis wir einen planaren Graphen G_i mit höchstens 6 Knoten erreicht haben. Dieser lässt sich offensichtlich mit 6 Farben färben. Nehmen wir nun zu G_i wieder den Knoten v_i hinzu, so hat dieser höchstens 5 Nachbarn in G_i. Daher lässt sich auch der Graph G_{i-1} mit 6 Farben färben. Dieses Argument können wir nun solange wiederholen, bis wir wieder bei dem Ausgangsgraphen G angelangt sind. □

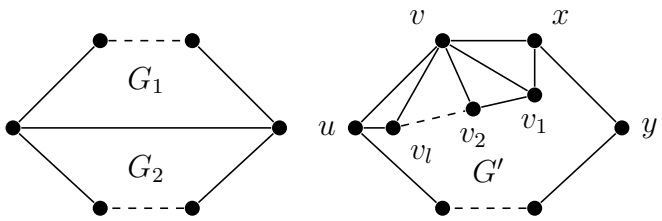
Wenn man allerdings probiert Landkarten zu färben, wird man feststellen, dass man immer mit weniger als 6 Farben auskommt. Daher wollen wir unsere Abschätzung noch weiter verschärfen.

Satz 10.15 *Es sei $G = (V, E)$ ein ungerichteter, einfacher und planarer Graph. Dann gilt $\chi(G) \leq 5$.*

Beweis Für alle planaren Graphen mit höchstens 2 Knoten ist die Aussage offensichtlich richtig. Für alle größeren Graphen zeigen wir durch Induktion über die Zahl der Knoten $n := |V|$ sogar die folgende, stärkere Aussage: Sei G ein planarer Graph mit $n \geq 3$ Knoten und einer planaren Graphendarstellung Γ mit geraden Kanten und P der Kreis, der die unbeschränkte Fläche umrandet. Dann existiert eine Knotenfärbung mit $c(v) \in \{1, 2, \ldots, 5\}$

für alle $v \in V$, wobei man vorher für jedes $v \in P$ zwei beliebige Farben ausschließen und für zwei benachbarte Knoten x, y auf P die Farbe vorher festlegen kann (natürlich unterschiedlich).

Für jeden planaren Graph mit $n \leq 3$ ist die Aussage offensichtlich richtig, da nur ein Knoten noch gefärbt werden muss (wenn man für zwei benachbarte Knoten die Farbe schon festgelegt hat) und dafür 3 Farben zur Verfügung stehen (wenn man für diesen Knoten zwei Farben ausschließt). Sei nun ein $n \geq 3$ gegeben und die Aussagen für alle planaren Graphen mit höchstens n Knoten richtig. Dann betrachten wir einen Graphen G mit $n+1$ Knoten und unterscheiden die folgenden zwei Fälle:



(a) Fall 1 (b) Fall 2

Abbildung 10.5: Illustration des Beweises von Satz 10.15

Fall 1: Hat der Kreis P Sehnen, d.h. gibt es in P zwei Knoten v, u zwischen denen eine Kante existiert, die nicht auf P liegt, so betrachten wir die von den Halbkreisen P_1 bzw. P_2 und der Kante $\{v, u\}$ eingeschlossenen Teilgraphen G_1 und G_2. Diese sind dann planar und besitzen höchstens n Knoten, da auf P zwischen v und u bzw. zwischen u und v jeweils noch ein anderer Knoten liegen muss. Sonst würde die Kante $\{v, u\}$ zu P gehören. Geben wir in zwei benachbarten Knoten x, y auf P die Farbe vor, so liegen beide entweder auf dem Rand von G_1 oder auf dem von G_2. Für den entsprechenden Teilgraphen existiert dann nach Induktionsvoraussetzung eine Färbung, die allen Bedingungen genügt. Die Knoten v und u müssen dabei eine für sie zulässige Farbe erhalten. In dem anderen Teilgraphen können wir diese beiden Knoten nun mit den entsprechenden Farben vorgeben und erhalten nach Induktionsvoraussetzung auch hier eine Färbung, die allen Bedingungen genügt. Die Färbungen der beiden Teilgraphen zusammen ergeben dann eine Färbung von G den gewünschten Eigenschaften.

