

Christina Büsing

Graphen- und Netzwerkoptimierung



Graphen- und Netzwerkoptimierung

Christina Büsing

Graphen- und Netzwerkoptimierung

Autorin:

Christina Büsing

TU Berlin

Fakultät II Mathematik und Naturwissenschaften, Institut für Mathematik

Straße des 17. Juni

10623 Berlin

cbuesing@math.tu-berlin.de

Wichtiger Hinweis für den Benutzer

Der Verlag, der Herausgeber und die Autoren haben alle Sorgfalt walten lassen, um vollständige und akkurate Informationen in diesem Buch zu publizieren. Der Verlag übernimmt weder Garantie noch die juristische Verantwortung oder irgendeine Haftung für die Nutzung dieser Informationen, für deren Wirtschaftlichkeit oder fehlerfreie Funktion für einen bestimmten Zweck. Der Verlag übernimmt keine Gewähr dafür, dass die beschriebenen Verfahren, Programme usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürfen. Der Verlag hat sich bemüht, sämtliche Rechteinhaber von Abbildungen zu ermitteln. Sollte dem Verlag gegenüber dennoch der Nachweis der Rechtsinhaberschaft geführt werden, wird das branchenübliche Honorar gezahlt.

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

© Spektrum Akademischer Verlag Heidelberg 2010

Spektrum Akademischer Verlag ist ein Imprint von Springer

10 11 12 13 14

5 4 3 2 1

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Planung und Lektorat: Dr. Andreas Rüdinger, Sabine Bartels

Herstellung: Crest Premedia Solutions (P) Ltd, Pune, Maharashtra, India

Satz: Autorensatz

Umschlaggestaltung: SpieszDesign, Neu-Ulm

Titelfotografie: © tom; Fotolia.com

ISBN 978-3-8274-2422-8

Vorwort

Alle Wege führen nach Rom! Und welcher ist der beste? Was heißt „der beste“ und wie findet mein Navi einen solchen? Diese Fragen bilden nur einen kleinen Teilaspekt dessen, was in diesem Buch behandelt wird. Andere Fragestellungen betreffen das Färben von Landkarten, die Analyse von Abwassersystemen oder die Planung von Massenhochzeiten.

Historisch gesehen begann die Graphentheorie im Jahr 1736, als Leonhard Euler sein Königsberger Brückenproblem vorstellte: Kann man den wöchentlichen Sonntagsspaziergang durch die Stadt so planen, dass man jede der sieben Brücken genau einmal überquert und am Ende wieder zu Hause ankommt (Abb. 1, links)? Beim Lösen dieses Problems schuf Euler die Graphen. Hierbei handelt es sich nicht etwa um Funktionsgraphen, wie man diese aus der Kurvendiskussion kennt; ein Graph in unserem Sinn besteht aus Knoten und Kanten, die diese Knoten verbinden (Abb. 1, rechts). Mit einem solchen Konzept lässt sich das Brückenproblem in Königsberg und auch in jeder anderen Stadt einfach lösen. Darin besteht auch heute noch ein Reiz der Graphentheorie: Viele Probleme aus der Praxis lassen sich leicht in ihre Sprache übersetzen. Häufig treten schon dadurch die Problemstruktur und die Hauptschwierigkeit deutlich zutage.

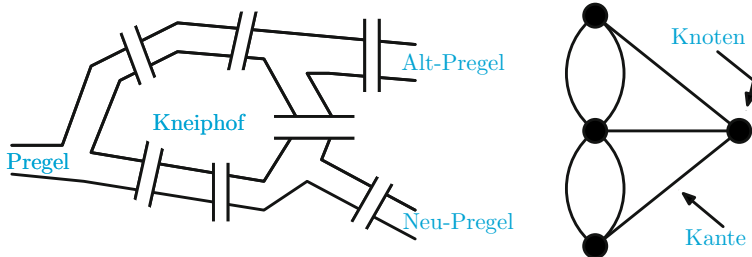



Abb. 1: Links sehen Sie Königsberg zu Eulers Zeiten und rechts eine Darstellung als Graph.

Heute stellt die Graphentheorie eines der wichtigsten Teilgebiete der Diskreten Mathematik dar. Ihre Anwendungsfelder wurden in den letzten Jahrzehnten vielfältiger und für die moderne Welt immer entscheidender. Konnten sich Unternehmen früher noch durch den Erwerb besserer Produktionsmittel voneinander absetzen, so ist heute die optimale Nutzung der vorhandenen Ausstattung entscheidend. Eine moderne Fluggesellschaft etwa muss mit den vorhandenen Flugzeugen und Flughafenzeiten möglichst viele Fluggäste befördern. Dieses Problem lässt sich beispielsweise mit der Graphentheorie als spezielles Multi-Commodity-Flow-Problem formulieren und optimal lösen. Auch beim Lotsen von Automatic Guided Vehi-

les durch den Hamburger Hafen kommt die Graphentheorie zum Tragen und hilft dabei die Schiffe so schnell wie möglich zu entladen.

Welche Fragestellungen beschäftigen einen Mathematiker, wenn er ein Problem untersucht? Als erstes muss natürlich die Problemstellung in die Sprache der Graphentheorie übersetzt werden. Betrachtet man das Königsberger Brückenproblem, so bilden die Landmassen die Knoten und die Brücken die Kanten. In diesem Graphen wird anstelle eines Spaziergangs eine so genannte Euler-Tour gesucht, die jede Kante genau einmal durchläuft. Des Weiteren stellt sich die Frage, welche Eigenschaft ein Graph besitzen muss, damit eine Euler-Tour existiert; und ob man diese Eigenschaft nutzen kann, um effiziente Algorithmen zu konstruieren, die eine solche Tour bestimmen. Diese beiden Fragen nach einer Charakterisierung und einer Lösungsmöglichkeit werden uns bei allen Problemstellungen begegnen. Die Antworten fallen jedoch sehr unterschiedlich aus. Beim Königsberger Brückenproblem können sie positiv beantwortet werden, bei anderen Problemen ist hingegen sowohl eine Charakterisierung als auch die Konstruktion von Lösungsverfahren schwierig. Einige Lösungsverfahren sind nur für kleine Probleminstanzen geeignet, wohingegen bei großen Problemen sogar ein Supercomputer Jahre zur Berechnung der Lösung benötigen würde.

Aufbau des Buches

Jedes Kapitel dieses Buchs behandelt ein Themengebiet aus der Graphentheorie bzw. der Netzwerkoptimierung. Den Ausgangspunkt bildet immer ein Problem aus der Praxis, anhand dessen die graphentheoretischen Begriffe und Definitionen eingeführt werden. An Beispielen und ersten Beobachtungen werden dann die Charakteristika eines Problems und die gängigen Lösungsverfahren erarbeitet. Der Text ist dabei in Definitionen, Sätze, Beispiele und Beweise gegliedert. Diese Form erlaubt einen schnellen Überblick über die Themen und erleichtert die Arbeit mit diesem Buch. Eine kurze Einführung in diese Struktur bietet der Anhang A *Satz, Beweis, Definition*. Wichtig beim Lesen des Textes ist es, sich neue Begriffe, Aussagen oder Algorithmen über Beispiele zu verdeutlichen und zu überprüfen. Deswegen finden Sie im Text immer wieder kleine Aufgaben, die mit einem Übungssymbol  am Rand des Textes gekennzeichnet sind. Einige dieser Fragen sind mit einer Nummer versehen. Eine Lösung befindet sich dann am Ende des jeweiligen Kapitels nach dem Aufgabenteil.

Das Buch richtet sich an interessierte **Schüler** der Oberstufe, **Studenten** der Mathematik und Informatik im ersten und zweiten Semester sowie Praktiker. Es wird neben den mathematischen Grundkenntnissen aus der Schule kein weiteres Wissen vorausgesetzt. In dem schon erwähnten Anhang A finden Sie eine Einführung in die wichtigsten Beweismethoden; gerade die Beweise im ersten Kapitel sind gut verständlich und lassen sich anhand eines Beispiels leicht nachvollziehen. Die



grundlegenden Inhalte eines Kapitels können Sie allerdings auch ohne die Beweise verstehen. Ebenfalls im Anhang enthalten ist ein Kapitel B über die in diesem Buch verwendeten Symbole. Falls Ihnen die Zeichen \in , \sum , \prod , und \forall unbekannt sind, sollten Sie einen kurzen Blick in diesen Abschnitt werfen.

Inhalt

Im *ersten Kapitel* werden grundlegende Begriffe wie Graph, Grad, Weg und Konzepte der Graphentheorie wie der Zusammenhang eingeführt. Außerdem wird die spezielle Graphenklasse der Bäume untersucht. Das *zweite Kapitel* handelt von der Breiten- und Tiefensuche, mit deren Hilfe sich ein gegebener Graph auf sein Grundgerüst, einen Baum, reduzieren lässt. Die Algorithmen können ebenso dafür verwendet werden, einen Gegenstand in einem Labyrinth zu finden, einen kürzesten Weg von einem Startknoten bezüglich der Anzahl der Kanten zu berechnen oder einen Graphen auf Bipartitheit zu überprüfen.

Auch im *nächsten Kapitel* beschäftigen wir uns mit Bäumen. Dabei werden wir das Minimal-Spannende-Baum-Problem lösen, d.h. zu einem gegebenen Graphen mit Kantengewichten einen günstigsten spannenden Baum in diesem Graphen finden. Mit dem Algorithmus von Prim oder dem Algorithmus von Kruskal, kann ein solcher spannender Baum schnell berechnet werden. Anwendung findet das Problem beispielsweise bei der Verlegung von Kabeln in einem Server-Raum.

Im *vierten Kapitel* wenden wir uns dem Reisen zu und behandeln das schon beschriebene Königsberger Brückenproblem. Neben der Modellierung durch Graphen hat Euler schon damals eine einfache Charakterisierung aller Graphen gefunden, die eine Euler-Tour enthalten. Ein Algorithmus zur Berechnung einer solchen Tour wurde allerdings erst von Hierholzer 1873 veröffentlicht. Als Alternative zu diesem Algorithmus wird noch der Algorithmus von Fleury vorgestellt. Beide Algorithmen können auch dazu verwendet werden einen Euler-Weg zu berechnen.

Nach den Euler-Graphen behandeln wir ein weiteres Reise-Problem, bei dem eine Rundreise gesucht ist, die anstelle aller Kanten, alle Knoten genau einmal durchläuft. Ein solcher Graph heißt hamiltonsch und schon hier sei vorweggenommen, dass eine einfache Charakterisierung hamiltonscher Graphen noch nicht bekannt ist. Der Satz von Dirac bietet lediglich eine hinreichende Bedingung über den Minimalgrad eines Graphen. Erweitert man das Problem auf vollständige Graphen mit Kantengewichten, so erhält man das berühmte *Travelling-Salesman-Problem*, für das noch keinen effizienten Algorithmus existiert. Wieso vermutlich auch nie einer gefunden wird und was genau „effizient“ bedeutet, wird im Abschnitt über die Komplexitätstheorie erklärt.

Im *sechsten Kapitel* beschäftigt uns die Frage, ob ein Graph so gezeichnet werden kann, dass sich keine Kanten überkreuzen. Solche planaren Graphen haben viele schöne Eigenschaften und erfüllen vor allem die Euler-Formel, welche einen Zu-

sammenhang zwischen der Anzahl an Knoten, Kanten und Flächen eines planaren Graphen herstellt. Des Weiteren wird diese Klasse durch den Satz von Kuratowski vollständig charakterisiert.

Das *siebte Kapitel* behandelt das Färben von Graphen. Darunter fällt insbesondere das berühmte Vier-Farben-Problem, das die Mathematiker über 100 Jahre in Atem hielt. Dabei sollte die Vermutung bewiesen werden, jede Landkarte könne mit vier Farben gefärbt werden, ohne dass benachbarte Länder dieselbe Farbe erhalten. Den Vier-Farben-Satz werden wir nicht beweisen, dafür den etwas schwächeren Fünf-Farben-Satz.

Im *nächsten Kapitel* betrachten wir gerichtete Graphen und erweitern viele schon behandelte Problemstellungen auf diese. Im Anschluss (Kapitel 9) beschäftigen wir uns mit der Berechnung eines kürzesten Wegs zwischen zwei gegebenen Orten. Neben einem Optimalitätskriterium lernen wir den Algorithmus von Dijkstra und den Algorithmus von Moore-Bellman-Ford kennen, der auch bei Graphen mit negativen Kantengewichten einen kürzesten Weg berechnet.

Kapitel 10 behandelt die Fragestellung, wieviel Wasser in einem Wassersystem an einer bestimmten Stelle maximal zur Verfügung stehen kann. Dazu entwickeln wir zuerst eine sinnvolle Modellierung der gegebenen Problemstellung über (maximale) Flüsse. Der Algorithmus von Ford-Fulkerson berechnet dazu eine optimale Lösung, deren Maximalität mit Hilfe des Max-Fluss-Min-Schnitt-Satzes bewiesen wird.

In *Kapitel 11* untersuchen wir das Problem, Güter möglichst kostengünstig in einem Netzwerk zu transportieren. Dazu erweitern wir die Definition eines Flusses aus dem vorherigen Kapitel, lernen ein neues Optimalitätskriterium über negative Zyklen kennen, und berechnen anschließend einen solchen kostenminimalen Fluss mit dem Cycle-Canceling- oder dem Successive-Shortest-Path-Algorithmus.

Das *letzte Kapitel* behandelt das Problem Paare zu bilden, wie es zum Beispiel bei der Zuordnung von Bewerbern zu Jobs auftreten kann. Für bipartite Graphen beweisen wir hier den Satz von König und den Satz von Hall und lernen dabei einen Algorithmus zur Berechnung einer maximalen Paarung bzw. eines maximalen Matchings kennen.

Literatur

Drei Bücher sollen hier kurz vorgestellt werden, die gut als weiterführende Literatur geeignet sind. Das Buch von Douglas West „Introduction to Graph Theory“ wurde als Begleitbuch einer Vorlesung zur Graphentheorie geschrieben. Das Buch enthält anschauliche Beispiele und behandelt alle Themen bis auf das der Flüsse (Kapitel 10 und 11) auf einem mathematisch abstrakterem Niveau als dies hier in diesem Buch geschieht. Allerdings wird von dem Leser ein hohes Maß an mathematischem Verständnis vorausgesetzt. Als Ergänzung zu dem Thema der Flüsse

sei das Buch „Network Flows“ von Ravindra Ahuja, Thomas Magnanti und James Orlin empfohlen. Das letzte Buch „Kombinatorische Optimierung erleben“, auf das hier verwiesen werden soll, stammt von Stephan Hußmann und Brigitte Lutz-Westphal. Ein großer Schwerpunkt des Buches liegt in der didaktischen Aufbereitung der Kombinatorischen Optimierung und seiner Anwendung in Schulen. Der Leser wird zum Mitarbeiten ermutigt und lernt so die Arbeitsweise von Mathematikern – insbesondere auch die vielen Irrwege bis zur Lösungen eines Problems – kennen.

Das hier vorliegende Buch soll den Leser in die grundlegenden Themengebieten der Graphentheorie und Netzwerkoptimierung einführen. Alle Sätze, Beweise und Algorithmen entsprechen dem Standardwissen in diesem Gebiet und werden hier mit vielen Beispielen und Anwendungen vorgestellt.

Danksagung

Dieses Buch wäre ohne die Unterstützung meiner Kollegen, Freunde und Familie nicht entstanden. Die Idee zu diesem Buch entstand bei der Vorbereitung mehrerer Kurse für die Deutschen Schülerakademie, die ich mit Georg Hoever gehalten habe. Bei ihm möchte ich mich für die vielen Anregungen, kritischen Nachfragen und Diskussionen bedanken. Henrik Büsing, Wiebke Höhn, Mareike Massow, Jens Schulz und Madeleine Theile haben ebenfalls mit ihren Anmerkungen zu den Kapiteln wertvolle Unterstützung geliefert und das Buch um einige Beispiele reicher gemacht. Für den sprachlichen Schliff hat Tobias Kober gesorgt, der sich als Germanist und Theologe durch die vielen Kapitel gearbeitet hat.

Der Aufbau des Buches orientiert sich an dem von Professor Möhring entwickelten Skript zur Vorlesung Algorithmische Diskrete Mathematik. Für sein Engagement in der Vorlesung, seine ansteckende Begeisterung für dieses Gebiet und die viele Zeit, die er mir zur Verfügung gestellt hat, möchte ich mich ganz herzlich bedanken. Abschließend gilt mein Dank Frau Bartels und Herrn Rüdinger von Spektrum Akademischer Verlag für die gute Zusammenarbeit.

Inhaltsverzeichnis

| | | |
|----------|---|------------|
| 1 | Erste Orientierung in der Graphentheorie | 1 |
| 1.1 | Erster Schultag | 1 |
| 1.2 | Zusammenhang und Schnitte | 10 |
| 1.3 | Bäume | 14 |
| 1.4 | Aufgaben | 17 |
| 2 | Tiefen- und Breitensuche | 21 |
| 2.1 | Spannende Bäume | 21 |
| 2.2 | Wie findet man spannende Bäume? | 24 |
| 2.3 | Anwendungen von BFS und DFS | 29 |
| 2.4 | Aufgaben | 33 |
| 3 | Das Minimal-Spannende-Baum-Problem | 39 |
| 3.1 | Das Problem und zwei Algorithmen | 39 |
| 3.2 | Zwei Optimalitätskriterien | 43 |
| 3.3 | Aufgaben | 51 |
| 4 | Euler-Touren und -Wege | 57 |
| 4.1 | Das Königsberger Brückenproblem | 57 |
| 4.2 | Die Algorithmen von Hierholzer und Fleury | 61 |
| 4.3 | Euler-Wege oder das Haus vom Nikolaus | 65 |
| 4.4 | Aufgaben | 68 |
| 5 | Noch zwei Rundreise-Probleme | 71 |
| 5.1 | Hamilton und das Icosian-Spiel | 71 |
| 5.2 | Das <i>Travelling-Salesman</i> -Problem | 78 |
| 5.3 | Komplexitätstheorie | 84 |
| 5.4 | Aufgaben | 86 |
| 6 | Planarität | 91 |
| 6.1 | Gas-Wasser-Strom und Planarität | 91 |
| 6.2 | Outerplanare Graphen | 95 |
| 6.3 | Die Euler-Formel | 98 |
| 6.4 | Die Graphen $K_{3,3}$, K_5 und Kuratowski | 104 |
| 6.5 | Aufgaben | 108 |
| 7 | Knotenfärbung | 111 |
| 7.1 | Die chromatische Zahl | 111 |
| 7.2 | Das Vier-Farben-Problem | 115 |
| 7.3 | Aufgaben | 121 |
| 8 | Gerichtete Graphen und Turniergraphen | 125 |
| 8.1 | Gerichtete Graphen — Digraphen | 125 |
| 8.2 | Starker Zusammenhang | 128 |
| 8.3 | Gerichtete Euler-Graphen | 132 |

| | | |
|-----------|--|-----|
| 8.4 | Hamilton-Wege in Turniergraphen | 134 |
| 8.5 | Könige in Turniergraphen | 136 |
| 8.6 | Aufgaben | 141 |
| 9 | Kürzeste Wege | 145 |
| 9.1 | Der Kürzeste-Wege-Baum | 145 |
| 9.2 | Ein Optimalitätskriterium und der Dijkstra-Algorithmus | 149 |
| 9.3 | Negative Kosten | 155 |
| 9.4 | Aufgaben | 158 |
| 10 | Maximale Flüsse | 163 |
| 10.1 | Flüsse und der Dekompositionssatz von Ford-Fulkerson | 163 |
| 10.2 | Das maximale Fluss-Problem | 169 |
| 10.3 | Der Max-Fluss-Min-Schnitt-Satz | 176 |
| 10.4 | Aufgaben | 181 |
| 11 | Kostenminimale Flüsse | 187 |
| 11.1 | Problemstellung | 187 |
| 11.2 | Ein Optimalitätskriterium | 190 |
| 11.3 | Zwei Algorithmen | 193 |
| 11.4 | Aufgaben | 199 |
| 12 | Maximale Matchings | 203 |
| 12.1 | Definition und ein Optimalitätskriterium | 203 |
| 12.2 | Matchings in bipartiten Graphen | 207 |
| 12.3 | Aufgaben | 218 |
| 13 | Lösungshinweise | 223 |
| A | Satz, Beweis, Definition | 247 |
| A.1 | Folgerungen und Äquivalenzen | 248 |
| A.2 | Negationen | 249 |
| A.3 | Beweis durch Widerspruch | 250 |
| A.4 | Vollständige Induktion | 251 |
| A.5 | Ringschluss | 253 |
| B | Zeichen und Symbole | 257 |
| B.1 | Aus der Mengentheorie | 257 |
| B.2 | Aus der Arithmetik | 258 |
| B.3 | Zwei Quantoren | 259 |
| | Literaturverzeichnis | 261 |
| | Index | 263 |

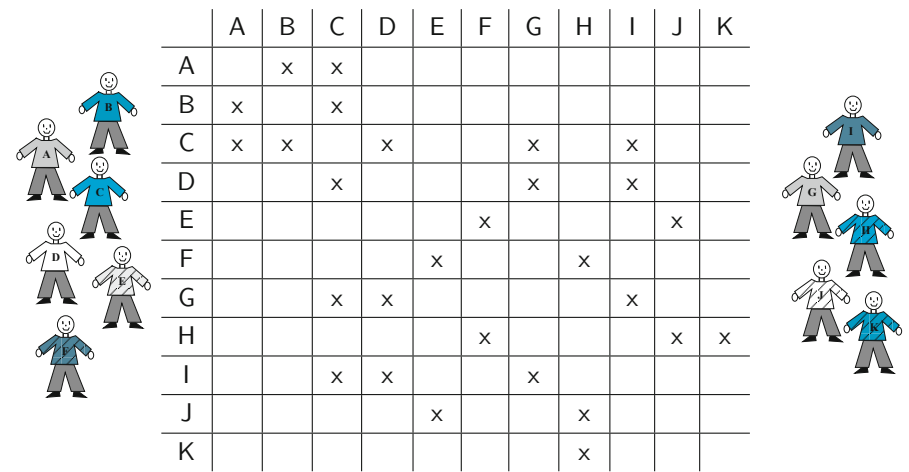
1 Erste Orientierung in der Graphentheorie

Übersicht

| | | |
|-----|---------------------------------|----|
| 1.1 | Erster Schultag | 1 |
| 1.2 | Zusammenhang und Schnitte | 10 |
| 1.3 | Bäume..... | 14 |
| 1.4 | Aufgaben | 17 |

1.1 Erster Schultag

Anne ist neu in der Klasse 7a des Reinhardt-Gymnasiums. Ihre Tischnachbarin Carolin möchte ihr helfen, sich so schnell wie möglich in der Klasse zurecht zu finden.



Tab. 1.1: Das Freundschaftsdiagramm eines Teils der Klasse 7a. A steht für Andreas, B für Beate, C für Claudia, D für Doris, E für Emil, F für Fabian, G für Gabi, H für Hans, I für Inga, J für Julia und K für Karsten.

Aber Anne kann sich die ganzen Namen und Geschichten gar nicht merken, die Carolin ihr erzählt. Damit sie zu Hause die Chance hat, alles zu wiederholen, entwirft Carolin ihr eine Tabelle, in die sie die wichtigsten Leute einträgt und wer sich mit wem gut versteht (Tab. 1.1).

Anne freut sich sehr über die Tabelle, aber zu Hause angekommen fällt es ihr schwer, sich ein Bild von der Klasse zu machen. Deswegen fertigt sie eine Zeichnung an, in der jede Person durch einen Punkt dargestellt ist. Wenn zwei Personen sich gut verstehen, d.h. wenn ein Kreuz in der Tabelle ist, verbindet sie die beiden Punkte durch eine Linie (Abb. 1.1).

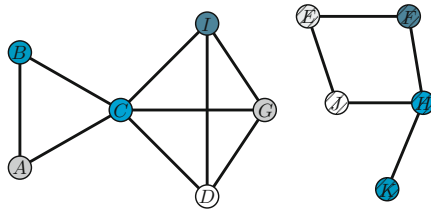


Abb. 1.1: Annes Zeichnung der Klasse.

Beim Zeichnen fällt ihr auf, dass es egal ist, wie genau sie die Punkte malt (sie könnte auch Dreiecke oder Vierecke nehmen) oder wie die Linien aussehen. Die Menschen und ihre Beziehungen bleiben immer dieselben. Etwas abstrakter gesprochen, hat sie eine Menge an Elementen (z.B. die Klassenkameraden) und paarweise Verbindungen zwischen diesen Elementen (z.B. Freundschaftsbeziehungen). Diese Art von Konstruktionen behandelt die Graphentheorie, in der man von Knoten und Kanten spricht.

Definition. Ein Graph $G = (V, E)$ besteht aus einer Menge V von Knoten (engl. *vertices*) und Kanten E (engl. *edges*), wobei jede Kante $e \in E$ zwei Knoten u, v aus V miteinander verbindet.

Eine Kante zwischen den Knoten u und v kann entweder mit uv , (u, v) oder $\{u, v\}$ bezeichnet werden. Die Endknoten einer Kante $e = (u, v)$ müssen nicht verschieden sein. Gilt $u = v$, so spricht man von einer Schlinge. Haben zwei Kanten dieselben Endknoten, so sind sie parallel. Ein einfacher Graph enthält weder Schlingen noch parallele Kanten (Abb. 1.2).

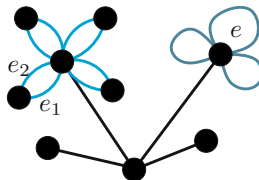


Abb. 1.2: Die Kante e ist eine Schlinge und die beiden Kanten e_1 und e_2 verlaufen parallel.

Bildlich kann man einen Graphen G (wie durch Anne schon geschehen) darstellen, indem man seine Knoten als Punkte zeichnet und zwei dieser Punkte immer dann durch eine Linie verbindet, wenn die entsprechenden Knoten eine Kante bilden. Wie man diese Punkte und Linien zeichnet, ob geradlinig oder geschwungen, kreuzungsfrei oder überkreuz, ist eine Frage der Zweckmäßigkeit und der Ästhetik: Ein Graph ist von seiner bildlichen Darstellung unabhängig.

Beispiel. Sei $G = (V, E)$ ein Graph mit $V = \{a, b, c, d, e\}$, der Knotenmenge, und $E = \{\{a, b\}, \{a, e\}, \{b, d\}, \{c, d\}, \{c, e\}, \{e, d\}\}$, der Kantenmenge. Welche der folgenden fünf Darstellungen zeigen den Graphen G (Abb. 1.3)? Fallen Ihnen weitere Darstellungen ein? Wie viele unterschiedliche Darstellungen gibt es? ■

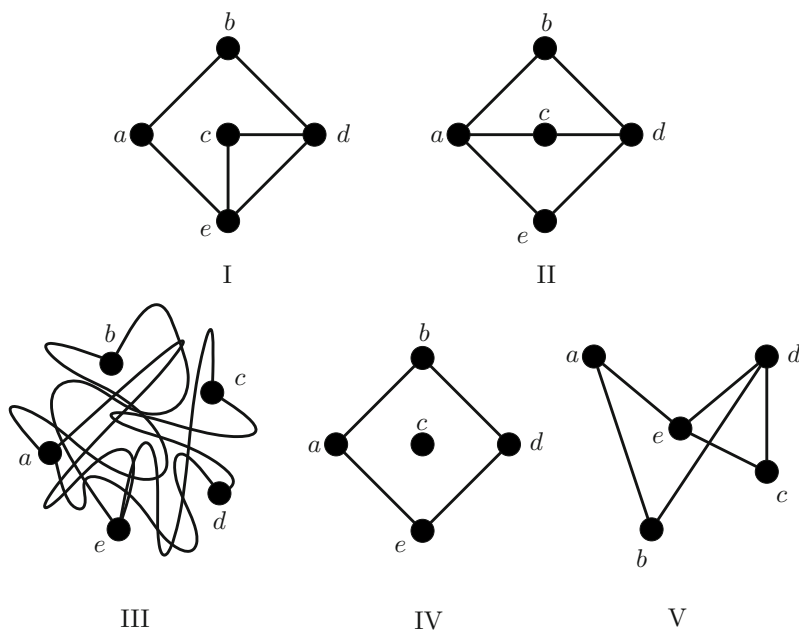


Abb. 1.3: Ein Graph $G = (V, E)$ ist unabhängig von seiner bildlichen Darstellung.

Durch die allgemeine Definition von Graphen können diese in jeder Situation verwendet werden, in der eine Beziehung (Kante) zwischen bestimmten Objekten (Knoten) definiert ist. Sie werden beispielsweise zur Beschreibung von Bindungen zwischen den Atomen innerhalb eines Moleküls, von Verbindungen zwischen Gehirnzellen oder zur Beschreibung von Stammbäumen verschiedener Spezies verwendet. Manchmal repräsentieren Knoten auch abstraktere Dinge: Sie können zum Beispiel die verschiedenen Stadien eines großen Bauprojekts darstellen, wobei eine Kante zwischen zwei Stadien, den Knoten, bedeutet, dass das eine aus dem anderen durch eine einzige Arbeitsphase hervorgeht. Um die Richtung des Arbeitsschritts anzudeuten, arbeitet man in diesem Fall auch mit gerichteten Graphen. Mit ih-

nen werden wir uns später beschäftigen. Die Knoten können auch alle möglichen Positionen in einem Spiel (z.B. Schach) repräsentieren, wobei zwei Knoten durch eine Kante verbunden werden, wenn die eine Spielsituation aus der anderen durch einen einzigen Zug hervorgehen kann. Wie müssten die verschiedenen Spielsituationen von Tic-Tac-Toe aus der Abbildung 1.4 miteinander verbunden werden? (Es sind hier nicht alle Spielstände dargestellt. Das wären weit über 3000.)

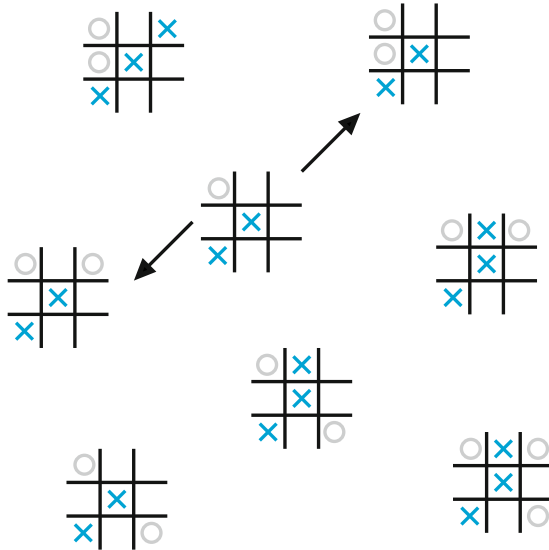


Abb. 1.4: Bei Tic-Tac-Toe spielen zwei Spieler (Kreuz und Kreis) gegeneinander. Ziel ist es, eine Vertikale, Horizontale oder Diagonale mit den eigenen Symbolen zu füllen. Dazu darf abwechselnd ein noch nicht belegtes Feld markiert werden.

Für die meisten Probleme gibt es eine einfache Sprechweise, um die Beziehung zwischen zwei Objekten zu beschreiben. Zum Beispiel kann man sagen, Andreas und Berta sind miteinander befreundet, oder Spielstand 1 lässt sich in Spielstand 2 übertragen. Für die Graphen müssen wir eine solche Sprechweise erst definieren.

Definition. Sei $G = (V, E)$ ein Graph mit $u, v \in V$ und $e \in E$. Ein Knoten v und eine Kante e **inzidieren** miteinander und heißen **inzident**, wenn v ein Endknoten von e ist. Gilt $u, v \in e$, dann sind u und v **adjazent** oder **benachbart** in G und heißen **Nachbarn**. Zwei Kanten $e, f \in E$ mit $e \neq f$ heißen **adjazent** oder **benachbart** in G , wenn sie einen gemeinsamen Knoten haben.

Die Frage „mit wem ist Andreas befreundet“ überträgt sich dann zu „zu welchen Knoten ist A adjazent“. Anne stellt sich aber nicht nur diese Frage zu ihrer neuen Klasse, sondern überlegt sich auch wer am beliebtesten ist. Anhand der Zeichnung (Abb. 1.1) lässt sich das leicht feststellen: Der Knoten mit den meisten inzidenten

Kanten entspricht der beliebtesten Person. Wer ist das in ihrer Klasse? Und wer steht eher unten in der Beliebtheitsskala?



Auch in der Graphentheorie interessiert man sich häufig für die „Beliebtheit“ eines Knoten, bzw. zu wie vielen Kanten ein Knoten v inzident ist. Diesen Wert bezeichnet man als Grad von v .

Definition. Sei $G = (V, E)$ ein Graph. Der **Grad** $d(v)$ eines Knotens $v \in V$ ist die Anzahl des Auftretens von v als Endknoten einer Kante. Ist G ein einfacher Graph, so entspricht der Grad von v der Anzahl der Nachbarn von v . Der Wert $\delta(G)$ gibt den **Minimalgrad** in G an, d.h. die kleinste in G vorkommende Gradzahl, und $\Delta(G)$ den **Maximalgrad** eines Knoten in G , d.h. die größte in G vorkommende Gradzahl.

Ein Knoten mit einem Maximalgrad entspricht also der beliebtesten Person, einer mit Minimalgrad hat in dieser Gruppe die wenigsten Freunde.

Graphen können auch über die eben eingeführten Eigenschaften beschrieben werden.

Beispiel. Der Graph G habe die folgenden Eigenschaften: Die Anzahl der Knoten von G sei 5. Der Minimalgrad von G habe den Wert 1 und $\Delta(G) = 3$. Jede Kante sei mindestens zu einer weiteren adjazent. Finden Sie einen Graphen G , der diese Eigenschaften erfüllt. Gibt es mehrere unterschiedliche Graphen, die so beschrieben werden können?



Nehmen wir an, von einem einfachen Graphen sind die Anzahl der Knoten und die in dem Graph vorkommenden Gradzahlen bekannt. Ist der Graph damit eindeutig beschrieben, d.h. existiert genau ein Graph, der diese Eigenschaften erfüllt?

Gibt es einen einfachen Graphen mit fünf Knoten und den Knotengraden 4, 3, 3, 2, 2? ■

Bevor wir zu den ersten Aussagen und Ergebnissen für Graphen kommen, wollen wir hier einige allgemeine Voraussetzungen festlegen:

- Die Anzahl der Knoten eines Graphen bezeichnen wir mit n .
- Die Anzahl der Kanten eines Graphen bezeichnen wir mit m .
- Wenn nichts anderes vorgegeben ist, beschäftigen wir uns mit *einfachen Graphen* G , d.h. zwischen zwei Knoten verläuft höchstens eine Kante und die Endknoten einer jeden Kante sind verschieden.

Jetzt können wir mit der ersten wichtigen Aussage beginnen:

Lemma 1.1 (*Handshaking-Lemma*). Die Summe der Knotengrade entspricht zweimal der Anzahl der Kanten, d.h.

$$\sum_{v \in V} d(v) = 2 \cdot |E| = 2m.$$



Wieso gilt diese Gleichung? Vielleicht hilft es Ihnen, wenn Sie sich die Entstehung des Namens „*Handshaking-Lemma*“ überlegen.

Aus dem *Handshaking-Lemma* folgt sofort:

Lemma 1.2. *In einem Graphen ist die Anzahl der Knoten mit ungeradem Grad gerade.*



Vielleicht ist die Formulierung mit den vielen „gerade“ und „ungerade“ erst einmal etwas verwirrend. Schauen Sie sich die Aussage am Freundschaftsgraphen (Abb. 1.1) an. Wie viele Personen haben eine ungerade Anzahl an Freunden? Können Sie durch das Hinzufügen von Kanten erreichen, dass dieser Personenkreis sich erweitert? Wie viele können maximal dazu gehören?

Beweis von Lemma 1.2. Wir wissen, dass die Summe aller Grade zweimal die Anzahl der Kanten ergibt, da jede Kante zweimal gezählt wird: $\sum_{v \in V} d(v) = 2|E|$. Also ist die Summe aller Grade eine gerade Zahl. Sei V_1 die Menge der Knoten mit ungeradem Grad und V_2 die Menge der Knoten mit geradem Grad. Dann gilt

$$2m = \sum_{v \in V} d(v) = \sum_{v \in V_1} d(v) + \sum_{v \in V_2} d(v).$$

Etwas umgeformt gilt

$$\underbrace{2m - \sum_{v \in V_2} \overbrace{d(v)}^{\text{gerade}}}_{\text{gerade}} = \sum_{v \in V_1} \underbrace{d(v)}_{\text{ungerade}}.$$

Die Aussage lautet in Worten: Da für jeden Knoten $v \in V_2$ der Grad $d(v)$ gerade ist, ist auch $\sum_{v \in V_2} d(v)$ und damit die linke Seite eine gerade Zahl. Damit muss auch $\sum_{v \in V_1} d(v)$ (eine Summe über ungerade Zahlen) gerade sein. Da nur eine gerade Anzahl von ungeraden Summanden gerade ist, muss die Anzahl der Knoten in V_1 gerade sein. \square



Mit Hilfe von Lemma 1.2 ist sofort klar, dass es keinen Graphen mit den Knotengraden 1, 2, 3, 4, 5 geben kann.

Als Nächstes schauen wir uns ein paar wichtige Graphenklassen an.

Beispiel. In der Graphentheorie tauchen immer wieder ähnliche Graphen auf. Hier werden einige davon vorgestellt:

- **Vollständige Graphen:** Ein Graph $G = (V, E)$ heißt **vollständig**, wenn jeder Knoten zu jedem anderen Knoten benachbart ist. Da der Graph also nur von der Anzahl der Knoten n abhängt, bezeichnet man ihn auch als K_n . In der Abbildung 1.5 sind der K_1, K_2, K_3, K_4 und K_5 abgebildet. Welchen Wert hat der Grad $d(v)$ für einen Knoten $v \in K_n$ und wie viele Kanten hat ein vollständiger Graph K_n ?¹



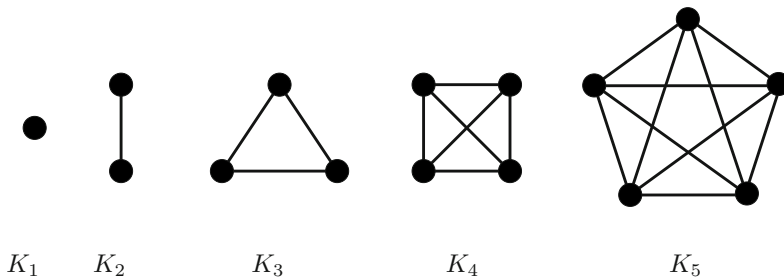


Abb. 1.5: In einem vollständigen Graphen ist jeder Knoten mit allen anderen Knoten benachbart.

- **Bipartite Graphen:** Ein Graph $G = (V, E)$ heißt **bipartit**, wenn seine Knoten in zwei Teilmengen A und B zerlegt werden können, sodass nur Kanten zwischen den beiden Teilmengen und nicht innerhalb von einer Knotenmenge verlaufen (Abb. 1.6 Graph G_1). Sind die Graphen G_2 und G_3 bipartit?

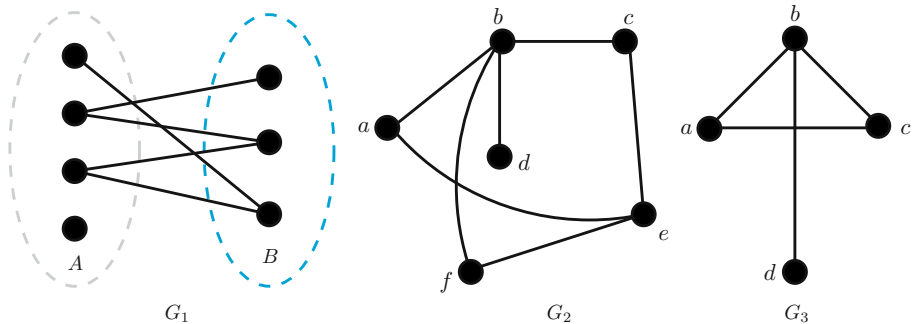


Abb. 1.6: Der Graph G_1 ist bipartit. Versuchen Sie dazu die Graphen G_2 und G_3 so zu zeichnen, dass die Knoten in zwei Kreise passen und nur Kanten zwischen diesen Kreisen verlaufen.

Kann es in einem bipartiten Graphen mehrere Möglichkeiten geben, die Mengen A und B zu definieren? Wie viele Kanten kann ein bipartiter Graph höchstens enthalten?² Ein bipartiter Graph mit der maximalen Anzahl an Kanten heißt auch **vollständiger bipartiter Graph**. Da ein vollständiger bipartiter Graph durch die Größen der Mengen A und B schon vollständig beschrieben ist, wird er auch als $K_{|A|,|B|}$ bezeichnet. Bipartite Graphen eignen sich vorzüglich für die Behandlung von Zuordnungsproblemen wie dies bei Arbeitnehmern auf Arbeitsplätze oder von Kindern auf Kindergärten der Fall ist.

- Zwei Graphen, die immer wieder als Beispiel benutzt werden, sind der **Petersen-Graph** und der **Würfelgraph** (Abb. 1.7). Finden Sie eine Zeichnung des Würfelgraphen, in der sich keine zwei Kanten kreuzen? Gibt es eine solche Zeichnung auch für den Petersen-Graph? Existiert eine solche Darstellung für einen Graphen, so bezeichnet man diesen als **planar**.



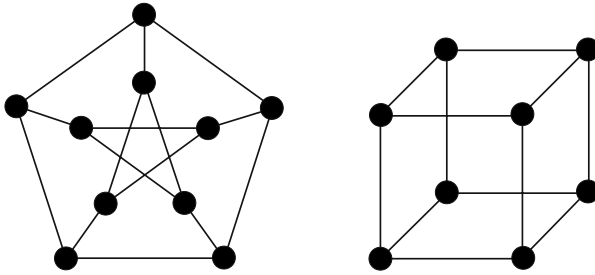


Abb. 1.7: Der Petersen- (links) und der Würfelgraph (rechts).

Nicht immer möchte man den ganzen Graphen G betrachten. Wenn Anne zum Beispiel wissen möchte, wie sich denn die Mädchen in ihrer Klasse untereinander so verstehen, reicht es aus, sich nur diesen Teil des Graphen anzuschauen. Formal definieren wir:

Definition. Sei $G = (V, E)$ ein Graph. Ein Graph $G' = (V', E')$ ist ein **Teilgraph** von G , wenn V' eine Teilmenge von V ist und jede Kante aus E' auch in E enthalten ist. Enthält E' jede Kante aus E , die zwei Knoten aus V' verbindet, so spricht man von einem von $V' \subseteq V$ **induzierten Teilgraphen** oder auch **Untergraphen** von G (Abb. 1.8).

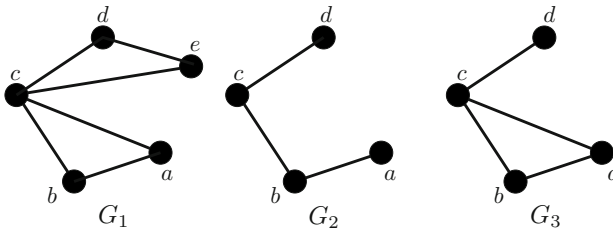


Abb. 1.8: Die Graphen G_2 und G_3 sind Teilgraphen von G_1 . Dabei ist G_3 ein Untergraph und G_2 nicht.



Wie sieht der von den Mädchen induzierte Teilgraph des Freundschaftsgraphen (Abb. 1.1) aus? Hat sich der Maximalgrad in diesem Teilgraphen verändert? Wie sieht es mit dem Minimalgrad aus?

Eine wichtige Klasse von Teilgraphen sind Wege und Kreise. Dabei entspricht die Definition eines Wegs auch der ersten Intuition: Man startet in einem Knoten, läuft dann eine Kante entlang bis zum zweiten Endknoten der Kante. Von dort sucht man sich eine weitere Kante und setzt so nach und nach den „Weg“ durch den Graphen fort. Bei einem Kreis landet man am Ende wieder dort, wo man begonnen hat.

Definition. Ein **Kantenzug** Z in G ist eine Folge von Knoten und Kanten aus G , die sich wie folgt schreiben lässt:

$$Z = x_0 e_0 x_1 e_1 x_2 e_2 \dots x_{k-1} e_{k-1} x_k$$

mit $e_i = (x_i, x_{i+1})$. (Die Definition schließt weder doppelte Knoten noch doppelte Kanten aus.)

Ein Kantenzug heißt **Weg**, wenn alle Kanten verschieden sind. Die Knoten x_0 und x_k eines Wegs p heißen **Endknoten** von p , und die Anzahl der Kanten k gibt die **Länge** des Wegs an. Ein Weg heißt **einfach**, wenn kein Knoten doppelt durchlaufen wird. Ein **Kreis** ist ein Weg mit gleichem Anfangs- und Endknoten.

Ein **einfacher Kreis** ist ein Kreis, in dem außer den Endknoten kein Knoten doppelt durchlaufen wird (Abb. 1.9).

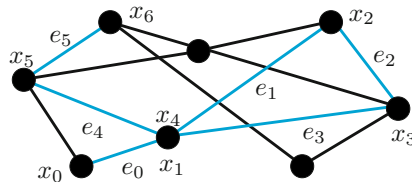


Abb. 1.9: Ein Knoten kann in einem Weg p durchaus zwei unterschiedliche Bezeichnungen (hier x_1 und x_4) bekommen.

Wie viele einfache Wege verlaufen zwischen A (Andreas) und D (Doris) im Freundschftsgraphen (Abb. 1.1)? Welchen würden Sie auswählen, wenn Sie an Andreas Stelle Doris eine Nachricht übermitteln wollen? Kreise und Wege werden nicht nur zum Verschicken von Nachrichten, sondern z.B. auch beim Kürzesten-Wege-Problem, bei Flussproblemen und bei vielen anderen Anwendungen benötigt.

Meistens gibt man bei einem Weg nur die Knoten $x_0 x_1 \dots x_k$ oder die Kanten $e_1 e_2 \dots e_k$ an, die er durchläuft. Nehmen wir an, p sei ein einfacher Weg, der die Knoten a und b verbindet, und q ein einfacher Weg, der b und c verbindet. Existiert dann ein einfacher Weg zwischen a und c ?³ Die Schwierigkeit liegt darin, dass die beiden Wege p und q ja durchaus die gleichen Knoten enthalten können. Als nächstes können Sie sich überlegen: Wenn G einen Weg zwischen zwei Knoten a und b enthält, dann existiert auch ein einfacher Weg von a nach b in G .

Der folgende Satz sagt aus, dass ein Graph mit einem hohen Minimalgrad auch lange Wege und Kreise enthält. Die Aussage entspricht folgender Intuition: Wenn in einem Graphen jeder Knoten mit vielen anderen Knoten benachbart ist, so muss auch ein langer Weg in dem Graphen existieren.

Satz 1.3. Jeder Graph G enthält einen einfachen Weg der Länge $\delta(G)$. Falls $\delta(G) \geq 2$ gilt, dann enthält er auch einen einfachen Kreis von mindestens der Länge $\delta(G) + 1$. Dabei gibt $\delta(G)$ den Minimalgrad von G an (Definition auf Seite 5).

Beweis: Es sei $p = x_0 \dots x_k$ ein längster einfacher Weg in G . Angenommen, ein Nachbar a von x_k liegt nicht auf diesem Weg. Dann kann der Weg p um die Kante (x_k, a) verlängert werden. Der neue Weg ist wiederum ein einfacher Weg und ist länger als p . Das ist aber ein Widerspruch zu der Annahme, dass p ein längster einfacher Weg in G ist. Also liegen alle Nachbarn von x_k auf p und somit gilt $k \geq d(x_k) \geq \delta(G)$. Damit ist die erste Behauptung bewiesen.

Sei nun x_i der Nachbar von x_0 , der am weitesten von x_0 entfernt liegt (Abb. 1.10). Da alle Nachbarn von x_0 auf dem Weg $p = x_0 \dots x_k$ liegen, hat der einfache Weg $x_0 x_1 \dots x_i$ mindestens die Länge $\delta(G) \geq 2$ und der einfache Kreis $x_0 \dots x_i x_0$ mindestens die Länge $\delta(G) + 1 \geq 3$. In dem so entstandenen Kreis wird keine Kante und kein Knoten doppelt durchlaufen, womit auch die zweite Aussage bewiesen ist. □

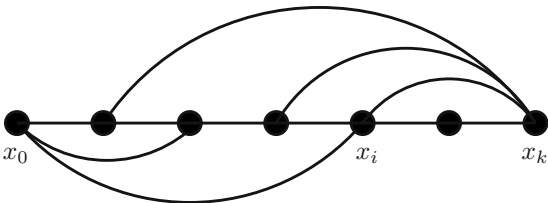


Abb. 1.10: Wir betrachten einen längsten Weg $p = x_0 \dots x_k$ von G wie abgebildet. Alle Nachbarn von x_k müssen auf p liegen. Der Knoten x_i ist der Nachbar, der von x_0 am weitesten auf dem Weg p entfernt liegt.

1.2 Zusammenhang und Schnitte

Sie haben sicher schon festgestellt, dass der Freundschaftsgraph quasi aus zwei Teilen besteht: Zum einen aus dem Teilgraphen der Claudia enthält und zum anderen aus dem um Hans. Der Graph ist also nicht zusammenhängend. Der Zusammenhang ist einer der Schlüsselbegriffe in der Graphentheorie. Intuitiv ist dessen Bedeutung ohnehin klar (Abb. 1.11). Aber auch eine formale Definition dieser Eigenschaft gelingt schnell.

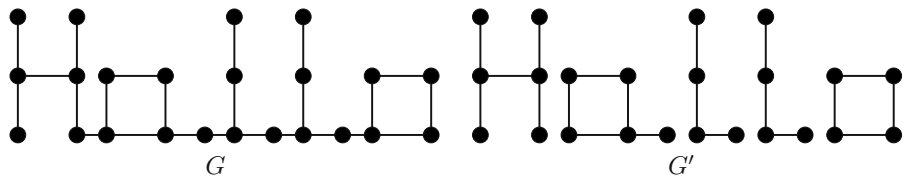


Abb. 1.11: Der Graph G ist zusammenhängend. In G' ist jeder Buchstabe eine Zusammenhangskomponente. Zum Beispiel führt kein Weg in G' von dem Buchstaben H zum Buchstaben a .

Definition. Ein Graph G heißt **zusammenhängend**, wenn zu je zwei Knoten u, v von G ein Weg von u nach v existiert. Ist ein Graph nicht zusammenhängend, heißt er **unzusammenhängend**. Eine **Zusammenhangskomponente** H von G ist ein maximal zusammenhängender Teilgraph von G .