Fall 2: Gibt es in P keine Knoten v, u, zwischen denen eine Kante existiert, die nicht auf P liegt, und keine Knoten innerhalb von P, so ist G ein Kreis und daher entsprechend färbbar. Nehmen wir nun also an, es gibt noch Knoten innerhalb von P. Weiterhin können wir annehmen, dass G so viele Kanten besitzt, dass alle Flächen innerhalb von P von genau 3 Kanten umrandet werden, Ist dies nicht der Fall, so können wir die fehlenden Kanten einfügen und erhöhen damit die chromatische Zahl höchstens. Seien nun x und y die Knoten auf P mit vorgegebener Farbe und v der andere Nachbar von x auf P. Da P keine Sehnen hat, hat v Nachbarn $v_1, \ldots, v_l$ mit $l \geq 1$ innerhalb von P und einen weiteren Nachbarn u auf P. Dann betrachten wir den von P ohne v_0 und dem Weg $(x, v_1, \ldots, v_l, u)$ berandeten Teilgraphen G'. Dieser ist dann planar und hat n Knoten. Für v sind (mindestens) zwei

von $c(x)$ verschiedene Farben c_1, c_2 zulässig und für $v_1, \ldots, v_l$ bisher alle Farben. Da diese Knoten nun aber auf dem Rand von G' liegen, können wir dort die Farben c_1, c_2 ausschließen und trotzdem nach Induktionsvoraussetzung eine allen Bedingungen genügende Färbung finden. Mindestens eine der für v zulässigen Farben c_1, c_2 unterscheidet sich dann von $c(u)$ und kann für $c(v)$ verwendet werden. □

Tatsächlich gilt sogar die noch schärfere Aussage des folgenden Satzes, welcher unter dem Namen *Vierfarbensatz* bekannt ist. Diese Behauptung wurde zwar schon 1852 formuliert, bis zu einem ersten Beweis, der auf dem Einsatz von Computern beruht, dauerte es jedoch bis 1976. In den ersten Versionen des damals verwendeten Programms fand sich jedoch ein Fehler. Viele Mathematiker lehnten den computerbasierten Beweis außerdem ab, weil er von einem Menschen unmöglich nachvollzogen werden konnte. Deswegen wurden 1997 und 2005 neuere und einfachere Beweise vorgeschlagen, die aber immer noch nicht ohne den Einsatz von Computern auskommen.

Satz 10.16 (Vierfarbensatz) *Es sei $G = (V, E)$ ein ungerichteter, einfacher und planarer Graph. Dann gilt $\chi(G) \leq 4$.*

Zum Abschluss dieses Abschnitts wollen wir noch einmal auf das Museumswärterproblem zurückkommen. Um dieses zu lösen hatten wir das Museum in einen planaren Graphen umgewandelt, den wir mit drei Farben färben wollten. Nach dem Vierfarbensatz lässt sich der Graph immer mit höchstens vier Farben färben. Wir wollen uns nun überlegen, warum in dem hier betrachteten Fall schon drei Farben ausreichen. Dazu nutzen wir eine Eigenschaft des Graphen aus: Da der zu überwachende Raum keine „Löcher“ hatte, ist der entstehende Graph kreisplanar. Wir zeigen daher, dass jeder kreisplanare Graph mit drei Farben gefärbt werden kann.

Satz 10.17 *Es sei $G = (V, E)$ ein ungerichteter einfacher kreisplanarer Graph.*

(a) Dann gibt es mindestens einen Knoten $v \in V$ mit $g(v) \leq 2$.

(b) Es gilt $\chi(G) \leq 3$.