Mit anderen Worten: H ist eine Zusammenhangskomponente, wenn sie selbst zusammenhängend ist und kein anderer Knoten außerhalb von H von ihr aus über einen Weg im Graphen erreicht werden kann. Wie sehen die Zusammenhangskomponenten in dem folgenden Graphen aus (Abb. 1.12)?

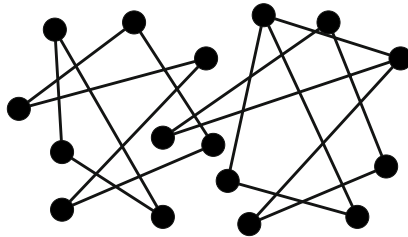


Abb. 1.12: Ist der Graph zusammenhängend? Wie viele Zusammenhangskomponenten gibt es?

Betrachten wir die Klasse 8b. Von der Klasse 8b wissen Sie, dass alle Mädchen miteinander befreundet sind, und auch die Jungen bilden eine große Clique, die jeden Mittwoch zusammen Fußball spielen geht. Zwischen den beiden Gruppen gibt es keine großen Berührungspunkte. Nur der Bäckersfrau nahe der Schule vertrauen die Schülerinnen immer wieder ihre Probleme an und auch die Jungen verstecken sich gut mit ihr, weil sie hin und wieder Gummibärchen verteilt (Abb. 1.13). Seit ein paar Wochen hat Jessica jedoch ein Auge auf Markus geworfen und fragt sich, wie sie ihm am geschicktesten eine Nachricht zukommen lassen kann. Ist das möglich?

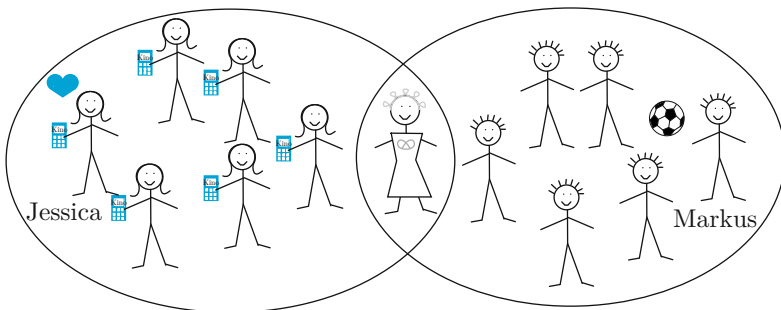


Abb. 1.13: Ob Jessica eine Chance hat Markus kennen zu lernen?

In der Graphentheorie wird die Situation durch folgendes Lemma repräsentiert. Jessica kann sich also Hoffnungen machen. Die Kontaktaufnahme klappt schon mal.

Lemma 1.4. *Sei G ein Graph und seien $H_1 = (V_1, E_1)$ und $H_2 = (V_2, E_2)$ zwei jeweils zusammenhängende Teilgraphen von G . Wir nehmen an, dass H_1 und H_2 mindestens einen Knoten gemeinsam haben. Sei H die Vereinigung der beiden Graphen H_1 und H_2 , d.h. $H = (V', E')$ mit $V' = V_1 \cup V_2$ und $E' = E_1 \cup E_2$. Dann ist H zusammenhängend.*



Der Beweis zu dem Lemma ist nicht schwierig. Betrachten Sie noch einmal die Beobachtung von Seite 9 und wenden Sie diese auf die Definition des Zusammenhangs an.⁴

Aus Lemma 1.4 kann gefolgert werden, dass verschiedene Zusammenhangskomponenten von G keinen Knoten gemeinsam haben (andernfalls wäre ihre Vereinigung ein zusammenhängender Teilgraph, der sie beide enthält). Jeder Knoten von G ist also in genau einer Zusammenhangskomponente enthalten.

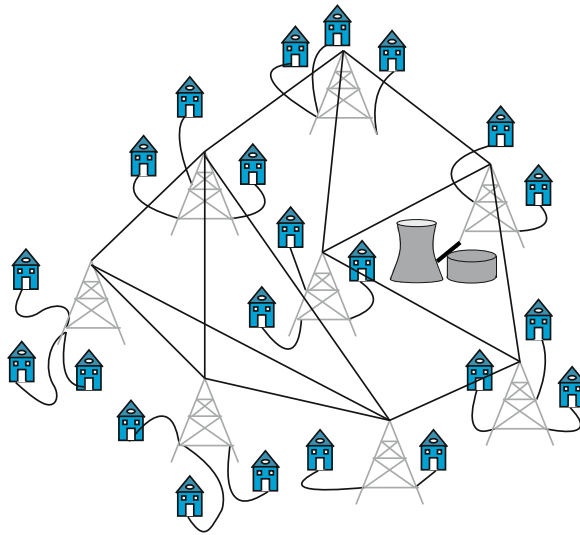


Abb. 1.14: Bei welchen Leitungen zwischen den Strommasten sollte es auf keinen Fall zu einem Ausfall kommen? Der Ausfall von einer Leitung, an der „nur“ ein Haus hängt, lässt sich nach Meinung der Stromfirma verkraften und die Leitung vom Kraftwerk ins Netz ist sicher.

Neben dem Zusammenhang eines Graphen interessiert man sich auch dafür, was beim Ausfall von bestimmten Knoten oder Kanten passiert. Bricht der Graph dann auseinander oder bleibt er zusammenhängend? Wie viele Kanten oder Knoten müssen gelöscht werden, bis der Graph in zwei Komponenten zerfällt? Dazu definieren wir $G - v$ als den Graphen G ohne den Knoten v und allen Kanten, die v als Endknoten haben. Der Graph $G - e$ entsteht durch das Löschen der Kante e aus dem Graphen G . Die beiden Endknoten von e werden dabei nicht mitgelöscht. Führt ein Knoten v beim Löschen dazu, dass eine neue Zusammenhangskomponente entsteht, so nennt man ihn einen **Artikulationspunkt**. Passiert dies bei einer

Kante, so nennt man sie auch eine **Brücke**. In Stromnetzen, in denen Knoten die Strommasten und Kanten die verbindenden Leitungen darstellen, versucht man Brücken und Artikulationspunkte zu vermeiden. Sollte nämlich eine Leitung ausfallen (was immer wieder passieren kann), so fließt in einem Teil des Netzes kein Strom mehr (Abb. 1.14). Enthält der Graph hingegen keine Brücke, so ist der Ausfall einer Leitung nicht so schlimm.

Eine Charakterisierung einer Brücke gibt das folgende Lemma.

Lemma 1.5. *Eine Kante ist genau dann eine Brücke, wenn sie auf keinem Kreis liegt.*

Beweis: Hin-Richtung (" \Rightarrow "): Sei $e = (u, v)$ eine Brücke. Angenommen, es existiert ein Kreis mit der Kante e . Dann gibt es auch einen Weg von u nach v in $G - e$. Durch das Löschen von e wird ja nur die direkte Verbindung zwischen u und v gekappt. Dann ist aber durch das Löschen von e keine neue Zusammenhangskomponente entstanden. Damit haben wir einen Widerspruch, dazu, dass e eine Brücke ist. Also kann e auf keinem Kreis liegen.

Rück-Richtung (" \Leftarrow "): Nehmen wir nun an, dass die Kante e auf keinem Kreis liegt. In G sind die Knoten u und v miteinander verbunden. Betrachten wir den Graphen $G - e$. Was wäre, wenn es doch einen Weg zwischen u und v geben würde? Also liegen u und v in unterschiedlichen Zusammenhangskomponenten. Damit ist e eine Brücke. □

Mit anderen Worten besagt das Lemma: Eine Kante e , die auf einem Kreis liegt, kann keine Brücke sein. Weiter gilt, wenn e eine Brücke und v einer ihrer Endknoten ist, dann ist v auch ein Artikulationspunkt. Gilt die Aussage auch umgekehrt? Das Konzept einer Brücke, d.h. einer Kante, die den Graphen „auseinander brechen“ lässt, kann durch den Begriff des Schnittes noch verallgemeinert werden. Dabei kann man sich vorstellen, dass man mit einer Schere nach und nach Kanten durchschneidet, bis der Graph in zwei Teile zerfällt. Die formale Definition sieht etwas kompliziert aus, da nicht mehr Kanten als unbedingt notwendig zerschnitten werden sollen (Abb. 1.15).

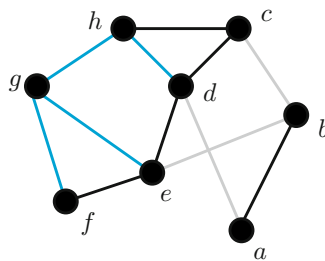


Abb. 1.15: Die grauen Kanten sind ein Schnitt der Menge $\{a, b\}$. Die blaue Kantenmenge gehört zu keinem Schnitt. Ohne die Kante (h, d) wäre dies jedoch der Schnitt $\delta(\{g\})$.

Definition. Sei $G = (V, E)$ ein Graph und $\emptyset \neq X \subsetneq V$ eine Teilmenge der Knoten. Der **Schnitt** $\delta(X)$ ist die Menge aller Kanten, die je einen Endknoten in X und $V \setminus X$ haben, d.h.

$$\delta(X) := \{(u, v) \in E \mid u \in X, v \in V \setminus X\}.$$

Vielleicht haben Sie gerade gestutzt, da die Bezeichnung $\delta(G)$ auch den Minimalgrad eines Graphen bezeichnet. Allerdings lässt sich aus dem Kontext immer feststellen, ob gerade der Minimalgrad $\delta(G)$ oder ein Schnitt $\delta(X)$ gemeint ist.



Stellt sich nun natürlich die Frage nach der Ausfallsicherheit in dem Stromnetz (Abb. 1.14)? Die Ausfallsicherheit wird dabei anhand der Anzahl der Leitungen gemessen, die ausfallen müssen, damit das Netzwerk nicht mehr zusammenhängend ist.

Zwischen Schnitten und Zusammenhang besteht eine einfache Korrelation.

Lemma 1.6. *Ein Graph $G = (V, E)$ ist genau dann zusammenhängend, wenn jeder Schnitt $\delta(X)$ zu einer beliebigen Menge $\emptyset \neq X \subsetneq V$ mindestens eine Kante enthält.*

Beweis: Wir beginnen dieses Mal mit der Rück-Richtung: Angenommen G ist nicht zusammenhängend und sei X die Knotenmenge einer Zusammenhangskomponente von G . Dann gilt $\delta(X) = \emptyset$. Das ist ein Widerspruch zur Voraussetzung. Sei nun G zusammenhängend und sei X eine beliebige nicht leere, echte Teilmenge von V . Sei $s \in X$ und $t \in V \setminus X$. Da G zusammenhängend ist, existiert ein Weg von s nach t . Damit muss mindestens eine Kante in der Menge $\delta(X)$ enthalten sein. \square

Schnitte werden uns später insbesondere bei Flüssen noch einmal begegnen.

1.3 Bäume

Die Theorie der Bäume wurde ursprünglich aus dem Studium der Kohlenwasserstoffverbindungen und anderer Isomere entwickelt. Cayley warf Ende des vorigen Jahrhunderts die Frage auf, wie viele verschiedene Isomere einer bestimmten Zusammensetzung existieren. Dies führte zur Abzähltheorie von Graphen. Des Weiteren bilden Bäume die fundamentalen Bausteine für alle Graphen. Sie sind nicht nur als Graphen interessant, sondern sie ergeben auch die geeignete Datenstruktur für viele diskrete Probleme – insbesondere für Such- und Sortierprobleme.

Definition. Ein Graph heißt **Baum**, falls er zusammenhängend und kreisfrei ist. Ein Graph ist **kreisfrei**, wenn er keinen Kreis enthält. Die Knoten vom Grad 1 eines Baums heißen **Blätter**.

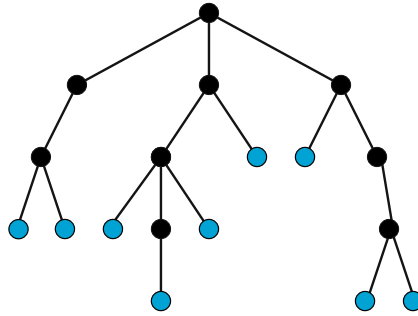


Abb. 1.16: Im Gegensatz zu den Bäumen in der Natur werden Bäume in der Graphentheorie meist von oben nach unten gezeichnet. Die blauen Knoten bilden die Blätter des Baums.

Man sollte sich klar machen, dass die beiden Eigenschaften, durch welche Bäume definiert werden, entgegengesetzt wirken: Zusammenhang bedeutet, der Graph kann nicht „zu wenig“ Kanten besitzen, während der Ausschluss von Kreisen beinhaltet, dass der Graph nicht „zu viele“ Kanten besitzen darf. Betrachten wir nun einige wichtige Eigenschaften von Bäumen.

Lemma 1.7. *Sei G ein Baum mit mindestens zwei Knoten. Dann besitzt G mindestens einen Knoten vom Grad 1, also mindestens ein Blatt.*

Versuchen Sie den Beweis selber zu führen. Ansatz: Bestimmen Sie einen beliebigen Knoten $x_0 \in G$ und einen nicht mehr verlängerbaren Weg.⁵



Weiter gibt es viele Kriterien, die nur Bäume erfüllen. Der folgende Satz gibt einige an.

Satz 1.8. *Sei G ein Graph mit n Knoten. Folgende Aussagen sind äquivalent:*

1. G ist ein Baum (d.h. kreisfrei und zusammenhängend).
2. G ist kreisfrei und hat $n - 1$ Kanten.
3. G ist zusammenhängend und hat $n - 1$ Kanten.

Den Beweis führen wir über einen Ringschluss, d.h. wir zeigen: Aus (1) folgt (2), aus (2) folgt (3) und aus (3) folgt (1). Man kann dies leicht anhand eines Graphen darstellen (Abb. 1.17). Mit dem Ringschluss können wir uns die Richtungen aus (2) folgt (1), aus (1) folgt (3) und aus (3) folgt (2) sparen.

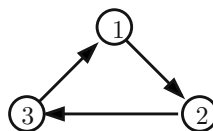


Abb. 1.17: Sollen mehrere Äquivalenzen gezeigt werden, erspart man sich über einen Ringschluss einiges an Arbeit.

Der wichtigste Teil des Beweis steckt in folgendem Lemma:

Lemma 1.9. *Sei G kreisfrei mit n Knoten, m Kanten und p Zusammenhangskomponenten. Dann gilt $n = m + p$.*

Beweis: Den Beweis führen wir über Induktion nach der Anzahl der Kanten m .

Induktionsanfang: Sei $m = 0$. Dann bildet jeder Knoten eine eigene Zusammenhangskomponente. Also gilt $n = p$.

Induktionsvoraussetzung: Ein kreisfreier Graph mit n Knoten und m Kanten hat $p = n - m$ Zusammenhangskomponenten.

Induktionsschluss von m auf $m + 1$: Sei G ein kreisfreier Graph mit n Knoten, $m + 1$ Kanten und p Zusammenhangskomponenten. Wir wollen im Folgenden zeigen, dass $n = (m + 1) + p$ gilt.

Das Löschen einer Kante e aus G erhöht die Anzahl der Zusammenhangskomponenten um 1. Der Graph $G - e$ hat dann n Knoten, m Kanten, $p + 1$ Zusammenhangskomponenten und ist kreisfrei. Auf diesen Graphen können wir die Induktionsvoraussetzung anwenden und erhalten

$$n = m + (p + 1) = (m + 1) + p.$$

Somit haben wir die Behauptung auch für G bewiesen. □

Jetzt können wir uns an den Beweis von Satz 1.8 machen:

Beweis von Satz 1.8. (1) \Rightarrow (2): Der Graph $G = (V, E)$ ist ein Baum und wir müssen zeigen, dass er kreisfrei ist und $n - 1$ Kanten hat. Nach der Definition von Bäumen ist G zusammenhängend und kreisfrei. Es reicht also $m = n - 1$ zu zeigen. Aus Lemma 1.9 wissen wir, dass für kreisfreie Graphen $n = m + p$ bzw. $m = n - p$ gilt. Da in unserem Fall $p = 1$ gilt, erhalten wir $m = n - 1$. Damit ist (2) bewiesen.

(2) \Rightarrow (3): Der Graph G ist kreisfrei mit $n - 1$ Kanten. Wir müssen zeigen, dass G zusammenhängend ist, also für die Anzahl der Zusammenhangskomponenten $p = 1$ gilt. Dies folgt aus der Behauptung 1.9, denn $p = n - m = 1$.

(3) \Rightarrow (1): Der Graph G ist zusammenhängend mit $n - 1$ Kanten. Zu zeigen ist, dass G kreisfrei ist. Wir nehmen an, dass der Graph einen Kreis enthält. Dann gibt es eine Kante e , sodass $G - e$ weiter zusammenhängend ist. Wir entfernen so lange Kanten aus dem Graphen G , bis wir einen Graphen G' erhalten, der weiterhin zusammenhängend, aber auch kreisfrei ist. Wenden wir die Behauptung 1.9 an, so ergibt sich $m' = n - 1$. Nach der Voraussetzung gilt auch $m = n - 1$. Daraus folgt, gilt $m' = m$, ein Widerspruch. □

Da Bäume viele besondere Eigenschaften haben und die meisten Probleme auf ihnen leicht zu lösen sind, versucht man allgemeine Graphen auf Bäume zu reduzieren. Auch beim folgenden Problem ist es sinnvoll, sich einen Baum anzuschauen:

Die Lehrerin der Klasse 7a (Sie erinnern sich an die neue Klasse von Anne) möchte am nächsten Tag einen Ausflug mit ihrer Klasse machen. Da die Abfahrtszeit des Busses noch nicht genau feststeht, organisiert sie eine Telefonkette. Dazu verteilt sie an die einzelnen Schüler Listen mit Namen von Schülern, die anzurufen sind. Auf den Listen sollen jeweils nur Freunde des Schülers stehen. Wie könnten solche Listen aussehen und wie viele Personen muss die Lehrerin mindestens anrufen? Wie viele Anrufe müssen mindestens getätigt werden? Mit dieser Art von Problemen werden wir uns im nächsten Kapitel noch ausführlicher beschäftigen.



1.4 Aufgaben

Aufgabe 1.1. Welche der folgenden Mengensysteme (V, E) sind Graphen? Wie sieht eine Zeichnung von (V, E) aus, falls es ein Graph ist.

1. $V = \{a, b, c\}$ und $E = \{\{a, b, c\}\}$
2. $V = \{a, b, c\}$ und $E = \{\{a, b\}\}$
3. $V = \{c, d, e\}$ und $E = \{\{c, d\}, \{f, d\}\}$
4. $V = \{a, b, c\}$ und $E = \emptyset$

Aufgabe 1.2. Fussballtrainer Michael ist seit einiger Zeit sehr unzufrieden mit seinen Jungs. Ständig liegen sie sich in den Haaren, von Teamgeist ist fast nichts mehr zu spüren. Um sich ein besseres Bild von der Situation machen zu können, hat er folgende Streittabelle aufgestellt (Tab. 1.2).

| | Andy | Mike | Jens | Oli | Franz | Jo | Mark | Sven | Tobi |
|-------|------|------|------|-----|-------|----|------|------|------|
| Andy | a | x | | x | | x | x | | x |
| Mike | x | a | x | | | | | x | |
| Jens | | x | a | | | | | | x |
| Oli | x | | | a | x | | x | | |
| Franz | | | | x | a | x | | x | |
| Jo | x | | | | x | a | | | x |
| Mark | x | | | | | | a | | |
| Sven | | x | | | x | | | a | x |
| Tobi | x | | x | | | x | | x | a |

Tab. 1.2: Ständig gibt es Streit.

1. Welches ist der größte Raufbold?
2. Mit wem hat er am wenigsten Ärger?

Aufgabe 1.3. Betrachten Sie den Graphen aus der Abbildung 1.18.

- Welche Kanten sind zum Knoten v inzident?

- Welche Knoten sind mit dem Knoten w benachbart?
- Welchen Knoten bestimmen den minimalen Grad δ ?
- Welcher Knoten bestimmt den maximalen Grad Δ ?
- Wie viele Kanten sind Brücken?
- Welcher Knoten ist ein Artikulationspunkt?
- Wie viele zusätzliche Kanten werden benötigt, damit der Graph zusammenhängend wird?
- Enthält der Graph einen Kreis der Länge 4?

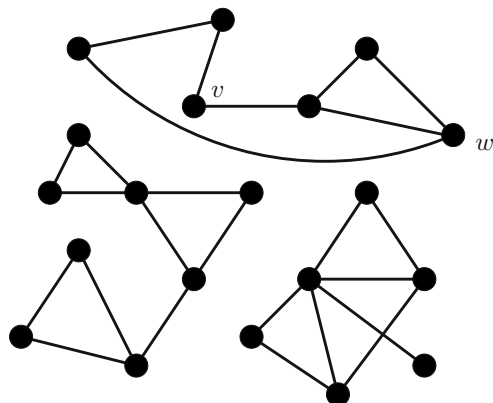


Abb. 1.18: Begriffe aus der Graphentheorie veranschaulicht man sich am besten an Bildern.

Aufgabe 1.4. Auf einer Party sind 7 Jungen und 6 Mädchen. Jeder Junge tanzt mit jedem Mädchen. Welcher Graph stellt die Tänze dar? Wie viele Kanten besitzt er? Wie lauten die Grade der Knoten?

Aufgabe 1.5. Zeichnen Sie

1. einen einfachen Graphen mit 4, 5 und 6 Knoten, sodass jeder Knoten den Grad 2 hat.
2. einen einfachen Graphen mit 4, 5 und 6 Knoten, sodass jeder Knoten den Grad 3 hat.

Aufgabe 1.6. Gibt es einen einfachen zusammenhängenden Graphen mit sechs Knoten und den Knotengraden

1. 1, 2, 3, 3, 4, 5
2. 1, 3, 3, 3, 4, 5
3. 3, 3, 4, 4, 5, 5
4. 1, 1, 1, 1, 2, 2?

Aufgabe 1.7. Welcher Baum mit acht Knoten hat genau zwei, genau vier, genau 7 und genau 8 Blätter?

Aufgabe 1.8. Wie viele Brücken und wie viele Artikulationspunkte hat ein Baum mit n Knoten?

Aufgabe 1.9. Zeigen Sie: Die Anzahl der Blätter in einem Baum ist nie kleiner als der Maximalgrad des Baumes.

Aufgabe 1.10. Seien a und b zwei Knoten des Baumes $T = (V, E)$. Dann gibt es einen eindeutigen Weg zwischen a und b .

Aufgabe 1.11. Wie viele Kanten enthält ein nicht zusammenhängender einfacher Graph mit n Knoten maximal?

Aufgabe 1.12. Gibt es einen Graphen, in dem alle Knoten unterschiedlichen Grad haben?

Lösungen zu den Fragen im Text

- 1 Der Grad an jedem Knoten beträgt $n - 1$ und die Anzahl der Kanten $\frac{n(n-1)}{2}$.
- 2 Ein bipartiter Graph kann maximal $|A| \cdot |B|$ Kanten enthalten.
- 3 Ja, es existiert dann immer auch ein Weg von a nach c . Sei p_1 der Weg von a nach b und p_2 der Weg von b nach c . Sei weiter x_i der erste Knoten auf p_1 , den die beiden Pfade gemeinsam haben. Ein solcher Knoten existiert, da c sowohl auf p_1 als auch auf p_2 liegt. Wir betrachten den (einfachen) Teilweg p_1 von a zu x_i und den (einfachen) Teilweg von p_2 von x_i zu b . Nimmt man die beiden Wege zusammen, so ergibt sich ein einfacher Weg von a nach b .
- 4 Es gilt zu zeigen, dass zwischen je zwei Knoten $u, v \in H$ ein Weg von u nach v existiert. Sei x der Knoten, den H_1 und H_2 gemeinsam haben, und $u \in V_1$ und $v \in V_2$ (alle anderen Fälle sind trivial). Dann existiert ein Weg P_1 von u nach x und ein Weg P_2 von x nach v , da H_1 und H_2 zusammenhängend sind. Der Weg $P_1 \cup P_2$ bildet dann einen Weg von u nach v in H . Damit ist die Aussage bewiesen.
- 5 Sei x_0 ein Knoten. Wählen Sie einen beliebigen Weg p in G , der von x_0 ausgeht und nicht mehr verlängerbar ist. Sei x_k der Knoten, in dem der Weg endet. Angenommen x_k ist kein Blatt. Dann existiert eine Kante von x_k zu einem Knoten, der auf p liegt. (Ansonsten könnte der Weg ja verlängert werden). Damit existiert aber ein Kreis in G . Widerspruch.

2 Tiefen- und Breitensuche

Übersicht

| | | |
|-----|---------------------------------------|----|
| 2.1 | Spannende Bäume | 21 |
| 2.2 | Wie findet man spannende Bäume? | 24 |
| 2.3 | Anwendungen von BFS und DFS | 29 |
| 2.4 | Aufgaben | 33 |

2.1 Spannende Bäume

Vor nicht allzu langer Zeit hat Karl auf dem Dachboden seiner Großmutter eine alte verstaubte Karte gefunden, auf der die Umrisse eines alten Palastes eingezeichnet sind. Nach einigen Wochen des Nachforschens in verschiedenen Bibliotheken meint er sein Ziel gefunden zu haben und reist mit einer Gruppe von Freunden zu dem vermeintlichen Ort. Dort stellen sie auch bald mit Hilfe neuer Radartechnik fest, dass tief unter der Erde ein Gebäude verschüttet liegt. Nach einigen Wochen Arbeit sind sie in den ersten Raum vorgedrungen und müssen nun etwas enttäuscht feststellen, dass alle Gänge mit einer starken Eisentür versperrt sind (Abb. 2.1).

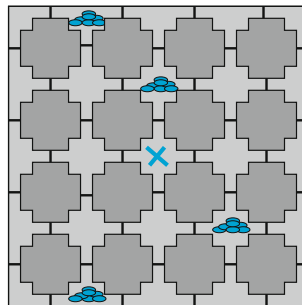


Abb. 2.1: Die Freunde sind im Zimmer mit dem blauen Kreuz gelandet. Wie können sie sicherstellen, dass ihnen der Zugang zu keinem Schatz verwehrt bleibt und sie dabei möglichst wenig Türen sprengen müssen? Die schwarzen Balken in den Gängen symbolisieren die dicken Türen.

In den Eisentüren befindet sich zwar ein Schlüsseloch, aber da es so dunkel ist, kann man den Inhalt der angrenzenden Räume nicht sehen. Zu allem Überfluss lassen sich die Schlüssel nicht finden; nun überlegen die Freunde Tag und Nacht, wie sie die Schätze wohl am besten bergen können. Immer von oben nach unten zu graben, würde viel zu lange dauern. Darauf haben sie sich schnell geeinigt. Karls nächste Idee, die Türen zu sprengen, wird jedoch mit großer Begeisterung angenommen. Jetzt brauchen sie nur noch einen Plan, wo die Sprengladungen nach und nach platziert werden. Der Sprengplan soll garantieren, dass nach seiner Durchführung alle Räume zu besichtigen sind, und dabei möglichst wenige Sprengungen enthalten. Wie sieht Ihr Vorschlag aus? Wie viele Sprengungen sind mindestens notwendig, um alle Kammern zu erreichen?



Das Problem lässt sich in die Sprache der Graphen übertragen: Die Knoten des Graphen stellen die unterschiedlichen Räume dar und die Gänge zwischen den Räumen die Kanten. Damit haben wir den Palastgrundriss modelliert. Fehlt noch der Sprengplan. In der Graphentheorie entspricht dieser einem Teilgraphen und jede Kante des Teilgraphen spiegelt einen Sprengvorgang wider. Da Max von dem mittleren Raum aus zu allen anderen Räumen gelangen will, suchen wir einen zusammenhängenden Teilgraphen, der alle Knoten enthält. Außerdem sollen möglichst wenig Sprengungen durchgeführt werden. Insbesondere Kreise kann man sich sparen. Also suchen wir nach einem Baum. Ein solcher Baum wird auch als spannender Baum bezeichnet, da er den ursprünglichen Graphen aufspannt.

Definition. Sei $G = (V, E)$ ein Graph. Ein Teilgraph $T = (V_0, E_0)$ von G , der ein Baum ist und für den $V_0 = V$ gilt, heißt **spannender Baum** (Abb. 2.2).

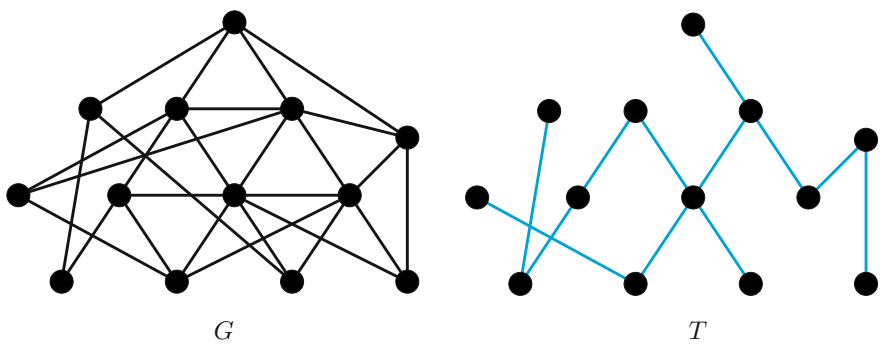


Abb. 2.2: Die blauen Kanten (rechts) bilden einen spannenden Baum in G (links).

Besitzt denn jetzt jeder Graph einen spannenden Baum, bzw. können wir Karl garantieren, dass er auf jeden Fall einen günstigen Sprengplan finden wird? In dem folgenden Graphen (Abb. 2.3) erkennt man recht schnell, dass ein spannender Baum nicht existieren kann. Wie soll der Baum denn auch die Knoten a und b miteinander verbinden, wenn zwischen den beiden Knoten schon im ursprüngli-

chen Graphen kein Weg existiert? Wie würde ein entsprechender Palastgrundriss aussehen, für den es keinen geeigneten Sprengplan gibt?

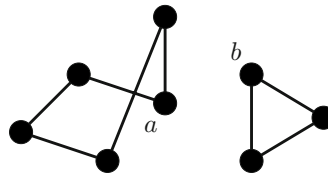


Abb. 2.3: Der Graph ist nicht zusammenhängend. Also kann auch kein spannender Baum in ihm existieren.

Wir müssen also fordern, dass der Graph zusammenhängend ist. Diese Bedingung reicht dann schon aus.

Satz 2.1. *Jeder zusammenhängende Graph enthält einen spannenden Baum.*

Beweis: Die Idee des Beweises ist es, nach und nach Kanten aus einem gegebenen Graphen G zu löschen, sodass der neue Graph zusammenhängend bleibt, aber keine Kreise mehr enthält. Sei G also ein zusammenhängender Graph. Entweder ist G bereits ein Baum, dann ist der Beweis erbracht, oder G besitzt einen Kreis C . Entfernen wir aus G eine Kante e des Kreises C , so ist $G' = G - e$ nach wie vor zusammenhängend (Abb. 2.4).

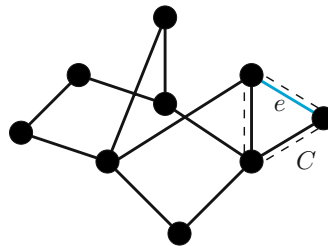


Abb. 2.4: Löscht man die blaue Kante aus dem Graphen, so ergibt sich ein neuer, zusammenhängender Teilgraph des ursprünglichen Graphen.

Entweder ist G' nun kreisfrei oder G' besitzt wieder einen Kreis C' . Im zweiten Fall entfernen wir wieder eine Kante e' aus C' . Diesen Löschvorgang wiederholen wir so lange, bis der Graph keine Kreise mehr enthält. (Wie viele Kanten müssen in einem gegebenen Graphen mit m Kanten gelöscht werden? Wie viele Kanten können es maximal sein?)⁶ Da der neue Graph dieselben Knoten hat, wie der ursprüngliche, zusammenhängend und kreisfrei ist, ist er ein spannender Baum (Theorem 1.8). \square



Der Beweis enthält schon eine Idee, wie man mit Hilfe eines Algorithmus einen spannenden Baum konstruieren kann. Allerdings ist es schwierig in einem gegebenen Graphen einen Kreis zu finden, ohne vorher einen spannenden Baum berech-

net zu haben. Im nächsten Abschnitt werden wir zwei einfache Methoden kennen lernen, einen spannenden Baum zu berechnen.

2.2 Wie findet man spannende Bäume?

Auf dem Schatzplan einen spannenden Baum zu finden, ist nicht so schwierig. Ihnen sind sicher schon einige Möglichkeiten eingefallen. Aber wie erhalten wir einen spannenden Baum, wenn uns ein viel größerer Graph gegeben ist? Der Graph basierend auf dem Stadtplan von Berlin hat zum Beispiel 10.000 Knoten und 30.000 Kanten. Wir brauchen also Regeln, mit denen wir das Problem lösen können.

Schauen wir noch einmal auf den Schatzplan und überlegen uns, wie wir hier sinnvoll einen Sprengplan konstruieren können. Eine erste Idee dazu ist die folgende: Wir starten in dem ersten Zimmer mit dem Kreuz und zünden dort eine helle Fackel an. Von dort aus betrachten wir alle Gänge nacheinander, die von diesem Zimmer aus losgehen. Als Erstes schauen wir durch das Schlüsseloch der Tür in dem Gang. Wenn wir ein Licht durch das Schlüsseloch sehen, wissen wir, dass wir in dem angrenzenden Zimmer schon einmal waren. Wir markieren den Gang mit einem dicken Kreuz, damit wir die Tür auf keinen Fall sprengen (Abb. 2.5). Ist es hingegen in dem anderen Zimmer dunkel, so waren wir noch nicht dort und sprengen die Tür. Als Nächstes kann dann ein Suchtrupp das Zimmer untersuchen und eine Fackel anzünden, während der Sprengmeister die restlichen Türen und Gänge begutachtet. Die Schatzsucher könnten auf diese Weise zu dem folgenden Netz gekommen sein (Abb. 2.5). Welche Türen würden sie danach sprengen? Hat die Wahl des nächsten Zimmers eine Auswirkung darauf, welche Türen zerstört werden?

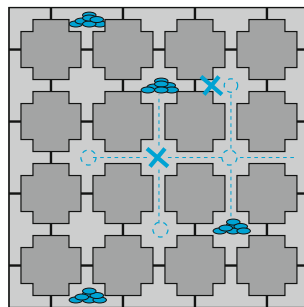


Abb. 2.5: Bisher haben die Schatzsucher erfolgreich die ersten Zimmer begutachtet. Wie sollten sie weitermachen?

Damit wir die obige Idee auch allgemein umsetzen können, sodass zum Beispiel auch ein Computer sie versteht, sollten wir sie noch einmal etwas genauer aufschreiben:

Algorithmus 2.1 Breitensuche (engl. *breadth first search*, BFS)

Input: Zusammenhängender Graph $G = (V, E)$.

Output: Aufspannender Baum $T = (V, E_0)$ von G .

- Schritt 1:
 - Sei E_0 zunächst leer.
 - Bereiten Sie zwei ebenfalls leere Knotenlisten L_1 und L_2 vor.
 - Wählen Sie einen beliebigen Startknoten $v_0 \in V$ und schreibe v_0 in L_1 und L_2 .
- Schritt 2: Falls L_1 leer ist, dann STOPP und return $T = (V, E_0)$. Ansonsten nehmen Sie den ersten Knoten v aus L_1 .
- Schritt 3: Falls alle Nachbarknoten von v in L_2 sind, so löschen Sie v aus L_1 . Ansonsten wählen Sie einen der Nachbarknoten w aus, der nicht in L_2 ist, schreiben Sie diesen ans Ende der Listen L_1 und L_2 und fügen Sie die Kante (v, w) zu E_0 hinzu.
- Schritt 4: Gehen Sie zu Schritt 2.

Übertragen wir den Algorithmus wieder zurück in das Schatzkartenbeispiel, so gilt: Die Kantenmenge E_0 speichert die Türen, die gesprengt werden; in der Liste L_2 merken wir uns alle Zimmer, die wir schon besucht haben, und die Liste L_1 enthält all diejenigen erreichten Zimmer, von denen aus noch nicht alle Türen überprüft bzw. gegebenenfalls auch gesprengt wurden.

Wenden wir den Algorithmus doch einmal auf den folgenden Graphen an.

Beispiel. Wir beginnen im Graphen aus der Abbildung 2.6 mit dem Knoten a . Dann gilt also $v_0 = a$, $L_1 = \{a\}$, $L_2 = \{a\}$ und E_0 enthält noch keine Kante. In Schritt 2 wählen wir jetzt den ersten Knoten aus der Liste L_1 , d.h. den Knoten a . Bisher ist noch keiner seiner Nachbarn in der Liste L_2 eingetragen. Wir fügen nun einen seiner Nachbarn, z.B. b , in die Liste L_1 und L_2 und die Kante (a, b) in E_0 ein. Damit gilt

$$L_1 = \{a, b\}, L_2 = \{a, b\} \text{ und } E_0 = \{(a, b)\}.$$

Da noch nicht alle Nachbarn von a in L_2 sind, betrachten wir nun den Knoten c und somit gilt nach Schritt 3

$$L_1 = \{a, b, c\}, L_2 = \{a, b, c\} \text{ und } E_0 = \{(a, b), (a, c)\}.$$

Jetzt sind alle Nachbarn von a in L_2 und wir löschen a aus L_1 , d.h. $L_1 = \{b, c\}$. Als Nächstes gehen wir zum Knoten b . Auch hier haben wir noch zwei Nachbarn, die nicht in L_2 enthalten sind, nämlich d und g . Nachdem wir diese beiden betrachtet haben, können wir auch b aus L_1 löschen und erhalten

$$L_1 = \{c, d, g\}, L_2 = \{a, b, c, d, g\} \text{ und } E_0 = \{(a, b), (a, c), (b, d), (b, g)\}$$

(Abb. 2.6). Wie sieht die Fortführung des Algorithmus aus?



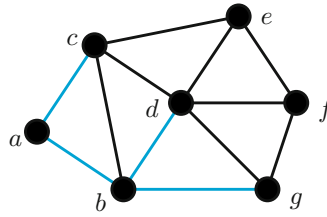


Abb. 2.6: Wir starten mit dem Knoten a und führen nach und nach die Schritte des Algorithmus aus.

Auf diesem Graphen hat der Algorithmus funktioniert und einen spannenden Baum berechnet. Bisher haben wir aber noch nicht bewiesen, dass der Algorithmus immer einen solchen Baum berechnet. Wir könnten auch einfach Glück gehabt haben. Damit wir uns nicht noch länger mit dieser Unsicherheit abfinden müssen, wollen wir seine Korrektheit beweisen.

Satz 2.2. *Wenn ein Graph zusammenhängend ist, so lässt sich ein spannender Baum mit der Breitensuche (Algorithmus 2.1) berechnen.*

Beweis: Den Beweis führen wir in drei Teilen: Zunächst zeigen wir, dass am Ende des Algorithmus alle Knoten in L_2 sind, d.h. $L_2 = V$. Danach beweisen wir, dass der berechnete Graph (L_2, E_0) zusammenhängend ist, und dann, dass er kreisfrei ist. Aus den beiden letzten Eigenschaften können wir sofort folgern, dass der Graph ein Baum ist (Satz 1.8).

Teil 1: Am Ende des Algorithmus enthält L_2 sämtliche Knoten.

Beweis: Beweis durch Widerspruch. Angenommen, am Ende des Algorithmus gibt es einen Knoten u_0 , der nicht in L_2 enthalten ist, d.h. $u_0 \notin L_2$.

Da G zusammenhängend ist, gibt es einen Weg vom Startknoten v_0 zu u_0 . Sei u_1 der erste Knoten auf diesem Weg, der nicht in L_2 ist, und v_1 dessen Vorgängerknoten auf dem Weg, also $u_1 \notin L_2$ und $v_1 \in L_2$. Da niemals Knoten aus L_2 gelöscht werden, war u_1 niemals in L_2 (Abb. 2.7).

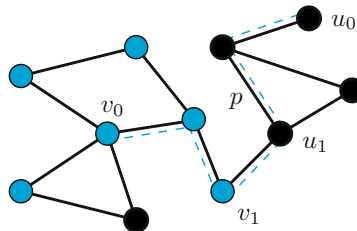


Abb. 2.7: Der Knoten u_0 ist mit v_0 über einen Weg verbunden. Die blauen Knoten sind in der Menge L_2 , die anderen nicht.

Nach der Anweisung in Schritt 2 werden Knoten nur gleichzeitig in L_1 und L_2 eingetragen. Da $v_1 \in L_2$ ist, war v_1 auch in L_1 . Der Algorithmus stoppt erst,

wenn L_1 leer ist, d.h. v_1 muss irgendwann gelöscht worden sein. Allerdings erfolgt dies erst, wenn alle Nachbarknoten in L_2 sind. Das ist ein Widerspruch dazu, dass v_1 den Nachbarknoten u_1 besitzt, der nach unserer Wahl noch nicht in L_2 enthalten ist. Die Annahme $u_0 \notin L_2$ ist also falsch. Somit enthält L_2 am Ende sämtliche Knoten. \square

Teil 2: Der berechnete Graph (V, E_0) ist zusammenhängend.

Beweis: Den Zusammenhang erhalten wir durch die Art und Weise, wie Knoten und auch Kanten in die Liste L_2 bzw. in die Kantenmenge E_0 in Schritt 3 eingefügt werden. Wir beweisen die Aussage mit vollständiger Induktion, indem wir zeigen, dass zu jedem Zeitpunkt im Algorithmus die Knoten aus L_2 und E_0 einen zusammenhängenden Graphen bilden. Die Induktion geht über die Anzahl der Knoten in L_2 . Zum Start des Algorithmus stimmt die Aussage, da der erste Knoten v_0 zusammenhängend ist. Damit haben wir den *Induktionsanfang*.

Die *Induktionsannahme* lautet: Die bisherigen k Knoten aus der Liste L_2 zusammen mit den bisherigen Kanten in E_0 bilden einen zusammenhängenden Graphen. Im *Induktionsschluss* müssen wir zeigen, dass der Knoten, der als nächstes in die Liste L_2 eingefügt wird, ebenfalls über einen Weg mit den anderen k Knoten aus der Liste verbunden ist. Betrachten wir also den Schritt des „Einfügens“ eines neuen Knotens w etwas genauer (Abb. 2.8). Der Knoten w wird in die Liste L_2 eingefügt, wenn ein Knoten v aus L_1 (und somit auch aus L_2) zu ihm benachbart ist. Zu diesem Zeitpunkt wird dann die Kante (v, w) in die Menge E_0 einbezogen und w sowohl in der Liste L_1 als auch in der Liste L_2 gespeichert. Da v mit jedem Knoten in der Liste L_2 über einen Weg verbunden ist (Induktionsannahme), ist auch w mit allen Knoten verbunden (wir gelangen über die Kante (w, v) vom Knoten w zum Knoten v und von dort aus zu jedem anderen Knoten). Damit ist L_2 und E_0 auch nach dem Hinzufügen des nächsten Knotens zusammenhängend. So haben wir auch diese Behauptung bewiesen. \square

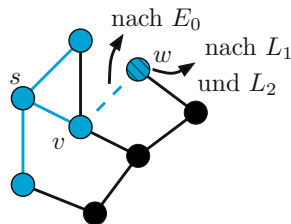


Abb. 2.8: Die blauen Knoten sind schon in L_1 und L_2 und die blauen Kanten in E_0 enthalten. Wird w über v erreicht, so wird die Kante (v, w) in E_0 und der Knoten w in die Listen L_1 und L_2 eingefügt.

Teil 3: Der Teilgraph (V, E_0) ist kreisfrei.

Beweis: Gäbe es einen Kreis, so müsste dieser irgendwann in Schritt 3 geschlossen werden. Die schließende Kante verbindet dann zwei Knoten, die schon in L_2 sind; in Schritt 3 werden aber nur Kanten zu E_0 hinzugefügt, bei denen ein Knoten zu diesem Zeitpunkt noch nicht in L_2 ist. \square

Aus den drei bewiesenen Behauptungen folgt sofort die Aussage des Satzes. \square

Eine einfache Variante des Algorithmus fügt die Knoten in Schritt 3 nicht ans Ende, sondern an den Anfang der Listen L_1 und L_2 . Dieser Algorithmus wird auch als *Tiefensuche* bezeichnet.

Algorithmus 2.2 Tiefensuche (engl. *depth first search*, DFS)

Input: Zusammenhängender Graph $G = (V, E)$.

Output: Spannender Baum $T = (V, E_0)$ von G .

Schritt 1: • Sei E_0 zunächst leer.
 • Bereiten Sie zwei ebenfalls leere Knotenlisten L_1 und L_2 vor.
 • Wählen Sie einen beliebigen Startknoten $v_0 \in V$ und schreiben Sie v_0 in L_1 und L_2 .

Schritt 2: Falls L_1 leer ist, dann STOPP und return $T = (V, E_0)$. Ansonsten nehmen Sie den ersten Knoten v aus L_1 .

Schritt 3: Falls alle Nachbarknoten von v in L_2 sind, so löschen Sie v aus L_1 . Ansonsten wählen Sie einen der Nachbarknoten w aus, der nicht in L_2 ist, schreiben Sie diesen **an den Anfang der Listen L_1 und L_2** und fügen Sie die Kante (v, w) zu E_0 hinzu.

Schritt 4: Gehen Sie zu Schritt 2.



Können Sie sich vorstellen, wie man auf die beiden Namen, Breiten- und Tiefensuche, gekommen ist? Probieren Sie die Algorithmen an dem folgenden Graphen aus (Abb. 2.9). Starten Sie dabei mit dem Knoten v_0 und vergleichen Sie anschließend die berechneten spannenden Bäume miteinander.

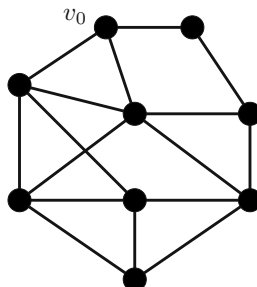


Abb. 2.9: Welche Bäume berechnen die beiden Algorithmen BFS und DFS?

2.3 Anwendungen von BFS und DFS

In Abschnitt 1.2 hatten wir schon gesagt, dass der Zusammenhang eines Graphen einen Schlüsselbegriff darstellt. Bisher hatten wir jedoch noch keine Methode kennen gelernt, um einen Graphen auf seinen Zusammenhang zu überprüfen bzw. die einzelnen Zusammenhangskomponenten zu bestimmen. Mit Hilfe der beiden Algorithmen BFS und DFS ist das leicht möglich. Wie könnte ein solcher Algorithmus aussehen? Versuchen Sie auch zu begründen, warum der von Ihnen vorgeschlagene Algorithmus die unterschiedlichen Zusammenhangskomponenten bestimmt?⁷



Aber nicht nur die einzelnen Zusammenhangskomponenten eines Graphen können durch spannende Bäume berechnet werden. Durch einen spannenden Baum, den wir aus der Breitensuche oder der Tiefensuche erhalten, kann auch überprüft werden, ob ein Graph bipartit ist oder nicht (Abschnitt 2.3.1). Außerdem haben die Bäume der Breitensuche eine ganz spezielle Eigenschaft: Wenn wir mit dem Knoten v_0 starten, so ist jeder Weg in dem Baum vom Knoten v_0 aus ein kürzester Weg bezüglich der Anzahl der Kanten (Abschnitt 2.3.2). Diese letzten beiden Anwendungen betrachten wir in den nächsten zwei Abschnitten genauer.

2.3.1 Erkennen von bipartiten Graphen

In Beispiel 1.1 hatten wir bipartite Graphen schon vorgestellt. Ein Graph $G = (V, E)$ heißt **bipartit**, wenn seine Knoten in zwei Teilmengen A und B zerlegt werden können, sodass nur Kanten zwischen den beiden Teilmengen und nicht innerhalb dieser verlaufen (Abb. 2.10).

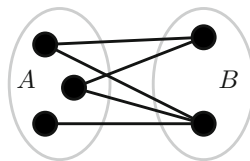


Abb. 2.10: Die Knotenmenge lässt sich in zwei Mengen zerlegen, wobei innerhalb der Mengen keine Kante verläuft.

Bipartite Graphen treten häufig bei der Darstellung von Zuordnungsproblemen auf. Ein typisches Zuordnungsproblem hat jeder Personaler zu bewältigen: Den freien Arbeitsplätzen in seinem Unternehmen muss er geeignete Kandidaten bzw. Bewerber zuordnen. Oder auch die Zuweisung von Räumen an der Universität zu den unterschiedlichen Veranstaltungen kann über einen bipartiten Graphen modelliert werden. Mit solchen Problemen werden wir uns später genauer beschäftigen (Kapitel 12.2). In diesem Abschnitt soll es darum gehen, wie man bei einem gegebenen Graphen feststellen kann, ob er bipartit ist oder nicht. Eine einfache Charakterisierung über die in einem bipartiten Graphen enthaltenen Kreise macht dies möglich.

Satz 2.3. *Ein Graph ist genau dann bipartit, wenn er keinen Kreis ungerader Länge enthält.*

Der Beweis ist konstruktiv, d.h. er zeigt nicht nur, dass die Aussage stimmt, sondern enthält auch die Methode, einen Kreis ungerader Länge zu finden, falls er in dem Graphen vorhanden ist. Diese Methode basiert auf der Berechnung eines spannenden Baums.



Beweis: Im Beweis gehen wir davon aus, dass der gegebene Graph G zusammenhängend ist. Wieso ist der Satz trotz dieser Einschränkung bewiesen?⁸

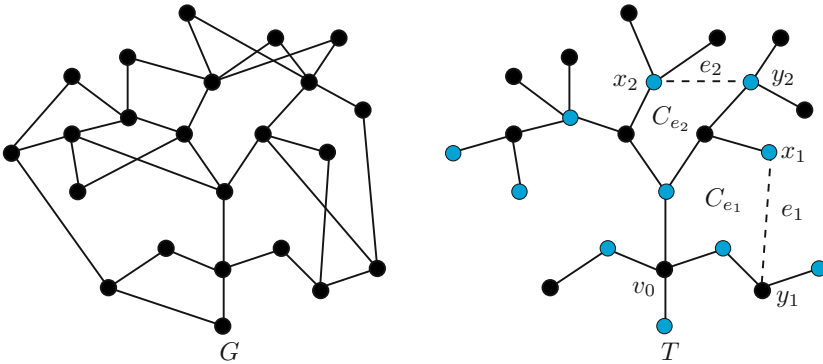


Abb. 2.11: Der Abstand zu r in T definiert eine bipartite Partition von G (blaue und schwarze Knoten). Jede Kante $e = (x, y) \in E(G) \setminus G(T)$ bildet einen Kreis C_e von gerader Länge. Also müssen sowohl x_1 und y_1 als auch x_2 und y_2 in unterschiedlichen Komponenten liegen.