Beweis

(a) Ohne Einschränkung sei G maximal kreisplanar. (Sonst fügen wir solange weitere Kanten ein, bis dies der Fall ist. Dadurch werden die Knotengrade höchstens größer.) Dann ist G zusammenhängend und planar. Wir betrachten eine Darstellung in der alle Knoten am Rand der unbeschränkten Fläche liegen. Nach der eulerschen Polyederformel gilt für die Anzahl der Knoten n, der Kanten m und der Flächen f der Zusammenhang

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

Andererseits ist G trianguliert und die unbeschränkte Fläche wird von einem einfachen Kreis mit n Kanten berandet. Abzählen der beschränkten Flächen und der berandenden Kanten liefert daher

$$3(f - 1) = 2m - n.$$

Beide Formeln zusammen ergeben

$$f - 1 = n - 2,$$

d.h. es gibt $n-2$ Dreiecke. Da jede der n Kanten, die die unbeschränkte Fläche beranden, auch zum Rand eines dieser Dreiecke gehört gibt es mindestens ein Dreieck, das zwei Kanten mit dem Rand der unbeschränkten Fläche teilt. Der dazwischenliegende Knoten v hat dann Grad $g(v) = 2$.

(b) Wir zeigen die Aussage durch Induktion über die Zahl der Knoten n. Für $n \leq 3$ ist die Behauptung offensichtlich wahr. Sei die Aussage nun für alle Graphen mit n Knoten gezeigt und G ein kreisplanarer Graph mit $n + 1$ Knoten. Dann entfernen wir den nach Teil (a) existierenden Knoten v mit $g(v) \leq 2$. Nach Induktionsvoraussetzung lässt sich der entstehende kreisplanare Graph mit n Knoten mit drei Farben färben. Da v nur zwei Nachbarn in G hat, lässt sich folglich auch ganz G mit drei Farben färben.

□

10.3 Aufgaben

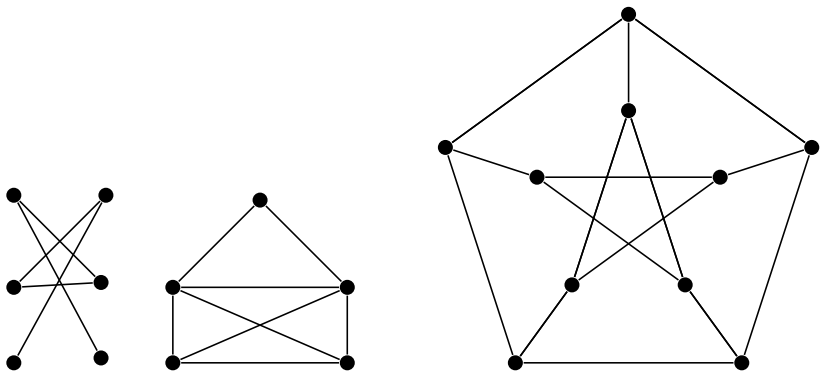


Abbildung 10.6: Was ist die chromatische Zahl dieser Graphen?

Aufgabe 10.1 Bestimmen Sie die chromatische Zahl der Graphen aus Abbildung 10.6.

Aufgabe 10.2 Abbildung 10.7 zeigt die Kreuzung am Eingang zum Nordcampus der Universität Würzburg. Wieviele Ampelphasen sind mindestens nötig, damit alle Autos und Fußgänger, die gleichzeitig Grün haben, sich nicht gegenseitig behindern, d.h. sich ihre Wege nicht kreuzen und sie nicht in die gleiche Spur einbiegen?

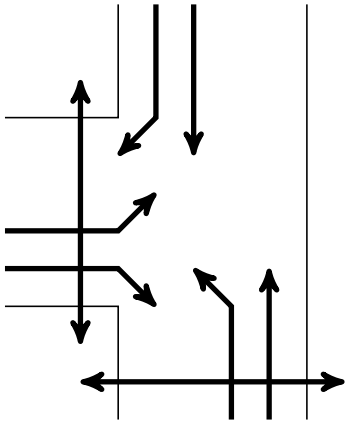


Abbildung 10.7: Kreuzung in Würzburg

Aufgabe 10.3 Beweisen Sie den Satz von De Morgan: Es kann auf der Welt nie 5 Länder geben, so dass jedes Land mit jedem benachbart ist.