„ \Leftarrow “: Wir zeigen als Erstes, dass ein zusammenhängender Graph G , der keinen Kreis ungerader Länge enthält, bipartit ist. Dazu teilen wir die Knoten V im Folgenden in zwei Mengen A und B auf, sodass zwischen den beiden Mengen keine Kante verläuft. Für die Aufteilung benötigen wir einen spannenden Baum. Sei $T \subseteq G$ ein spannender Baum und v_0 ein beliebiger Knoten aus G . Dann definieren wir die Menge A als die Menge aller Knoten, die mit v_0 durch einen Weg ungerader Länge verbunden sind, und mit B die Menge aller Knoten, die mit v_0 durch einen Weg gerader Länge verbunden sind (Abb. 2.11). Diese Definition ist sinnvoll, da in T ein eindeutig bestimmter Weg von v_0 zu jedem Knoten existiert. Wir müssen nun zeigen, dass diese Aufteilung bipartit ist, d.h. dass keine Kante von G zwei Knoten aus A oder zwei Knoten aus B verbindet.

Sei $e = (x, y)$ eine beliebige Kante von G . Wir betrachten zwei Fälle: *Fall 1:* Die Kante e ist eine Baumkante des spannenden Baums. Dann liegen x und y nach der Definition von A und B in unterschiedlichen Mengen. *Fall 2:* Die Kante e ist eine Nichtbaumkante. Nehmen wir an, sie verbindet zwei Knoten aus derselben Menge, wie z.B. die Kante e_2 aus der Abbildung 2.11. Wieso kann diese Kante e_2 nicht im ursprünglichen Graphen G (der nur Kreise gerader Länge enthält) existiert haben?⁹ Damit haben wir bewiesen, dass G bipartit ist.



„ \Rightarrow “: Der Graph G ist bipartit und wir wollen zeigen, dass er keinen Kreis ungerader Länge enthält. Betrachten wir zunächst einen beliebigen Weg, der in der Menge A startet und in A wieder endet. Der Weg muss eine gerade Länge haben. Nehmen wir an, es existiert ein Kreis ungerader Länge, der in dem Knoten x_1 startet und endet. Der Kreis entspricht aber einem Weg ungerader Länge. Also kann er nicht in derselben Menge starten und enden. Damit erhalten wir einen Widerspruch. \square

Wie schon gesagt, haben wir über den Beweis nicht nur eine Charakterisierung aller bipartiter Graphen, sondern auch die Idee für einen Algorithmus erhalten, mit dem wir einen bipartiten Graphen erkennen können. Überlegen Sie sich die Formulierung eines Algorithmus, um bipartite Graphen und ihre Knotenzerlegung zu finden. Probieren Sie den Algorithmus an unterschiedlichen Graphen aus und überprüfen Sie damit, ob der Algorithmus wirklich funktioniert.¹⁰



2.3.2 Mit wenigen Kanten zum Ziel

Betrachten wir den Baum, der durch den BFS-Algorithmus 2.1 berechnet wird, etwas genauer. Wir sehen, dass er stark in die Breite wächst. Außerdem fällt auf, dass für alle Knoten, die vom Startknoten v_0 über eine Kante mit v_0 verbunden sind, dies auch in dem berechneten Baum gilt. Kann ein Knoten erst über zwei Kanten von v_0 aus erreicht werden, so ist das in dem Baum ebenso.

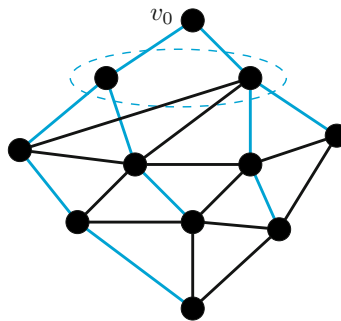


Abb. 2.12: Die eingekreisten Knoten können direkt von v_0 aus erreicht werden. Auch zu den anderen Knoten speichert der Baum einen kürzesten Weg bezüglich der Anzahl der Kanten.

Etwas verallgemeinert erhält man die folgende Aussage:

Satz 2.4. Sei G ein zusammenhängender Graph und T ein vom BFS-Algorithmus berechneter spannender Baum mit dem Startknoten v_0 . Dann ist jeder einfache Weg in T von v_0 aus ein kürzester Weg in G bezüglich der Anzahl der Kanten.

Beweis: Sei G der Graph, den wir betrachten, und T der spannende Baum, der vom BFS-Algorithmus berechnet wurde. Mit $\text{dist}(v)$ bezeichnen wir die Länge eines kürzesten Wegs von v_0 zu v und mit $\text{level}(v)$ die Länge des einfachen Wegs in T von v_0 zu v . Da jeder Weg in T auch ein Weg in G ist, gilt immer $\text{dist}(v) \leq \text{level}(v)$. Wir wollen nun zeigen, dass $\text{level}(v) = \text{dist}(v)$ für alle Knoten $v \in V$ gilt. Angenommen, das ist nicht der Fall. Dann gibt es mindestens einen Knoten w mit $\text{level}(w) > \text{dist}(w)$. Wir wählen nun einen solchen Knoten w_0 mit minimalem Wert $\text{dist}(w_0)$. Man bezeichnet w_0 auch als den „kleinsten Verbrecher“. Sei nun p ein kürzester Weg in G von v_0 zu w_0 und sei (u_0, w_0) die letzte Kante auf p (Abb. 2.13).

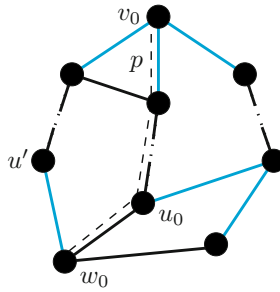


Abb. 2.13: Der Weg p ist ein kürzester Weg von v_0 nach w_0 in G .

Für den Knoten u_0 gilt $\text{dist}(u_0) = \text{level}(u_0)$, weil w_0 der kleinste Verbrecher ist, und es gilt $\text{dist}(w_0) = \text{dist}(u_0) + 1$. Außerdem erhalten wir aus der Annahme $\text{dist}(w_0) < \text{level}(w_0)$ und damit die Abschätzung

$$\text{level}(w_0) > \text{dist}(w_0) = \text{dist}(u_0) + 1 = \text{level}(u_0) + 1,$$

also

$$\text{level}(w_0) > \text{level}(u_0) + 1. \quad (2.1)$$

Daraus folgt, dass der Knoten u_0 vor dem Knoten w_0 in die Liste L_1 gekommen ist. Dann wird der Knoten w_0 aber entweder von u_0 aus betrachtet, d.h. es gilt $\text{level}(w_0) = \text{level}(u_0) + 1$, oder von einem anderen Knoten u' , der vor u_0 in L_1 enthalten ist, d.h. $\text{level}(w_0) = \text{level}(u') + 1 \leq \text{level}(u_0) + 1$. Das ist aber ein Widerspruch zur Ungleichung 2.1. \square

Schauen wir uns ein Beispiel an, in dem genau nach dieser Eigenschaft eines spannenden Baums gefragt ist.

Beispiel. Maren ist gerade in Berlin angekommen und möchte nun von der Peterstraße zum Eulerhof mit der S-Bahn fahren (Abb. 2.14). Für die Strecke zwischen zwei Stationen benötigt der Zug immer drei Minuten; außerdem gehen wir davon aus, dass das Umsteigen zwischen zwei unterschiedlichen Linien keine Zeit in Anspruch nimmt. Wie kommt Maren möglichst schnell an ihr Ziel? \blacksquare



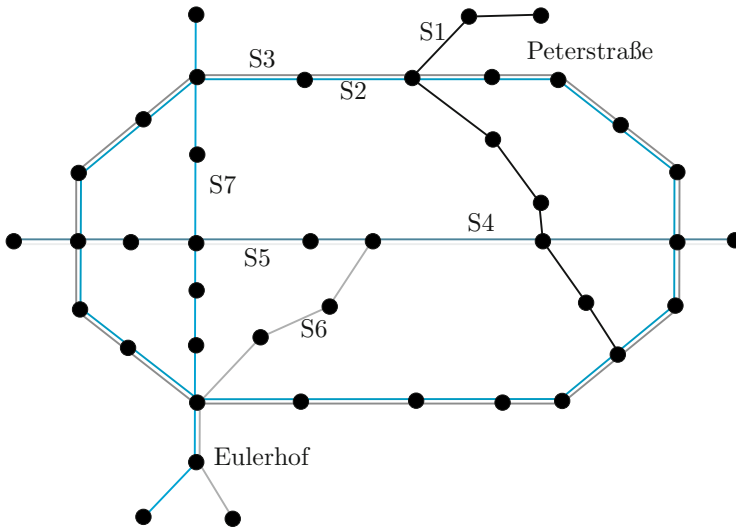


Abb. 2.14: Wie kommt Maren am schnellsten an ihr Ziel?

In Kapitel 9 werden wir uns noch intensiver mit der Suche nach kürzesten Wegen beschäftigen. Zunächst verweilen wir jedoch noch bei spannenden Bäumen.

2.4 Aufgaben

Aufgabe 2.1. Drei Gefangene wollen aus ihrem Gefängnis ausbrechen. Ein bestochener Wärter hat ihnen den Raumplan des Gefängnisses in die Zelle geschmuggelt (Abb. 2.15). Welchen Weg sollten die Gefangenen benutzen, um ins Freie zu kommen? Zum Öffnen einer Tür benötigen sie 15 Minuten.

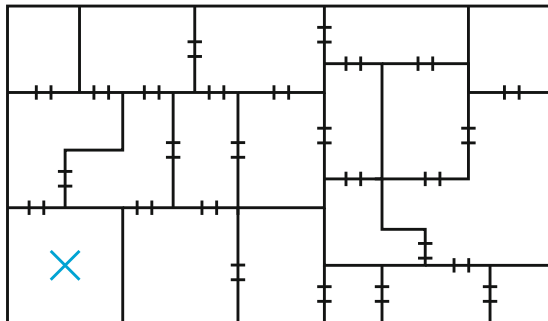


Abb. 2.15: Die Gefangenen befinden sich im Raum mit dem Kreuz des Hochsicherheitstraktes.

Aufgabe 2.2. Bisher haben wir Graphen meist als Zeichnung mit Knoten und Kanten dargestellt. Häufig werden Graphen jedoch als Adjazenzmatrizen gespeichert: Jede Zeile und jede Spalte ist einem Knoten zugeordnet und in der Matrix

steht eine Eins, wenn die beiden Knoten benachbart sind. Ansonsten steht dort eine Null (Abb. 2.16).

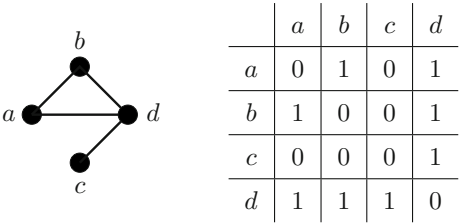


Abb. 2.16: Wenn zwischen zwei Knoten eine Kante existiert, dann steht in der Adjazenzmatrix eine Eins.

- 1. Wie kann man bei einer solchen Matrix den Grad eines Knotens bestimmen?
- 2. Müssen die Breitensuche und die Tiefensuche verändert werden, damit er auch auf einen Graphen angewendet werden kann, der nur als Adjazenzmatrix dargestellt ist? Wenden Sie den Algorithmus auf die Matrix in der Abbildung 13.4 an. Das Ergebnis lässt sich gut überprüfen, wenn man zuerst den Graphen zeichnet und anschließend den berechneten Baum einträgt.

| | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| a | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| b | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| c | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| d | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| e | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| f | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| g | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| h | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

Abb. 2.17: Wie sieht der zu dieser Matrix gehörende Graph aus?

Aufgabe 2.3. Ein Freund erzählt von seiner Strategie, aus einem Labyrinth herauszufinden.

Rechte-Hand-Regel

Starten Sie in einem Zimmer bzw. am Eingang und wählen Sie eine Wand aus. Berühren Sie diese mit der rechten Hand. Laufen Sie nun entlang dieser Wand und bleiben Sie dabei immer mit der rechten Hand in Kontakt der Wand bis das Ziel erreicht ist.

Wenden Sie seine Regel auf das Labyrinth in der Abbildung 2.18 an. Wann funktioniert die Regel und wann nicht? Welche Regel bzw. Strategie wäre sinnvoller und was hat das Problem mit spannenden Bäumen zu tun?

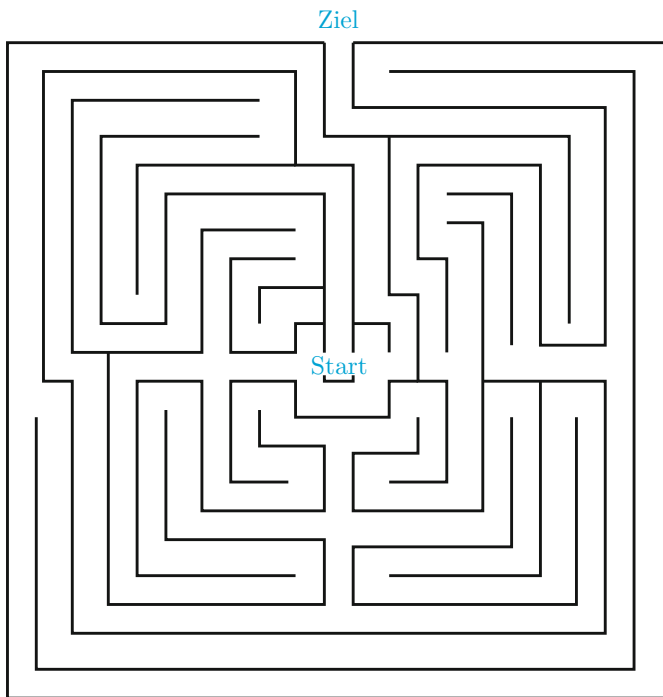


Abb. 2.18: Wie kommt man aus dem Irrgarten von Chevening in Kent?

Aufgabe 2.4. Welche Bäume wurde von s aus mit der Breitensuche, welche mit der Tiefensuche und welche mit keiner der beiden Algorithmen berechnet?

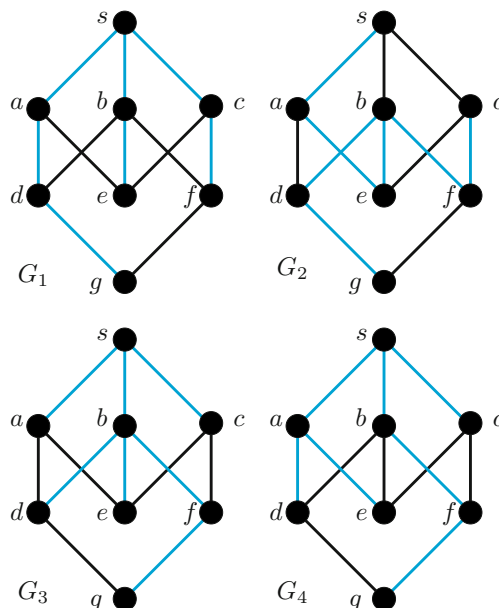


Abb. 2.19: Wie wurden die einzelnen Bäume konstruiert?

Aufgabe 2.5. Gibt es Fälle, in denen der vom DFS-Algorithmus berechnete Baum von v_0 aus zu jedem Knoten einen kürzesten Weg enthält? Berechnet der DFS-Algorithmus immer einen längsten Weg von v_0 zu den anderen Knoten?

Aufgabe 2.6. Sei $G = (V, E)$ ein zusammenhängender einfacher Graph. Beweisen Sie oder widerlegen Sie durch ein Beispiel:

1. Sei s der Knoten, in dem der DFS-Algorithmus startet, und $T = (V, E_0)$ der konstruierte Baum. Dann ist jede Kante $(s, v) \in E$ auch in E_0 .
2. Sei s der Knoten, in dem der BFS-Algorithmus startet. Dann ist jede Kante $(s, v) \in E$ auch in E_0 .

Aufgabe 2.7. Seien G ein Graph, T_{BFS} der von der Breitensuche berechnete spannende Baum, und T_{DFS} der von der Tiefensuche berechnete Baum. Mit $d_G(v)$ bezeichnen wir den Grad eines Knotens v im Graphen G , mit $d_{\text{BFS}}(v)$ den im T_{BFS} und mit $d_{\text{DFS}}(v)$ den in T_{DFS} . Welche Aussagen sind richtig und welche sind falsch?

1. Für alle Knoten $v \in V$ gilt $d_G(v) = d_{\text{BFS}}(v)$.
2. Es gibt immer einen Knoten mit $d_G(v) = d_{\text{BFS}}(v)$.
3. Es gibt immer einen Knoten mit $d_G(v) = d_{\text{DFS}}(v)$.

Aufgabe 2.8. Sei T ein spannender Baum von $G = (V, E)$ und $s \in V$. Betrachtet man $T = (V, E_T)$ als einen gerichteten Graphen, in dem alle Kanten von s weg orientiert sind, so heißen zwei Knoten u, v auch T -verbunden, wenn in T ein (u, v) -Weg oder ein (v, u) -Weg enthalten ist (Abb. 2.20).

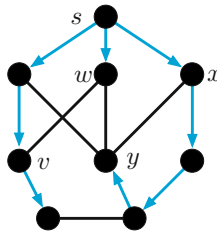


Abb. 2.20: Die Knoten a und b sind T -verbunden, die Knoten c und d nicht.

Sei T_{DFS} ein vom Knoten s aus durch die Tiefensuche berechneter Baum und sei (x, y) eine Kante, die nicht in T_{DFS} enthalten ist. Dann sind die Knoten x, y ebenfalls T_{DFS} -verbunden.

Aufgabe 2.9. Zeigen Sie, dass die Aussage 2.8 allgemein nicht für einen Breitensuchbaum gilt.

Lösungen zu den Fragen im Text

- 6 $|E| - n + 1$ Kanten müssen aus G gelöscht werden, damit aus G ein Baum entsteht. Da die Anzahl der Kanten von G maximal $\frac{n(n-1)}{2}$ beträgt, sind wir spätestens nach $\frac{n(n-1)}{2} - n + 1$ Iterationen fertig.
- 7 Die Idee des Algorithmus ist die folgende: Sei G ein Graph und $Z_1 = (V_1, E_1), \dots, Z_k = (V_k, E_k)$ seine Zusammenhangskomponenten, d.h. Teilgraphen von G . Wenden wir die Breitensuche mit dem Startknoten v_i auf die Zusammenhangskomponente Z_i an, so erhalten wir einen spannenden Baum $T = (V', E')$ der Zusammenhangskomponente. Insbesondere gilt $V' = V_i$. Allerdings kennen wir die Zusammenhangskomponenten ja noch nicht, sondern wollen sie erst berechnen. Wenn wir den Algorithmus aber auf den kompletten Graphen G anwenden und ebenfalls mit v_i starten, wird wiederum derselbe Baum $T = (V', E')$ berechnet. Von diesem Baum wissen wir $V' = V_i$ und somit haben wir die Knoten der i -ten Zusammenhangskomponenten bestimmt. Löschen wir diese Knoten aus dem Graphen G , können wir wieder mit einem beliebigen Knoten starten und die Prozedur von vorne beginnen. Nach und nach erhalten wir so alle Zusammenhangskomponenten.

Algorithmus 2.3 Berechnung der Zusammenhangskomponenten.

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

Output: Knoten der Zusammenhangskomponenten V_1, \dots, V_k .

Schritt 1: Setzen Sie $i = 1$.

Schritt 2: Wählen Sie einen beliebigen Knoten $v_i \in V$.

Schritt 3: Berechnen Sie einen Baum $T = (V', E)$ in $G = (V, E)$ von v_i , wie in Algorithmus 2.2 vorgegeben.

Schritt 4: Setzen Sie $V_i = V'$ und löschen Sie V_i aus G , d.h. $G = G \setminus V_i$. Falls G noch einen Knoten enthält, erhöhen Sie i um eins und gehen Sie zurück zu Schritt 2. Ansonsten Return V_1, \dots, V_i .

- 8 Sei G ein Graph, der aus mehreren Zusammenhangskomponenten besteht. Falls G bipartit ist, dann ist auch jede Zusammenhangskomponente bipartit. Wenn wir zeigen, dass ein zusammenhängender bipartiter Graph keinen Kreis ungerader Länge enthält, dann enthält auch keine Zusammenhangskomponente einen Kreis ungerader Länge. Damit ist ein solcher Kreis auch in dem ganzen Graphen G nicht vorhanden. Auf der anderen Seite: Wenn in G kein Kreis ungerader Länge existiert, dann auch in keiner Zusammenhangskomponente. Wenn jeder zusammenhängende Graph ohne einen Kreis ungerader Länge bipartit ist, dann ist jede Zusammenhangskomponente bipartit. Dann ist aber auch ganz G bipartit.
- 9 Angenommen, es existiert eine Kante $e_2 = (x_1, x_2)$, die zwei Knoten der gleichen Komponente miteinander verbindet. Wir bezeichnen mit p_1 den Weg von v_0 zu x_1 und mit p_2 den Weg von x_2 zu v_0 . O.B.d.A. bestehen beide Wege aus einer geraden Anzahl an Kanten. Dann ist $p_1 \cup e_1 \cup p_2$ ein Kreis in G mit ungerader Länge. Damit haben wir einen Widerspruch zur Voraussetzung.
- 10 Der folgende Algorithmus konstruiert, falls möglich, eine Partition der Knoten.

Algorithmus 2.4 Erkennung bipartiter Graphen

Input: Graph G .

Output: Zwei Knotenmengen A, B oder die Aussage, dass G nicht bipartit ist.

Schritt 1: Wählen Sie einen beliebigen Knoten $v_0 \in G$ und bestimmen Sie einen spannenden Baum T in G .

Schritt 2: Färben Sie alle Knoten mit ungeradem Abstand in T zu r blau, die anderen schwarz.

Schritt 3: Überprüfen Sie alle Kanten $e \in E(G) \setminus E(T)$, die nicht in T enthalten sind, ob sie zwischen zwei blauen oder zwei schwarzen Knoten verlaufen.

- a) Verläuft eine Kante zwischen zwei gleich gefärbten Knoten, return: Der Graph ist nicht bipartit.
 - b) Andernfalls return: Der Graph ist bipartit mit $V = V_{\text{blau}} \dot{\cup} V_{\text{schwarz}}$.
-

3 Das Minimal-Spannende-Baum-Problem

Übersicht

| | | |
|-----|--|----|
| 3.1 | Das Problem und zwei Algorithmen | 39 |
| 3.2 | Zwei Optimalitätskriterien | 43 |
| 3.3 | Aufgaben | 51 |

3.1 Das Problem und zwei Algorithmen

Der Wüstenstaat des Großvisirs Abu al Mi besteht aus sieben wunderschönen Oasen, die untereinander durch Handelsstraßen verbunden sind. Allerdings sind diese durch die ständigen Winde und starke Sonne immer brüchiger geworden und müssen dringend renoviert werden. Nach gründlicher Überprüfung der Lage ist der Großvisir zu dem Entschluss gekommen, nicht alle Handelsrouten wieder herstellen zu lassen. Es reicht ihm, wenn es möglich ist, von jeder Oase zu jeder anderen zu gelangen.

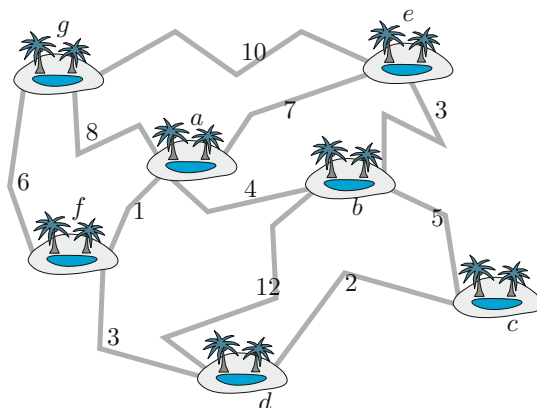


Abb. 3.1: Welche Straßen sollen tatsächlich renoviert werden?

Seine Minister sollen sich jetzt überlegen, wie dieses Vorhaben am kostengünstigsten durchgeführt werden kann. Natürlich wollen die Minister in dieser heiklen Angelegenheit nicht selber Hand an das Problem legen. Deswegen wurde eine Gruppe von Bauingenieuren zusammengerufen, die Kosten für die Renovierungsarbeiten zu schätzen und anschließend einen Vorschlag zu präsentieren. Schon einige Zeit sitzen sie nun über ihren Daten und grübeln, welche Routen überholt werden sollen (Abb. 3.1). Kann ihnen die Graphentheorie bei dieser Frage weiterhelfen?

Dazu müssen wir uns als Erstes überlegen, wie wir das Problem graphentheoretisch auffassen sollen. Eine Möglichkeit ist, die Oasen als Knoten und die vorhandenen Straßen als Kanten zu modellieren. Jeder Kante wird nun ein Gewicht zugeordnet, das angibt, wie teuer es wäre, sie zu renovieren. In diesem Graphen wird dann ein Teilgraph (das zu realisierende Bauvorhaben) gesucht, der zusammenhängend ist (es soll ja immer noch möglich sein, von jeder Oase zu jeder anderen zu kommen) und möglichst wenig kostet. Die Kosten für einen Teilgraphen entsprechen der Summe der in diesem Graphen enthaltenen Kanten (Abb. 3.2).

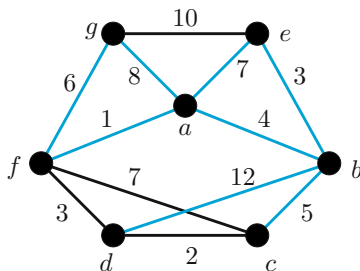


Abb. 3.2: Die Kosten für den blauen Teilgraphen belaufen sich auf $6 + 8 + 1 + 7 + 4 + 12 + 3 + 5 = 46$. Kann man auch eine Kante weglassen, so dass zwischen zwei Knoten weiterhin ein blauer Weg besteht? Welche Kosten ergeben sich für den neuen Graphen?

Es stellt sich die Frage, ob uns wirklich jeder zusammenhängende Teilgraph interessiert, oder ob wir uns auf eine bestimmte Klasse von Teilgraphen beschränken können. Kann es sein, dass der billigste Teilgraph einen Kreis enthält? Eigentlich nicht, weil wir einfach eine Kante aus dem Kreis löschen können und dann immer noch einen zusammenhängenden Teilgraphen haben. Dieser neue Teilgraph kostet dann auch weniger als der alte, weil eine Kante weniger bezahlt werden muss. Wir können unsere Betrachtungen also auf Bäume reduzieren.

Definition (Minimal spannender Baum). Sei $G = (V, E)$ ein Graph mit Kantengewichten $c(e)$ für jede Kante $e \in E$. Ein spannender Baum mit minimalem Gewicht wird auch als **minimal spannender Baum** bezeichnet (Abb. 3.3). Das **Gewicht** $c(T)$ eines spannenden Baums T ergibt sich durch die Summe seiner Kantengewichte, d.h.

$$c(T) = \sum_{e \in E(T)} c(e).$$

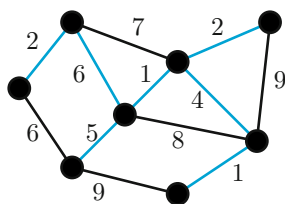


Abb. 3.3: Der blaue Graph ist ein minimal spannender Baum des gesamten Graphen.

Unser Oasenproblem lässt sich somit darauf zurückführen, einen minimal spannenden Baum in einem Graphen mit Kantengewichten zu suchen. Dies wird auch als **Minimal-Spannendes-Baum-Problem** (engl. *Minimum Spanning Tree Problem*, MST) bezeichnet.

Das MST-Problem hört sich erst mal gar nicht so schwierig an: Man könnte ja einfach alle spannenden Bäume mit ihren Gewichten in dem Graphen berechnen und dann den billigsten aussuchen. Leider gibt es in einem vollständigen Graphen insgesamt n^{n-2} solcher Bäume, wie Cayley bewiesen hat.

Satz 3.1 (Caley). *In einem vollständigen Graphen K_n gibt es n^{n-2} unterschiedliche spannende Bäume.*

Angenommen, man könnte pro Sekunde 10^6 Bäume berechnen, dann würden zur Berechnung von $n = 30$ immer noch $30^{28}/10^6 \text{ sec} = 7,25 \cdot 10^{27}$ Jahre ins Land gehen. Und auch schon für die sieben Oasen müssten wir stattliche $7^5 = 17.807$ Bäume durcharbeiten. Das ist etwas zu viel.

Bevor wir jedoch verzweifeln, hören wir uns erst mal an, was der jüngste Bauingenieur der Gruppe, Jack, den anderen vorschlägt: „Warum renovieren wir nicht zuerst die billigste Straße, dann die zweitbilligste usw. Wenn die Renovierung einer Straße bedeutet, dass wir einen Kreis schließen, dann bauen wir sie nicht. Stattdessen gehen wir zur nächstteureren Straße.“ Die anderen sind etwas skeptisch. Wie sieht Jacks Bauplan aus und wie viel kostet er?

Jack ist nicht der Erste, der auf diese Idee gekommen ist. Der Algorithmus, den er vorschlägt, ist auch als Algorithmus von Kruskal bekannt (Alg. 3.1).



Algorithmus 3.1 Der Algorithmus von Kruskal

Input: Graph $G = (V, E)$ mit Kantengewichten $c(e)$ für alle $e \in E$.

Output: Minimal spannender Baum T von G .

Schritt 1: Sortieren Sie die Kanten aufsteigend nach Gewichten, d.h. es soll $c(e_1) \leq c(e_2) \leq \dots \leq c(e_m)$ gelten.

Schritt 2: Gehen Sie die Kanten der Reihenfolge nach durch und fügen Sie sie in den Baum T ein, falls sie keinen Kreis schließen.

Schauen wir uns kurz an einem Beispiel an, wie der Algorithmus arbeitet.



Beispiel. In der Abbildung 3.4 wählt der Algorithmus zuerst die Kanten mit den Gewichten 1, 2 und 3. Die Kante mit dem Gewicht 4 wird nicht dem Baum hinzugefügt. Welche Kante wählt er als Nächstes und welche lässt er weg? Wieso können wir den Algorithmus abbrechen, sobald $n - 1$ Kanten in dem Graphen T enthalten sind?¹¹ ■

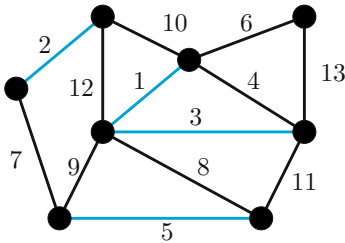


Abb. 3.4: Nach vier Schritten hat der Algorithmus die blauen Kanten ausgewählt. Welche Kante kommt als Nächstes?

Nachdem Jack den anderen seinen Vorschlag unterbreitet hat, meldet sich Maren zu Wort. Ihre Vorgehensweise unterscheidet sich etwas von Jacks Ansatz. Ihr Vorschlag ist, von dem Sitz des Großvisirs, der Oase a , aus mit der günstigsten Straße anzufangen und dann nach und nach den spannenden Baum mit der jeweils günstigsten Straße wachsen zu lassen. Sie würde also zuerst die Route von der Oase a zur Oase f , dann die von der Oase f zur Oase d , dann wieder eine Route von der Oase d zur Oase e usw. ausbauen lassen (Abb. 3.5). Welche Straßen werden nach dieser Vorgehensweise gebaut und welches Budget müsste der Großvisir hierfür bereitstellen?

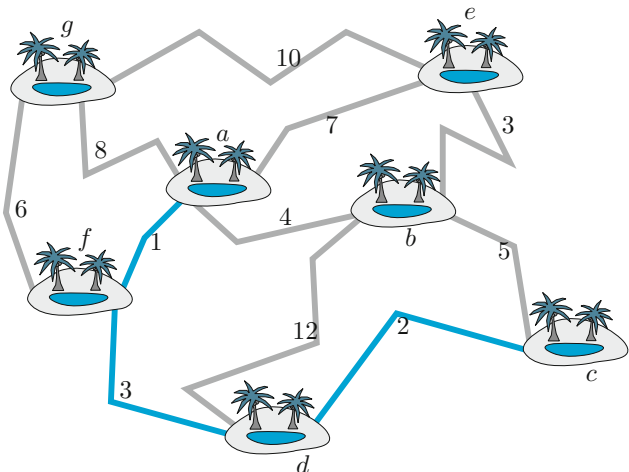


Abb. 3.5: Die blauen Straßen würde Maren als Erste renovieren.

Auch Marens Idee ist nicht neu. Der Mathematiker Prim hat sie schon 1957 als Algorithmus veröffentlicht (Alg. 3.2).

Algorithmus 3.2 Algorithmus von Prim

Input: Graph $G = (V, E)$ mit Kantengewichten $c(e)$ für alle $e \in E$.

Output: Minimal spannender Baum T von G .

Schritt 1: Wählen Sie einen Knoten $v_0 \in V$.

Schritt 2: Solange T noch nicht alle Knoten aus G enthält, wiederholen Sie die folgende Prozedur:

Schritt (a): Wählen Sie die billigste Kante aus, die von einem schon besuchten Knoten zu einem noch nicht besuchten Knoten geht.

Schritt (b): Fügen Sie die Kante und den nun erreichbaren Knoten in den Baum T ein.

In der Abbildung 3.6 sind wir beim Knoten v_0 gestartet und haben als Erstes die günstigste Kante, die von dem Knoten ausgeht, gewählt. Die neuen möglichen Kanten liegen auf der gestrichelten Linie. Welche Kante würde nach dem Algorithmus von Prim als Nächstes dem Baum hinzugefügt werden? Wie sieht der fertige Baum aus?

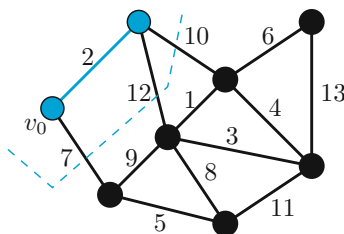


Abb. 3.6: Bisher sind die beiden blauen Knoten in T enthalten.

Jetzt haben wir zwei Algorithmen kennen gelernt, die beide behaupten, einen minimal spannenden Baum zu berechnen. Ob sie das wirklich tun, werden wir im nächsten Abschnitt sehen.

3.2 Zwei Optimalitätskriterien

Wir gehen das Problem noch einmal von einer anderen Seite an: Angenommen, uns ist ein spannender Baum gegeben. Wie können wir überprüfen, ob er optimal ist?

Die Kosten einer optimalen Lösung kennen wir ja nicht. Betrachten wir folgendes Beispiel.

Beispiel. Der Baum T (die blauen Kanten) ist ein spannender Baum in G mit einem Gewicht von 19 (Abb. 3.7). Die Kante e_1 ist nicht in dem spannenden Baum (sie wird auch als **Nicht-Baumkante** bezeichnet), hat aber nur ein Gewicht von 1. Wenn wir jetzt die Kante e_1 in den Baum einbeziehen, schließt die Kante einen Kreis. Damit wir wieder einen Baum erhalten, können wir eine beliebige Kante aus dem Kreis löschen. Am besten nehmen wir dazu die teuerste Kante e_2 mit den Kosten 6. Der neue Baum ist somit günstiger als der alte. Kann man diesen Austauschtrick noch auf eine andere Nicht-Baumkante anwenden? ■

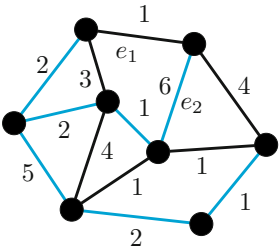


Abb. 3.7: Die Kante e_1 schließt einen Kreis im Baum T .

Damit haben wir schon ein erstes Kriterium kennen gelernt: In einem minimal spannenden Baum ist jede Nicht-Baumkante teurer als die Kanten auf dem entstehenden Kreis. Ein solcher Kreis wird auch als **Fundamentalkreis** bezeichnet.

Definition. Sei $T = (V, E(T))$ ein spannender Baum in $G = (V, E)$ und $e \in E \setminus E(T)$ eine **Nicht-Baumkante**. Der von e erzeugte **Fundamentalkreis** C_e bezüglich T ist der einfache eindeutige Kreis, der in $T + e$ entsteht (Abb. 3.8).

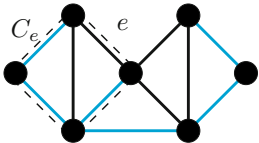


Abb. 3.8: Die gestrichelte Linie deutet den von e erzeugten Fundamentalkreis bezüglich des blauen Baums T an.

Zur Erinnerung: Der Graph $T + e$ ist definiert als der Graph T zuzüglich der Kante e . Wie sieht der von e_1 bzw. e_2 und e_3 induzierte Fundamentalkreis in dem Graphen der Abbildung 3.9 aus?



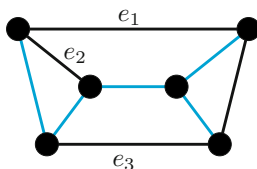


Abb. 3.9: Die blauen Kanten bilden einen spannenden Baum in dem Graphen.

Zurück zur Definition: Muss der Graph $T + e$ überhaupt einen Kreis enthalten? Wieso ist der Kreis in $T + e$ eindeutig? Könnte es nicht sein, dass doch zwei unterschiedliche Kreise mit der Kante e und den Baumkanten existieren? Da nicht sofort klar ist, ob die Definition sinnvoll ist, (wir sprechen in der Mathematik von **wohldefiniert**) überprüfen wir dies im nächsten Lemma.

Lemma 3.2. *Der von der Nicht-Baumkante e erzeugte Fundamentalkreis bezüglich des spannenden Baums T im Graphen G ist wohldefiniert.*

Beweis: Sei $e = (u, v)$ eine Nicht-Baumkante. Da T ein Baum ist, existiert in T ein einfacher Weg von u nach v . Dieser enthält nicht die Kante e , da diese nicht im Baum liegt. Der Weg einschließlich der Kante e bildet einen Kreis. Wir haben damit in $T + e$ einen Kreis gefunden, der e enthält (Abb. 3.10).

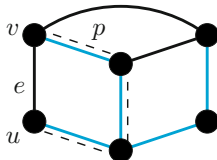


Abb. 3.10: Der Weg p verbindet die beiden Knoten u und v in T .

Jetzt müssen wir noch zeigen, dass nicht zwei unterschiedliche einfache Kreise in $T + e$ entstanden sind. Angenommen es gäbe zwei unterschiedliche Kreise C_1 und C_2 in $T + e$. Diese Kreise müssten die Kante e enthalten, da T ein Baum und somit kreisfrei ist. Löschen wir die Kante e aus beiden Kreisen, so bleiben zwei unterschiedliche einfache Wege von u nach v , die komplett im Baum T verlaufen. Wieso ist das ein Widerspruch dazu, dass T ein Baum ist?¹²

Wir haben also bewiesen, dass zu jeder Nicht-Baumkante ein eindeutiger Fundamentalkreis gehört. □



Ein weiteres Kriterium für einen minimal spannenden Baum erhalten wir über Schnitte. Auch hierzu schauen wir uns ein Beispiel an.

Beispiel. Der Baum T ist, wie schon im letzten Beispiel, ein spannender Baum von G mit dem Gewicht von 19 (Abb. 3.11).

Die Kante e_1 ist in dem Baum enthalten, ist aber relativ teuer mit Kosten von 5. Löschen wir diese Kante, so zerfällt der Baum in zwei Teile. Zwischen den beiden Teilen verlaufen mehrere Kanten (insbesondere die Kante e_1). Fügen wir eine beliebige dieser Kanten wieder in den Graphen, so entsteht wieder ein Baum. Da wir uns irgendeine aussuchen können, sollten wir die günstigste wählen, hier die Kante e_2 . ■

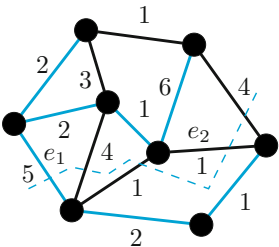


Abb. 3.11: Die Kante e_1 mit dem Gewicht 5 kann durch die Kante e_2 mit dem Gewicht 1 ersetzt werden, um einen billigeren spannenden Baum zu erreichen.

Statt eines Fundamentalkreises haben wir die Kanten betrachtet, die zwischen zwei Teilen eines Baums verlaufen. Diese Kanten bilden somit auch einen Schnitt, was in der folgenden Definition noch deutlicher wird.

Definition. Sei $T = (V, E(T))$ ein spannender Baum von $G = (V, E)$ und $e = (u, v) \in E(T)$ eine **Baumkante**. Sei X_u die Menge der Knoten, die in $T - e$ von u aus erreichbar ist. Der von e erzeugte **Fundamentalschnitt** bezüglich T ist der Schnitt $\delta(X_u)$ in G (Abb. 3.12).

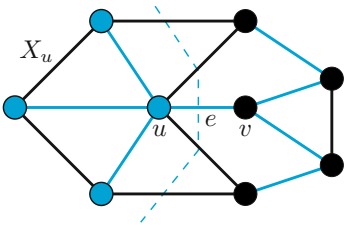


Abb. 3.12: Die blauen Knoten markieren die Menge X_u , wenn die Kante $e = (u, v)$ gelöscht wird.

Die Definition ist etwas komplizierter. Wie viele Kanten liegen in dem Fundamentalschnitt von e' in dem Graphen G' (Abb. 3.13 (a)) und wie viele Knoten liegen in X_u ? Im rechten Graphen G^* (Abb. 3.13 (b)) ist nur eine Kante e^* schon blau eingefärbt und dafür die Kanten, die zu ihrem Fundamentalschnitt gehören sollen, in grau gegeben. Wie sieht ein dazugehöriger spannender Baum aus?



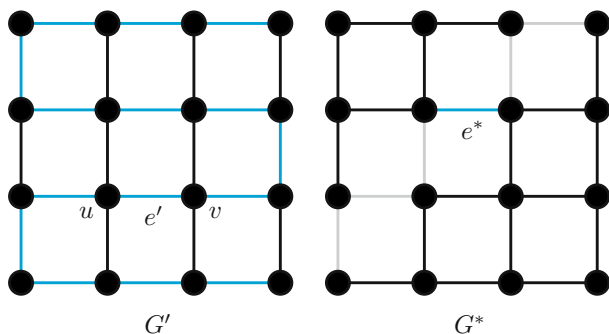


Abb. 3.13: In G' ist schon ein spannender Baum eingezeichnet. In G^* hingegen ist nur ein Fundamentalschnitt zur Baumkante e^* durch die blauen Kanten bekannt. Gibt es mehr als eine Möglichkeit, einen spannenden Baum in G^* einzuzichnen, der diesen Fundamentalschnitt enthält?

Auch hier stellt sich die Frage, ob die Definition sinnvoll ist. Ist die Definition des von der Kante $e = (u, v)$ erzeugten Fundamentalschnitts abhängig davon, ob wir die Knotenmenge X_u oder die Knotenmenge X_v betrachten? Außerdem muss überprüft werden, ob die Menge X_u eine echte Teilmenge von V ist.

Lemma 3.3. *Der von der Baumkante $e = (u, v)$ induzierte Fundamentalschnitt $\delta(X_u)$ bezüglich des spannenden Baums T im Graphen G ist wohldefiniert. Außerdem ist e die einzige Baumkante in $\delta(X_u)$.*

Beweis: Im Baum T existiert ein eindeutiger Weg zwischen je zwei beliebigen Knoten. Das gilt auch für die Knoten u und v , die über die Kante e direkt miteinander verbunden sind (Abb. 3.14). Wird diese Kante aus T gelöscht, so existiert kein Weg mehr zwischen u und v . Damit gilt $v \notin X_u$ und $u \in X_u$ und daraus folgt $\emptyset \neq X_u \subsetneq V$. Also bildet X_u einen zulässigen Schnitt (Definition auf Seite 1.2).

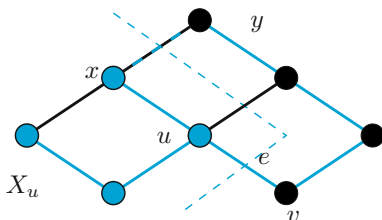


Abb. 3.14: Die Menge X_u kann den Knoten v nicht enthalten. Auch eine weitere Baumkante kann nicht in dem Schnitt $\delta(X_u)$ liegen.

Als Nächstes zeigen wir noch, dass e die einzige Baumkante von T in $\delta(X_u)$ ist. Angenommen, die Kante (x, y) ist eine Baumkante und liegt ebenfalls in $\delta(X_u)$. Dann befindet sich ein Knoten — nehmen wir an x — in der Menge X_u und der andere Knoten — hier y — in der Menge $V \setminus X_u$. In der Menge X_u sind aber alle Knoten enthalten, die von u aus in $T - e$ erreichbar sind. Da x von u aus

erreicht wird und der Knoten y über die Baumkante (x, y) mit x verbunden ist, wird auch y von u aus in $T - e$ erreicht. Damit ist auch y in X_u . Daraus folgt der Widerspruch, dass der Schnitt $\delta(X_u)$ die Kante (x, y) enthält. \square

Für spannende Bäume haben wir jetzt zwei Kriterien an der Hand, mit denen wir die Optimalität eines Baums überprüfen können.

Satz 3.4 (Optimalitätskriterium für MST). *Sei T ein spannender Baum des Graphen $G = (V, E)$ und seien $c(e)$ Kosten auf den Kanten $e \in E$. Dann sind die folgenden Aussagen äquivalent:*

1. *Der Baum T ist ein minimal spannender Baum.*
2. *Für jede Nicht-Baumkante e gilt: e ist eine teuerste Kante in dem von e erzeugten Fundamentalkreis bzgl. T (*Kreiskriterium*).*
3. *Für jede Baumkante e gilt: e ist eine billigste Kante in dem von e erzeugten Fundamentalschnitt bzgl. T (*Schnittkriterium*).*



Welche Nicht-Baumkanten in Abbildung 3.15 verletzen das Kreiskriterium und welche Baumkanten erfüllen das Schnittkriterium nicht?

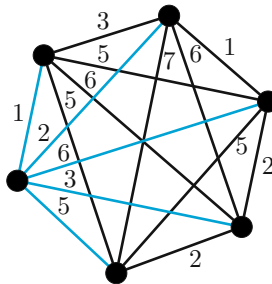


Abb. 3.15: Ist der spannende Baum kostenminimal?

Den Beweis von Satz 3.4 führen wir über einen Ringschluss. Zuerst zeigen wir: Aus (1) folgt (2), dann aus (2) folgt (3) und schließlich noch aus (3) folgt (1). Damit können wir uns einige Richtungen sparen (z.B. (2) \Rightarrow (1)).

Beweis von Satz 3.4. (1) \Rightarrow (2): Wir gehen also davon aus, dass T ein minimal spannender Baum ist. Der Beweis entspricht der Idee aus dem Beispiel von Seite 44. Angenommen, das Kreiskriterium ist für eine Nicht-Baumkante e nicht erfüllt. Sei f eine Kante auf dem Fundamentalkreis von e bzgl. T , die teurer ist als e , d.h. $c(f) > c(e)$. Dann erhalten wir einen neuen Baum T' , indem wir die Kante f aus T löschen und die Kante e hinzufügen (Abb. 3.16). Die Kosten von T' sind dann geringer als die von T . Das steht aber im Widerspruch zur Optimalität von T . Also wird das Kreiskriterium von jedem minimal spannenden Baum erfüllt.

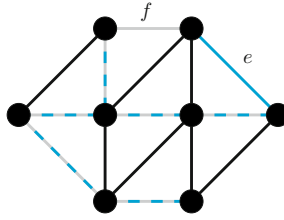


Abb. 3.16: Die beiden Bäume T' und T (grau und blau) unterscheiden sich nur in den Kanten e und f . Ihr Gewicht differiert um $|c(e) - c(f)|$.

(2) \Rightarrow (3): In diesem Fall gehen wir davon aus, dass jede Nicht-Baumkante eine teuerste Kante in ihrem Fundamentalkreis ist. Dann ist zu zeigen, dass jede Baumkante zu den billigsten Kanten in ihrem Fundamentalschnitt gehört.

Angenommen, es existiert eine Baumkante $e = (x, y)$, die nicht eine billigste Kante in ihrem Fundamentalschnitt ist. Dann gibt es eine billigere Kante $f = (u, v)$ in ihrem Fundamentalschnitt, d.h. $c(f) < c(e)$. Da in dem von e erzeugten Fundamentalschnitt nur e eine Baumkante ist (das haben wir in Lemma 3.3 gezeigt), ist f eine Nicht-Baumkante. Damit schließt die Nicht-Baumkante f einen Fundamentalkreis bezüglich T (Abb. 3.17). Als Nächstes zeigen wir, dass die Kante e in dem Fundamentalkreis von f liegt.

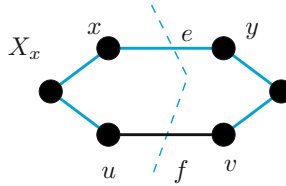


Abb. 3.17: Die Kante f schließt in T einen Kreis, auf dem auch die Kante e liegt.

Die Kante f liegt im Fundamentalschnitt von e , d.h. $u \in X_x$ oder $v \in X_x$ und der andere Knoten der Kante liegt in $V \setminus X_x$, wobei X_x wieder alle Knoten enthält, die von x aus in $T - e$ erreichbar sind. Ein Weg von v nach u in T muss also insbesondere auch eine Kante enthalten, die im Schnitt $\delta(X_x)$ liegt (Abb. 3.17). Da e die einzige Baumkante in diesem Schnitt ist, gehört e zu dem Weg von u nach v in T , bzw. liegt dann auf dem Fundamentalkreis von f . Jetzt können wir die Voraussetzung (2) anwenden und wissen, dass $c(f) \geq c(e)$ gilt. Das ist aber ein Widerspruch zu unserer Annahme, $c(f) < c(e)$.

(3) \Rightarrow (1): Dieses Mal gehen wir davon aus, dass jede Baumkante von T eine billigste Kante in ihrem Fundamentalschnitt ist, und zeigen damit, dass T ein minimal spannender Baum ist.

Sei T^* ein minimal spannender Baum, der maximal viele Kanten mit T gemeinsam hat. Wir zeigen jetzt, dass $T^* = T$ gilt und somit auch T ein minimal spannender Baum ist. Angenommen, die beiden Bäume sind nicht gleich. Dann gibt es eine Kante $e = (x, y)$, die in T , aber nicht in T^* liegt. In T bildet die Kante einen Fundamentalschnitt $\delta(X_x)$ und in T^* einen Fundamentalkreis C_e (Abb. 3.18).