Aufgabe 10.4 Was hat Sudoku mit Graphentheorie zu tun?

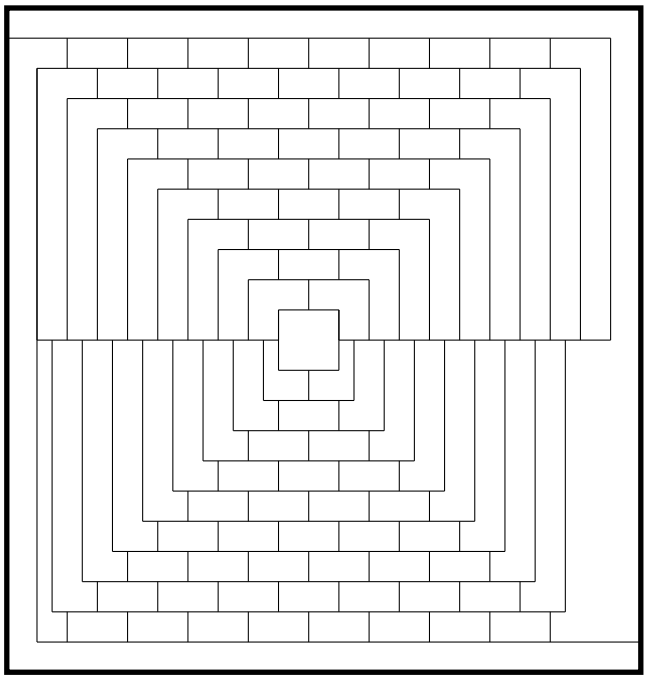


Abbildung 10.8: Ein Gegenbeispiel zum Vierfarbensatz?

Aufgabe 10.5 Als Aprilscherz wurde 1975 im Scientific American die Karte aus Abbildung 10.8 abgedruckt mit der Behauptung, man habe hiermit ein Gegenbeispiel zum Vierfarbensatz gefunden. Es war doch ein Aprilscherz, oder?

Literaturverzeichnis

[1] C. Büsing: *Graphen- und Netzwerkoptimierung*, Leitfäden der Informatik, Spektrum Akademischer Verlag, Heidelberg, 2010.

[2] R. Diestel: *Graph Theory*, Graduate Texts in Mathematics, Springer Verlag, New York, 1997.

[3] T.L. Gertzen und M. Grötschel: *Flinders Petrie, the Travelling Salesman Problem, and the Beginning of Mathematical Modeling in Achaeology*, in M. Grötschel (Ed.): *Documenta Mathematica: Optimization Stories*, 2012, pp. 199–210

[4] M. Grötschel und Y.-X. Yuang: *Euler, Mei-Ko Kwan, Königsberg and a Chinese Postman*, in M. Grötschel (Ed.): *Documenta Mathematica: Optimization Stories*, 2012, pp. 43–50

[5] C. Hurkens: *PhD-Course in Combinatorial Optimization*, http://www.win.tue.nl/ wscor/OW/CO1b/brooks.pdf, 2007.

[6] S. Hußmann und B. Lutz-Westphal (Hrsg.): *Kombinatorische Optimierung erleben*, Vieweg Verlag, Wiesbaden, 2007.

[7] C. Kanzow: *Operations Research*, Vorlesungsskript, Würzburg, 2010.

[8] M. Lihoreau, L. Chittka und N.E. Raine: *Travel Optimization by Foraging Bumblebees through Readjustements of Traplines after Discovery of New Feeding Locations*, The American Naturalist, Vol. 176-6, 2010, pp. 744–757.

[9] S.O. Krumke und H. Noltemeier: *Graphentheoretische Konzepte und Algorithmen*, Teubner Verlag, Wiesbaden, 2005.