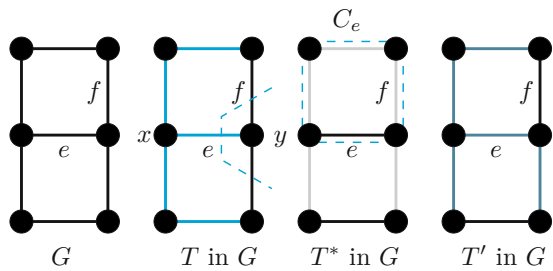


Abb. 3.18: Die Kante e erzeugt einen Fundamentalschnitt bzgl. T und einen Fundamentalkreis bzgl. T^* .

Wir zeigen nun, dass es eine Kante f gibt, die sowohl in C_e als auch in $\delta(X_x)$ liegt: Im Kreis C_e werden die Knoten x und y durch einen Weg verbunden, der die Kante e nicht enthält. Weiter wissen wir, dass der Knoten y nicht in X_x liegt. Damit gibt es eine Kante f , die zwischen den Knotenmengen X_x und $V \setminus X_x$ verläuft und auf C_e liegt. Eine ähnliche Argumentation hatten wir schon im letzten Beweisteil geführt. Betrachten wir nun diese Kante f . Sie ist dann insbesondere eine Baumkante von T^* und eine Nicht-Baumkante von T . Da T^* ein minimal spannender Baum ist, gilt das Kreiskriterium (das hatten wir schon gezeigt). Die Kante e ist eine Nicht-Baumkante von T^* und f liegt auf dem Fundamentalkreis C_e . Daraus folgt

$$c(e) \geq c(f). \tag{3.1}$$

Jedoch ist e eine Baumkante von T und f liegt im Fundamentalschnitt $\delta(X_x)$. Nach unserer Annahme gilt somit $c(e) \leq c(f)$. Zusammen mit der Gleichung 3.1 ergibt sich daraus $c(e) = c(f)$.

Betrachten wir nun den neuen Baum $T' = T^* - f + e$ (Abb. 3.18). Dieser hat mit T eine Kante mehr gemeinsam als T^* . Seine Kosten entsprechen denen von T^* , da das Austauschen der beiden Kanten keine neuen Kosten verursacht. Damit ist T' auch ein minimal spannender Baum. Hier ergibt sich nun ein Widerspruch. Schließlich hatten wir gefordert, dass T^* die Anzahl der gemeinsamen Kanten mit T unter allen minimal spannenden Bäumen maximiert. \square



Der Beweisgang war etwas langwierig. Welches Verfahren würden Sie entwickeln, um einen spannenden Baum auf seine Optimalität zu überprüfen? Wie sieht es mit dem folgenden spannenden Baum aus (Abb. 3.19). Ist er minimal?

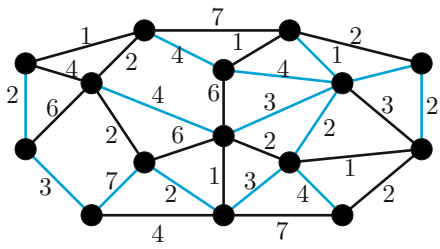


Abb. 3.19: Wie sieht hier ein minimal spannender Baum aus?

Mit Satz 3.4 können wir jetzt beweisen, dass die Algorithmen von Kruskal (Alg. 3.1) und Prim (Alg. 3.2) einen minimal spannenden Baum berechnen.

Satz 3.5. *Der Algorithmus von Kruskal berechnet einen MST.*

Beweis: Wir zeigen, dass der Algorithmus von Kruskal einen spannenden Baum T konstruiert, der das Kreiskriterium aus Satz 3.4 erfüllt. Sei e eine Nicht-Baumkante von T . Dann hat diese Kante zu dem Zeitpunkt, zu dem sie betrachtet wurde, einen Kreis in T geschlossen, d.h. ihren Fundamentalkreis (Abb. 3.20).

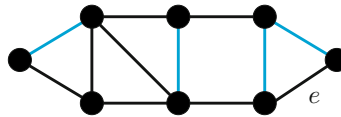


Abb. 3.20: Zum Zeitpunkt, zu dem e vom Algorithmus betrachtet wird, ist der „Baum“ T noch nicht vollständig. Aber die Kante e schließt schon jetzt einen Kreis.

Dieser Kreis besteht nur aus Kanten, die schon vorher zu T hinzugefügt wurden. Nach der Reihenfolge, in denen die Kanten betrachtet werden, ist somit die Kante e die teuerste von allen Kanten auf ihrem Fundamentalkreis. Damit ist das Kreiskriterium erfüllt, und wir haben einen minimal spannenden Baum berechnet. \square

Über das Optimalitätskriterium kann man auch beweisen, dass der Algorithmus von Prim einen minimal spannenden Baum bestimmt.

Satz 3.6. *Der Algorithmus von Prim berechnet einen MST.*

Wir haben jetzt zwei Arten kennen gelernt, mit denen man einen minimal spannenden Baum berechnen kann. Aber eigentlich ist nur wichtig, dass am Ende eines der Optimalitätskriterien erfüllt ist. Fallen Ihnen noch andere Möglichkeiten ein, solche Bäume zu bestimmen? Was ist, wenn man nicht zuerst die günstigsten Kanten nimmt, sondern mit den teuersten Kanten anfängt und diese nach und nach aus dem Graphen löscht, bis nur noch ein Baum übrig bleibt?



Mit diesem Kapitel schließen wir vorerst die Betrachtung von Bäumen ab. Als Nächstes betrachten wir eines der klassischen Reiseprobleme.

3.3 Aufgaben

Aufgabe 3.1. In einer Stadt soll der Straßenbelag erneuert werden. Deswegen hat die Stadt Kostenvoranschläge eingeholt, wie viel diese Renovierungsarbeiten für die einzelnen Straßenzüge kosten. Der Unternehmer Straßenfrei bietet an, für 10.000,00 Euro einen Kilometer zu sanieren.

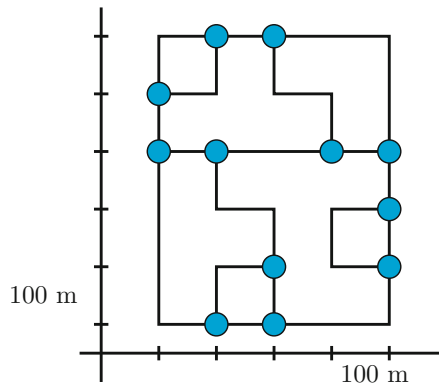


Abb. 3.21: Die Stadtverwaltung hat alle relevanten Orte mit einem blauen Kreis gekennzeichnet.

1. Welche Straßen sollte die Stadt renovieren lassen?
2. Bei Baubeginn stellt sich heraus, dass pro Straße doch eine erhebliche Vorarbeit von Nöten ist, die unabhängig von der Länge der Straße anfallen. Das Unternehmen verlangt pro Straße Fixkosten von 1.000,00 Euro. Sollte die Stadt ihren bisherigen Bauplan ändern?
3. Außerdem stellt Straßenfrei fest (nachdem er den Auftrag bekommen hat), dass der Aufwand pro Kilometer doch das Doppelte des Angenommen beträgt. Sollte die Stadt jetzt andere Straßenzüge bevorzugen?

Aufgabe 3.2. Der folgende Algorithmus dreht die Idee von Kruskal um, immer die günstigste Kante auszuwählen, ohne dabei einen Kreis zu schließen.

Algorithmus 3.3 Algorithmus für Pessimisten

Input: Graph $G = (V, E)$ mit Kantengewichten $c(e)$ für alle $e \in E$.

Output: Minimal spannender Baum T von G .

Schritt 1: Sortieren Sie die Kanten ihrer Gewichte nach absteigend, d.h. $c(e_1) \geq c(e_2) \geq \dots \geq c(e_m)$.

Schritt 2: Gehen Sie die Kanten e_i der Reihenfolge nach durch und löschen Sie sie aus G , wenn der Restgraph $G - e_i$ zusammenhängend bleibt.

1. Konstruieren Sie einen spannenden Baum für den Graphen G (Abb. 3.22) nach dem Algorithmus für Pessimisten.
2. Wie unterscheidet sich der Baum von dem minimal spannenden Baum nach Kruskal oder Prim?
3. Berechnet der Algorithmus wirklich einen minimal spannenden Baum?

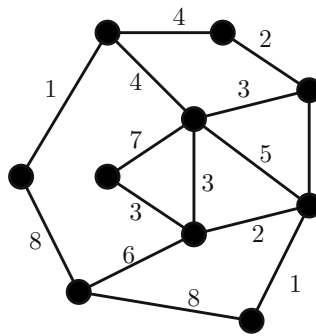


Abb. 3.22: Wie sieht ein MST in G aus?

Aufgabe 3.3. Sind die mit Blau in die Graphen G_1 und G_2 eingezeichneten Bäume minimal spannend (Abb. 3.23)? Wie können die Bäume gegebenenfalls in minimal spannende Bäume umgewandelt werden?

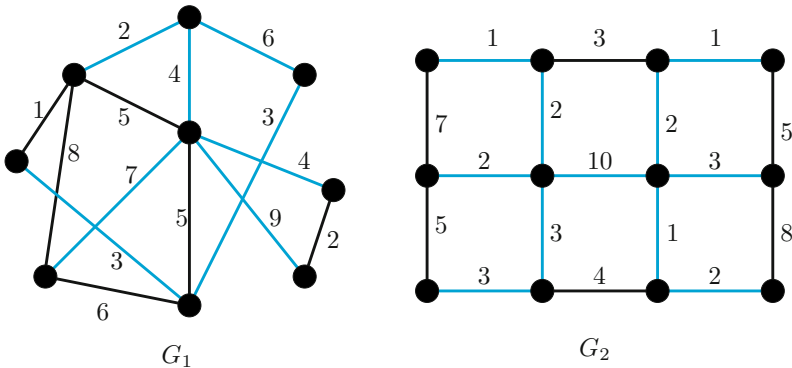


Abb. 3.23: Die blauen Kanten spannen die jeweiligen Graphen auf, aber sind sie auch minimal spannend?

Aufgabe 3.4. Bisher wurde immer die Sichtweise einer Stadt oder des Kalifen betrachtet, die möglichst wenig Kosten erreichen wollten. Welchen spannenden Baum würde ein Bauunternehmer seinen jeweiligen Partnern anbieten? Können die Algorithmen von Prim und Kurshkal auch verwendet werden, um einen maximal spannenden Baum zu berechnen?

Aufgabe 3.5. Betrachten Sie den folgenden minimal spannenden, in blau eingezeichneten Baum (Abb. 3.24).

1. In welcher Reihenfolge müssen die Kanten sortiert werden, damit der Algorithmus von Kruskal diesen Baum berechnet?
2. Nehmen wir an, der Algorithmus von Prim startet im Knoten a . Welche Kante aus dem Schnitt müsste der Algorithmus von Prim jeweils wählen, damit der Baum entsteht?

3. Zeigen Sie: Sei T ein minimal spannender Baum. Dann gibt es immer eine Reihenfolge der Kanten, sodass der Algorithmus von Kruskal diesen Baum berechnet.

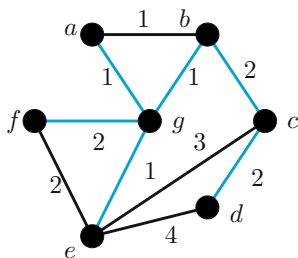


Abb. 3.24: Könnte der spannende Baum auch von dem Algorithmus von Kruskal berechnet worden sein?

Aufgabe 3.6. Beweisen oder widerlegen Sie die folgenden Aussagen:

1. Wenn alle Kantengewichte in einem zusammenhängenden Graphen unterschiedlich sind, gibt es einen eindeutigen minimal spannenden Baum.
2. Wenn es einen eindeutigen minimal spannenden Baum gibt, dann sind auch alle Kantengewichte unterschiedlich.

Aufgabe 3.7. Wie sind die Kantengewichte $\{1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6\}$ in dem Graphen G (Abb. 3.25) verteilt, damit ein eindeutiger spannender Baum entsteht? Wie ergibt sich ein nicht eindeutiger spannender Baum?

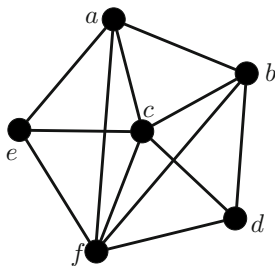


Abb. 3.25: Durch geschicktes Verteilen der Kantengewichte erhält G einen eindeutig spannenden Baum.

Aufgabe 3.8. Betrachten Sie den K_n auf den Knoten $\{1, \dots, n\}$. Für jede Kante seien die Kosten $c((i, j)) = i + j$.

1. Konstruieren Sie einen minimal spannenden Baum in K_n .
2. Welches Gewicht hat der in 1. berechnete Baum?
3. Ist der Baum eindeutig?

Aufgabe 3.9. Zwei Bäume heißen 1-benachbart, wenn sie $n - 2$ gemeinsame Kanten haben (d.h. jeder Baum enthält genau eine Kante, die nicht in dem anderen enthalten ist). Seien T und T' zwei spannende Bäume von G , die k gemeinsame Kanten besitzen. Wie kann eine Folge T_0, \dots, T_{n-1-k} von 1-benachbarten spannenden Bäumen konstruiert werden, sodass der erste Baum T entspricht und der letzte T' ?

Aufgabe 3.10. Beweisen oder widerlegen Sie: In jedem minimal spannenden Baum ist für jeden Knoten eine Kante e_v mit $c(e_v) \leq c(e)$ für alle zu v inzidenten Kanten $e \in E$ enthalten.

Aufgabe 3.11. Sei $G = (V, E)$ ein Graph und $c_1 : E \rightarrow \mathbb{N}$ und $c_2 : E \rightarrow \mathbb{N}$ seien zwei Kantengewichtsfunktionen. Es gelte

$$c_1(e_1) \leq c_1(e_2) \Leftrightarrow c_2(e_1) \leq c_2(e_2) \quad \forall e_1, e_2 \in E.$$

Dann ist T genau dann ein minimal spannender Baum bzgl. c_1 in G , wenn er auch ein minimal spannender Baum bzgl. c_2 ist.

Lösungen zu den Fragen im Text

- 11 Da T ein Baum ist, kann er nicht mehr als $n - 1$ Kanten enthalten.
 12 **Satz.** In einem Baum existiert immer ein eindeutiger einfacher Weg zwischen zwei Knoten.

Beweis: Zwischen zwei Knoten in einem Baum existiert immer ein Weg, da jeder Baum zusammenhängend ist. Angenommen, in einem Baum gibt es zwei verschiedene einfache Wege p_1 und p_2 zwischen zwei Knoten u und v . Sei x_1 der erste Knoten von u aus gesehen, der auf p_1 und p_2 liegt, an dem sich jedoch die Wege trennen (Abb. 3.26). Sei weiter x_2 der nächste Knoten, den beide Wege wieder enthalten. Schränken wir die Wege p_1 und p_2 auf die Teile zwischen x_1 und x_2 ein, so bilden diese einen Kreis. Das ist aber ein Widerspruch zur Definition eines Baums. \square

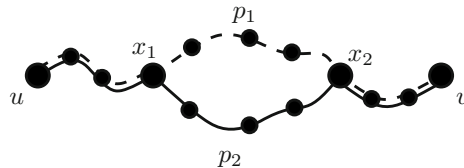


Abb. 3.26: Die Wege p_1 und p_2 verbinden die Knoten u und v und bilden deswegen einen Kreis.

4 Euler-Touren und -Wege

Übersicht

| | | |
|-----|---|----|
| 4.1 | Das Königsberger Brückenproblem | 57 |
| 4.2 | Die Algorithmen von Hierholzer und Fleury | 61 |
| 4.3 | Euler-Wege oder das Haus vom Nikolaus | 65 |
| 4.4 | Aufgaben | 68 |

4.1 Das Königsberger Brückenproblem

Im Jahr 1736 veröffentlichte der Mathematiker Leonard Euler einen Artikel über das Königsberger Brückenproblem und entwickelte eine Lösung mit Hilfe der Graphentheorie (jedenfalls nennen wir seine Konstruktion heute so). Worum es bei diesem Problem geht, hat er folgendermaßen formuliert:

„Das Problem, das ziemlich bekannt sein soll, war folgendes. Zu Königsberg in Preußen ist eine Insel A , genannt „der Kneiphof“, und der Fluss, der sie umfließt, teilt sich in zwei Arme, wie dies aus Fig. 4.1 ersichtlich ist. Über die Arme dieses Flusses führen sieben Brücken a, b, c, d, e, f und g . Nun wurde gefragt, ob jemand seinen Spaziergang so einrichten könne, dass er jede dieser Brücken einmal und nicht mehr als einmal überschreite. Es wurde mir gesagt, dass einige diese Möglichkeit verneinen, andere daran zweifeln, dass aber niemand sie erhärte. Hieraus bildete ich mir folgendes höchst allgemeine Problem: Wie auch die Gestalt des Flusses und seine Verteilung der Arme, sowie die Anzahl der Brücken ist, zu finden, ob es möglich sei jede Brücke genau einmal zu überschreiten oder nicht.

Was das Königsberger Problem von den sieben Brücken betrifft, so könnte man es lösen durch eine genaue Aufzählung aller Gänge, die möglich sind; denn dann wüsste man, ob einer derselben der Bedingung genügt oder keiner. Diese Lösungsart ist aber wegen der großen Zahl von Kombinationen zu mühsam und schwierig, und zudem könnte sie in anderen Fragen, wo noch viel mehr Brücken vorhanden sind, gar nicht mehr angewendet werden. Würde die Untersuchung in der eben erwähnten Weise geführt, so würde Vieles gefunden, wonach gar nicht gefragt war; dies ist zweifellos der Grund, warum dieser Weg so beschwerlich wäre. Darum habe ich diese Methode fallengelassen und eine andere gesucht, die nur so weit reicht,

dass sie erweist, ob ein solcher Spaziergang gefunden werden kann oder nicht; denn ich vermutete, dass eine solche Methode viel einfacher sein würde...“

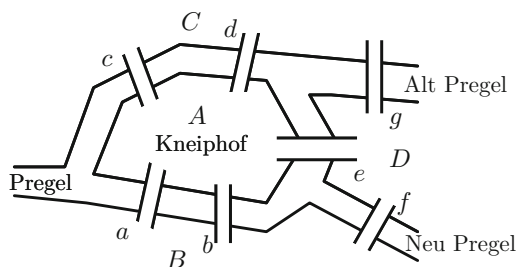


Abb. 4.1: Die Abbildung gibt eine leicht vereinfachte Version der Skizze von Euler wieder.



Soweit Eulers Einführung in das Königsberger Brückenproblem. Würden Sie eher dafür oder dagegen stimmen, dass es einen solchen Spaziergang gibt?

Die Methode zur Lösung des Königsberger Brückenproblems, von der Euler spricht, beinhaltete das erste Mal eine Konstruktion, die wir heute als Graphen bezeichnen. Dabei verwandelte er die Landmassen zu Knoten und die Brücken zu Kanten (Abb. 4.2).

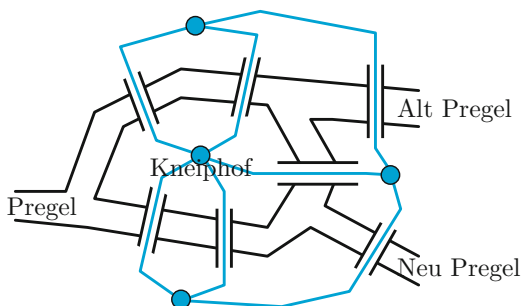


Abb. 4.2: Euler modellierte die Landmassen von Königsberg als Knoten und die Brücken als Kanten.

Die Karte von Königsberg ergibt dabei einen **Multigraphen**, d.h. einen Graphen mit parallelen Kanten und Schleifen. In diesem Kapitel wollen wir auch diese Art von Graphen zulassen. Der von Euler gesuchte Spaziergang, bei dem alle Brücken genau einmal überquert werden, entspricht dann einem Kreis, der alle Kanten des Graphen enthält. Zu Ehren von Euler bezeichnen wir Graphen, die einen solchen Kreis (manchmal auch Tour genannt) enthalten, auch als **eulersch** (Abb. 4.3).

Definition. Ein Graph heißt **eulersch** oder ist ein **Euler-Graph**, wenn es in ihm eine Euler-Tour gibt. Ein **Euler-Kreis** oder eine **Euler-Tour** in einem Graphen ist ein Kreis, der jede Kante genau einmal enthält.

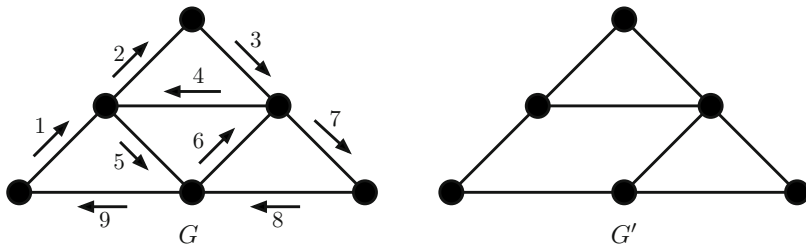


Abb. 4.3: Der Graph G ist eulersch. Die Pfeile und Zahlen geben eine Euler-Tour an. Der Graph G' ist hingegen nicht eulersch.

Übersetzt in die Graphentheorie müssen wir beim Königsberger Brückenproblem also entscheiden, ob der Graph aus der Abbildung 4.2 eulersch ist oder nicht. Euler lieferte zur Lösung des Königsberger Brückenproblems eine vollständige Charakterisierung aller eulerschen Graphen.

Satz 4.1 (Euler 1736). *Ein zusammenhängender Graph ist genau dann eulersch, wenn jeder Knoten einen geraden Grad hat.*

Der Satz enthält zwei wichtige Aussagen: Wenn ein Graph eulersch ist, dann hat jeder Knoten einen geraden Grad. Und in umgekehrter Argumentationsrichtung gilt: Hat ein Graph nur Knoten mit geradem Grad, so ist er eulersch. In seiner Veröffentlichung von 1736 hat Euler nur die erste Bedingung bewiesen. Damit war das Königsberger Brückenproblem jedoch schon gelöst. Zu welchem Schluss kommt man nach Satz 4.1? Den ersten vollständigen Beweis erbrachte Hierholzer im Jahr 1873, indem er einen Algorithmus konstruierte, der eine Euler-Tour berechnet. Diesen werden wir in Abschnitt 4.2 kennen lernen. Bevor wir nun den Satz von Euler beweisen, zeigen wir ein kleines Lemma, das uns den Beweis vereinfacht.



Lemma 4.2. *Gegeben sei ein Graph, in dem alle Knoten geraden Grad haben. Sei W ein Weg mit den Endknoten v_0 und v_k , der keine Kante doppelt durchläuft. Falls jede zu v_k inzidente Kante auch in W vorkommt, gilt $v_0 = v_k$. Der Weg W ist also ein Kreis.*

Beweis: Betrachten Sie einen Knoten v auf dem Weg W , der kein Endknoten von W ist. Jedes Mal, wenn der Weg den Knoten durchläuft, gibt es eine Kante, die in den Knoten hineinführt, und eine Kante, die aus dem Knoten herausführt. Damit gibt es eine gerade Anzahl inzidenter Kanten zu v , die ebenfalls auf dem Weg W liegen. Wie sieht es mit den beiden Endknoten von W aus? Muss hier auch eine gerade Anzahl an Kanten inzident sein?¹³

Sei $W = v_0 v_1 \dots v_k$ ein Weg, sodass alle zu v_k inzidenten Kanten auf dem Weg liegen, aber ohne dass eine Kante doppelt durchlaufen wird (Abb. 4.4). Der Weg W kann also am Knoten v_k nicht mehr verlängert werden. Nehmen wir $v_0 \neq v_k$ an. Dann endet in v_k eine ungerade Anzahl an Kanten. Da der Grad von v_k gerade

ist, existiert eine Kante, die zu v_k inzident ist und nicht im Weg W enthalten ist. Durch diese Kante kann der Weg W verlängert werden, ohne dass der neue Weg eine Kante mehrfach enthält. Dies bildet einen Widerspruch zur Wahl von W . \square

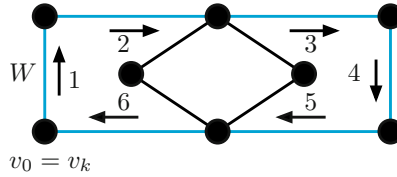


Abb. 4.4: Der Weg W kann am Knoten v_k nicht verlängert werden. Deswegen muss W jedoch nicht zwingend eine Euler-Tour sein.

Damit können wir nun den Satz von Euler beweisen.

Beweis, Satz von Euler 1736. „ \Rightarrow “ Wir beginnen mit der Hin-Richtung (Wenn der Graph G eulersch ist, dann hat jeder Knoten geraden Grad): Sei T eine Euler-Tour in G , d.h. ein Kreis, der alle Kanten durchläuft. Sei v ein Knoten, der in der Tour k -mal vorkommt. Dann führt jedes Mal eine Kante in den Knoten hinein und eine Kante aus dem Knoten heraus. Alle Kanten, die zum Knoten v inzident sind, werden genau einmal besucht. Insgesamt sind es $2k$ Kanten, und damit ist der Grad am Knoten v gerade.

„ \Leftarrow “ Wir wollen nun die Rück-Richtung beweisen (Wenn jeder Knoten geraden Grad hat, dann existiert eine Euler-Tour): Betrachten wir dazu einen Weg $W = v_0 e_0 v_1 e_1 \dots e_{k-1} v_k$ maximaler Länge in G , der keine Kante doppelt benutzt. Nach Lemma 4.2 ist W ein Kreis.

Fall 1: W benutzt alle Kanten. Dann ist W eine Euler-Tour und wir sind fertig.

Fall 2: Angenommen, es existiert eine Kante $e = (u, v) \in E$, die nicht in W enthalten ist. Die Idee besteht nun darin zu zeigen, dass wir einen Knoten in W finden können, der zu einer noch nicht besuchten Kante $e' = (v', v_i)$ inzident ist. Mit dieser Kante können wir dann den Weg W folgendermaßen verlängern:

$$W' := v' e' v_i e_i v_{i+1} e_{i+1} \dots e_{k-1} v_k e_0 v_1 e_1 \dots e_{i-1} v_i$$

Der Weg W' ist länger als W . Das ist aber ein Widerspruch zur Wahl von W , da W der längste Weg in G sein sollte.

Wir müssen also noch zeigen, dass die Kante e' existiert. Nach unserer Fallunterscheidung existiert eine Kante $e = (u, v)$, die nicht auf dem Weg liegt. Falls e zu einem Knoten in W inzident ist, stellt e sofort die gesuchte Kante dar (Fall A in Abb. 4.5). Ist dies nicht der Fall, so existiert ein Weg \overline{W} von v nach W , d.h. ein Weg in G , der nur Kanten enthält, die nicht in W liegen. Welche Eigenschaft von G garantiert, dass es einen solchen Weg gibt?¹⁴ Sei v_i der Endknoten von W' (Abb. 4.5, Fall B). Dann ist die gesuchte Kante e' die letzte Kante (v', v_i) auf dem Weg \overline{W} mit $v' \in \overline{W}$ und $v_i \in W$. \square



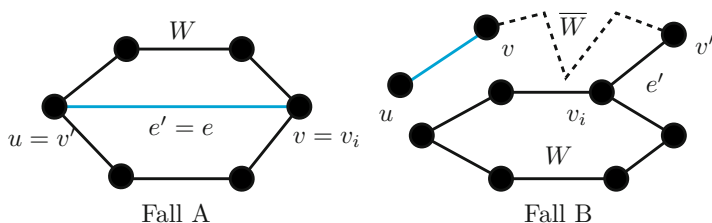


Abb. 4.5: Ist die Kante e inzident zu einem Knoten in W , so ist sie die gesuchte Kante (Fall A). Ansonsten liegt die gesuchte Kante auf einem Weg \overline{W} von e zu W (Fall B).

Mit Hilfe des Satzes ist es nun einfach, zu überprüfen, ob ein Graph eulersch ist oder nicht: Man braucht nur die Gradzahlen an den einzelnen Knoten zu berechnen. Sind sie alle gerade, so ist er eulersch, ansonsten nicht. Allerdings interessiert man sich ja auch noch für die tatsächliche Euler-Tour, sollte sie vorhanden sein. Im nächsten Abschnitt werden wir zwei Algorithmen kennen lernen, die uns zu einem Euler-Graphen eine Euler-Tour bestimmen. Auch die Korrektheit dieser Algorithmen beweist den Satz von Euler.

4.2 Die Algorithmen von Hierholzer und Fleury

Der Mathematiker Carl Hierholzer wurde 1840 in Freiburg im Breisgau geboren. Er studierte Mathematik am Polytechnikum in Karlsruhe. 1865 promovierte er in Heidelberg bei dem aus Königsberg stammenden Ludwig Otto Hesse. Wahrscheinlich kam er über Hesse mit dem Königsberger Brückenproblem in Berührung und nahm dazu seine eigene Forschung auf. Allerdings wurde sein Algorithmus, der, falls möglich, eine Euler-Tour berechnet und damit die Rück-Richtung des Satzes von Euler beweist, erst zwei Jahre nach seinem Tod im Jahr 1871 von einigen seiner Freunde veröffentlicht.

Am besten besprechen wir den Algorithmus von Hierholzer an einem Beispiel.

Beispiel. In der Abbildung 4.6 (a) ist ein Graph gegeben, für den wir eine Euler-Tour bestimmen wollen.

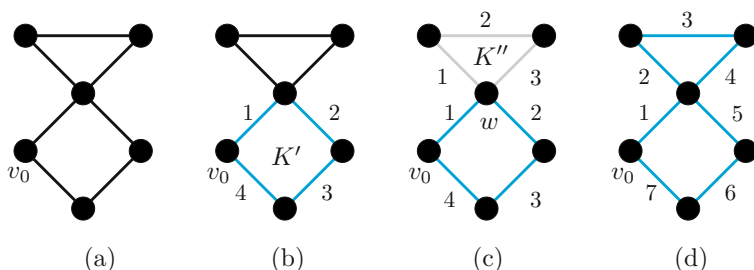


Abb. 4.6: In Teil (c) sind die beiden Kreise K' und K'' dargestellt, bevor sie zu einem Kreis (hier einer Euler-Tour) verschmolzen werden.

Jeder Knoten hat geraden Grad. Als Erstes wählen wir einen beliebigen Knoten v_0 und konstruieren einen Kreis K von v_0 aus (Abb. 4.6 (b)). Dieser enthält noch nicht alle Kanten und ist somit noch keine Euler-Tour. Um ihn zu erweitern, suchen wir einen Knoten w , von dem eine noch nicht besuchte Kante ausgeht. Danach bestimmen wir einen weiteren Kreis K'' aus den restlichen Kanten. In unserem Fall erhalten wir den blauen Kreis in der Abbildung 4.6 (c). Nachdem wir wieder in w angekommen sind, verschmelzen wir die beiden Kreise und überprüfen erneut, ob der Kreis eine Euler-Tour ist. Der neue Kreis aus Abbildung 4.6 (d) enthält nun alle Kanten, womit eine Euler-Tour in diesem Graphen gefunden wurde. ■

Die Idee des Algorithmus ist es, nach und nach Kreise zu konstruieren und diese zu einem Kreis zusammenzufügen. Etwas formaler lässt sich der Algorithmus in drei Schritten beschreiben:

Algorithmus 4.1 Algorithmus von Hierholzer

Input: zusammenhängender Graph $G = (V, E)$, der nur Knoten mit geradem Grad hat.

Output: Euler-Tour in G .

- Schritt 1: • Wählen Sie einen Knoten $v_0 \in V$.
 • Wählen Sie nach und nach unbesuchte Kanten, bis es nicht weitergeht. Der so entstandene Weg ist ein Kreis K .
- Schritt 2: Prüfen Sie, ob K eine Euler-Tour ist. Wenn ja, dann STOPP, ansonsten gehen Sie zu Schritt 3.
- Schritt 3: • Setzen Sie $K' = K$.
 • Wählen Sie einen in K' enthaltenen Knoten w , der mit einer nicht in K' enthaltenen Kante inzident ist.
 • Konstruieren Sie wie in Schritt 1 von w ausgehend einen Kreis K'' , der keine Kante von K' enthält.
 • Fügen Sie K' und K'' wie folgt zu K zusammen: Durchlaufen Sie K' von v_0 bis w , durchlaufen Sie dann K'' und anschließend den Rest von K' .
 • Gehen Sie zu Schritt 2.
-

Obwohl Algorithmus 4.1 im obigen Beispiel funktioniert, muss seine Korrektheit wie für alle mathematischen Aussagen erst bewiesen werden. Meistens ist diese Art von Beweisführung sehr technisch.

Satz 4.3. *Gegeben sei ein zusammenhängender Graph, in dem alle Kanten geraden Grad haben. Dann berechnet der Algorithmus von Hierholzer eine Euler-Tour.*

Beweis: Wir müssen nach und nach alle Schritte durchgehen und uns vergewissern, dass jeder einzelne durchführbar ist.

Wir starten mit Schritt 1: Nach der Wahl eines beliebigen Knoten v_0 bestimmen wir einen Weg W , der keine Kanten doppelt enthält und nicht verlängerbar ist. Dann ist W nach Lemma 4.2 ein Kreis. Schritt 1 ist somit sinnvoll.

In Schritt 2 gilt es zu überprüfen, ob der bisher gefundene Kreis K eine Euler-Tour ist. Dazu wird untersucht, ob jede Kante von G in K enthalten ist. Nur wenn dies der Fall ist, stoppt der Algorithmus, ansonsten nicht. Damit wird auf jeden Fall eine Euler-Tour berechnet.

Bei Schritt 3 müssen wir zunächst zeigen, dass ein Knoten $w \in K$ existiert, dessen Kante noch nicht besucht wurde. Damit Schritt 3 ausgeführt wird, muss es eine noch nicht besuchte Kante $e = (u, v) \in E(G) \setminus E(K)$ geben. Um den gewünschten Knoten w zu finden, unterscheiden wir zwei Fälle (ähnlich wie im Beweis zum Satz von Euler):

Fall 1: u oder v sind in K enthalten. Dann entsprechen beide dem gesuchten Knoten.

Fall 2: Weder u noch v sind in K enthalten. Da G zusammenhängend ist, existiert ein Weg W' von v nach W . Der letzte Knoten auf diesem Weg entspricht den Anforderungen.

Der letzte Teil in Schritt 3 besteht aus dem Zusammenkleben der Kreise K' und K'' . (K'' ist ein Kreis, da $G \setminus K'$ nur Knoten mit geradem Grad hat und K'' wieder ein nicht verlängerbarer Weg ist.) Anhand der formalen Schreibweise beider Kreise sieht man sofort, dass K ein Kreis ist. Da der Kreis K'' mindestens eine neue Kante dem Kreis K' hinzufügt, kann Schritt 3 nicht häufiger als $0.5 \cdot m$ durchgeführt werden. Der Algorithmus endet also immer, sodass seine Korrektheit bewiesen ist. \square

Folglich gilt: Der Hierholzer-Algorithmus berechnet in jedem Euler-Graphen eine Euler-Tour. Versuchen Sie sich noch einmal an dem folgenden Graphen (Abb. 4.7).

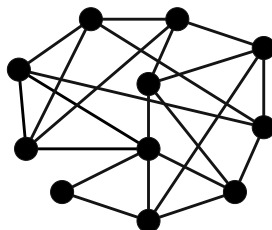


Abb. 4.7: Ist der Graph eulersch? Wie sieht die Anwendung des Algorithmus von Hierholzer in diesem Fall aus?

Bei der Konstruktion haben Sie die Tour an einigen Stellen unterbrochen, um ein weiteres Teilstück einzufügen. Eine Euler-Tour ohne diese Zwischenschritte direkt

zu konstruieren, gelingt mit Hilfe des Algorithmus von Fleury. Zur Vorbereitung benötigen wir jedoch noch eine spezielle Eigenschaft der Euler-Graphen.

Lemma 4.4. *Wenn in einem Graphen alle Knotengrade gerade sind, dann hat er keine Brücke.*

Zur Erinnerung: Eine *Brücke* ist eine Kante, deren Löschen den Graphen in mehrere Zusammenhangskomponenten zerfallen lässt (Definition auf Seite 13).

Beweis: Wir gehen davon aus, dass der Graph G zusammenhängend ist, da wir ansonsten das Lemma auf die einzelnen Zusammenhangskomponenten anwenden könnten. Angenommen, es existiert eine Kante $e = (v_1, v_2)$, die eine Brücke in G ist (Abb. 4.8). Der Graph $G - e$ besteht aus zwei Zusammenhangskomponenten. Sei G_1 die Komponente, die v_1 enthält, und G_2 diejenige, die v_2 enthält. Alle Knoten in G_1 haben geraden Grad mit Ausnahme des Knoten v_1 , dessen Grad nach dem Löschen einer Kante (der Kante e) ungerade ist. Der Teilgraph G_1 ist für sich genommen wieder ein Graph. Deswegen muss er auch alle Eigenschaften von Graphen erfüllen, insbesondere die, dass die Anzahl der Knoten ungeraden Grades gerade ist (siehe *Handshaking*-Lemma 1.2). Wir hatten aber gesagt, dass alle Knoten in G_1 bis auf v_1 geraden Grad haben. Damit ergibt sich jedoch ein Widerspruch. \square

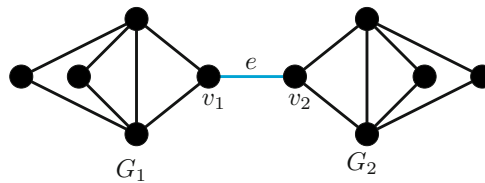


Abb. 4.8: Falls e eine Brücke ist, muss in G_1 ein weiterer Knoten ungeraden Grades existieren.

Der Algorithmus von Fleury lässt sich so zusammenfassen: Achten Sie darauf, die nächste Kante stets so zu wählen, dass die Menge der noch nicht gezeichneten Kanten zusammenhängend bleibt. Oder: Wählen Sie niemals eine Brücke aus dem Restgraphen, d.h. dem Graphen ohne die bisherige Tour (Abb. 4.9).

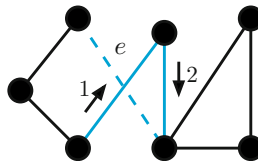


Abb. 4.9: Wählt der Algorithmus die Kante e , so kann der Weg nicht mehr in einem Zug zu einer Euler-Tour erweitert werden.

Etwas formaler liest sich der Algorithmus so:

Algorithmus 4.2 Algorithmus von Fleury.

Input: Euler-Graph.

Output: Euler-Tour.

Schritt 1: Beginnen Sie mit einer Kante.

Schritt 2: Wählen Sie als nächste Kante eine aus, die die folgenden zwei Eigenschaften erfüllt:

1. Sie ist inzident zu der zuletzt betrachteten Kante.
2. Sie ist keine Brücke in dem Graphen, der aus den unbesuchten Kanten besteht.

Schritt 3: Wiederholen Sie Schritt 2 bis alle Kanten besucht sind.

Es stellt sich natürlich die Frage, wie sich überprüfen lässt, ob eine Kante eine Brücke im Restgraphen bildet. Eine Kante ist genau dann keine Brücke, wenn sie auf einem Kreis liegt (Verneinung von Lemma 1.1). Wie kann diese Information für ein Kriterium genutzt werden, das angibt, wann wir eine Kante wählen dürfen und wann nicht?¹⁵

Die Algorithmen 4.1 und 4.2 werden wir erneut benötigen, um einen Euler-Weg zu berechnen. Was ein Euler-Weg ist und wie dieser charakterisiert werden kann, werden wir im nächsten Abschnitt besprechen.



4.3 Euler-Wege oder das Haus vom Nikolaus

Vom Haus des Nikolaus (Abb. 4.10) haben viele sicher schon im Kindergarten gehört. Die „Kunst“ besteht bekanntlich darin, es ohne Absetzen des Stiftes zu zeichnen. Oder vielleicht kennt der eine oder andere die Tierzeichnungen von Pablo Picasso, die mit derselben Methode nachgemalt werden können (Augen ausgeschlossen). Mathematisch gesehen spricht man hier von Euler-Wegen.

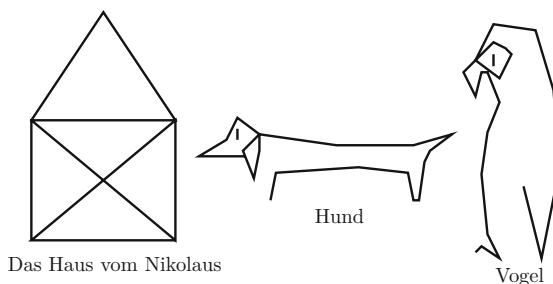


Abb. 4.10: Das Haus vom Nikolaus und auch die beiden Tiere (Picasso nachgeahmt) lassen sich ohne Absetzen des Stiftes nachzeichnen.

Definition. Ein Weg in einem Graphen, der alle Kanten genau einmal enthält, heißt **Euler-Weg**.

Die Definitionen von Euler-Tour und Euler-Weg sind sich recht ähnlich. Der Unterschied besteht darin, dass ein Euler-Weg in einem Knoten x starten kann und in einem anderen Knoten y endet, wohingegen bei einer Euler-Tour $x = y$ gilt. Wie sieht ein Euler-Weg für die einzelnen Graphen in Abbildung 4.10 aus? Gibt es in ihnen auch eine Euler-Tour?

Wie man sich gut vorstellen kann, gibt es auch hier eine einfache Charakterisierung von Graphen, die einen Euler-Weg enthalten.

Satz 4.5. *Ein Graph enthält genau dann einen Euler-Weg, wenn maximal zwei Knoten einen ungeraden Grad haben.*

Der Beweis ist nicht schwer. Versuchen Sie ihn sich selber zu überlegen: Was war genau der Unterschied zwischen einer Euler-Tour und einem Euler-Weg? Wie könnte man einen gegebenen Graphen mit maximal zwei Knoten ungeraden Grades in einen Euler-Graphen umwandeln?¹⁶

Stellen wir uns nun vor, eine Abbildung hat mehr als zwei Knoten ungeraden Grades. Dann könnte man sich fragen, wie häufig wir den Stift mindestens absetzen müssen, um sie zu zeichnen. Übertragen auf die Graphentheorie heißt das: Wir suchen nach einer Menge von Wegen, in der jede Kante in genau einem Weg vorkommt. Auch hier ist man natürlich an einer einfachen Antwort interessiert. Deswegen führen wir noch die folgende Definition ein.

Definition. Zwei Wege, die keine gemeinsame Kante besitzen, heißen **kantendisjunkt**.

Jetzt können wir die Antwort im folgenden Satz formulieren.

Satz 4.6. *Gegeben sei ein zusammenhängender Graph mit $2k$ Knoten ungeraden Grades und $k \geq 1$. Dann ist die minimale Anzahl von paarweise kantendisjunkten Wegen, die jede Kante des Graphen enthalten, k .*

Übersetzt bedeutet das: Ein Graph mit $2k$ Knoten ungeraden Grades kann mit k -maligem Absetzen des Stiftes gezeichnet werden. Der Satz beinhaltet mit $k = 1$ natürlich auch die Rück-Richtung von Satz 4.5. Gilt die Aussage auch noch, wenn der Graph nicht zusammenhängend ist?¹⁷ Wieso können wir von $2k$ Knoten mit ungeradem Grad ausgehen?

Beweis Satz 4.6. Sei W ein Weg in dem gegebenen Graphen G . Wie wir schon früher festgestellt haben, durchläuft der Weg W an jedem Knoten eine gerade Anzahl an Kanten, außer eventuell an seinen beiden Endknoten (siehe Beweis

zu Lemma 4.2). Deswegen muss an jedem Knoten mit ungeradem Grad in einer Wegzerlegung der Kanten von G ein Weg enden. Da jeder Weg maximal zwei unterschiedliche Endknoten hat, benötigen wir bei $2k$ Knoten mit ungeradem Grad mindestens k Wege, um alle Kanten zu überdecken. Das bedeutet also, dass die minimale Anzahl an Wegen mindestens k sein muss.

Wir müssen nun zeigen, dass k Wege auch ausreichen. Erweitern Sie dazu den Graphen G um k Kanten, sodass jeweils zwei Knoten ungeraden Grades miteinander verbunden sind (Abb. 4.11). Jetzt haben alle Knoten geraden Grad und wir können eine Euler-Tour in dem neuen Graphen bestimmen. Löschen wir aus dieser Euler-Tour die k extra Kanten, so erhalten wir k Wege, die kantendisjunkt sind und alle Kanten von G durchlaufen. Die k Wege bilden nun eine Wegzerlegung von G , womit bewiesen ist, dass eine minimale Wegzerlegung k Wege enthält. \square

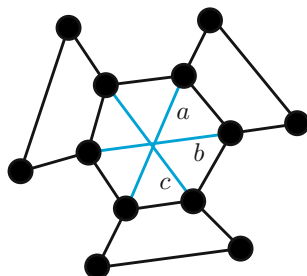


Abb. 4.11: Durch das Hinzufügen der blauen Kanten a , b und c erhalten wir einen eulerschen Graphen.

Der Beweis liefert uns eine Anleitung, wie wir eine Pfadzerlegung eines Graphen in disjunkte Pfade bestimmen können. Wie würden Sie diese Anleitung formulieren? Welche Wege ergeben sich bei der Anwendung auf die Graphen aus der Abbildung 4.12?

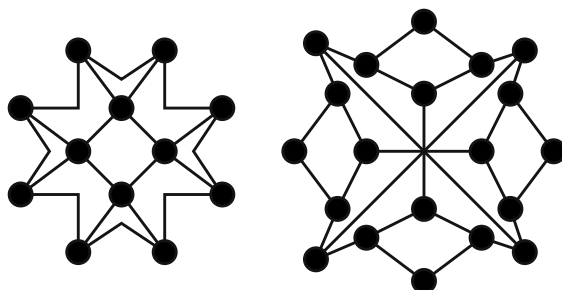


Abb. 4.12: Wie viele Knoten ungeraden Grades existieren in den beiden Graphen?

Im nächsten Kapitel werden wir uns mit dem wohl bekanntesten Rundreiseproblem, dem *Travelling-Salesman-Problem* oder dem Handlungsreisenden-Problem, beschäftigen. Dabei sollen nicht alle Kanten, sondern alle Knoten in einer Tour vorkommen.

4.4 Aufgaben

Aufgabe 4.1. Ein neu eingerichtetes Museum soll mit einem Rundgang ausgestattet werden (Abb. 4.13). An allen Wänden hängen Bilder, von denen die Tour keines auslassen soll. Außerdem will der Direktor natürlich die Besucher nicht langweilen und sie zweimal an ein und demselben Bild vorbeiführen. Der Rundgang soll beim Eingang beginnen und dort auch wieder enden. Welche Route würden Sie vorschlagen?

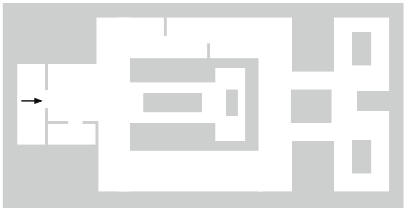


Abb. 4.13: Gibt es einen Rundgang, bei dem man an jedem Bild genau einmal vorbeikommt?

Aufgabe 4.2. Wie könnte ein Rundgang mit Start am Eingang in dieser Wohnung aussehen, sodass man durch jede Tür genau einmal kommt (Abb. 4.14)?

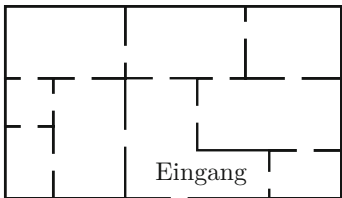


Abb. 4.14: Die Wohnung hat viele Türen. Aber kann man auf einem Rundgang jede Tür genau einmal durchqueren?

Aufgabe 4.3. Wenden Sie auf den folgenden Graphen (Abb. 4.15) zuerst den Algorithmus von Hierholzer und dann den Algorithmus von Fleury an, um eine Euler-Tour zu berechnen.

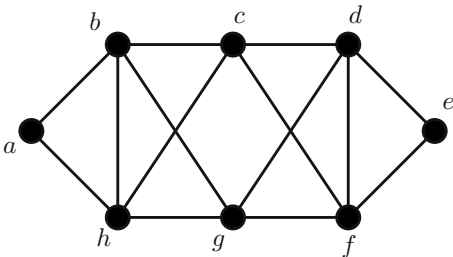


Abb. 4.15: Der Graph G ist eulersch, da jeder Knoten geraden Grad hat.

Aufgabe 4.4. Für welche Werte von a und b ist der $K_{a,b}$ eulersch?

Aufgabe 4.5. In einem Dominospiel gibt es für jedes Zahlenpaar, das aus den Zahlen 0 bis 6 besteht, einen Dominostein. Ziel des Spieles ist es, die Steine so zu legen, sodass immer zwei gleiche Zahlen miteinander benachbart sind (Abb. 4.16).

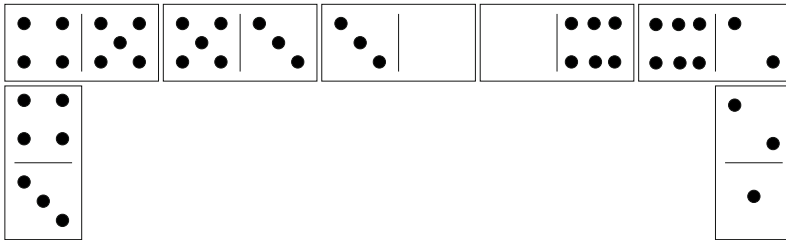


Abb. 4.16: Existiert eine Kette bestehend aus allen Dominosteinen?

1. Wie kann nach dieser Regel ein Kreis mit allen 28 Steinen gelegt werden (Abb. 4.16)?
2. Eine etwas vereinfachte Dominovariante enthält nur die Steine mit Werten von 0 bis 3. Ist es jetzt möglich, eine Kette aus allen Steinen zu bilden?
3. Die Werte auf den Steinen sollen von 0 bis k gehen, wobei k ungerade ist. Wenn nun die Steine $0/1, 2/3, \dots, k-1/k$ aus dem Spiel entnommen werden, dann existiert eine Dominokette bestehend aus allen Reststeinen. Begründen Sie, wieso dies stimmt.

Aufgabe 4.6. Wie viele einfache eulersche Graphen mit 3, 4 bzw. 5 Knoten gibt es?

Aufgabe 4.7. Beweisen oder widerlegen Sie: Sei G eulersch und seien g und f zwei Kanten in G , die einen gemeinsamen Knoten haben. Dann gibt es eine Euler-Tour, in der g und f direkt nacheinander durchlaufen werden.

Aufgabe 4.8. Beweisen oder widerlegen Sie:

1. Ein einfacher eulerscher bipartiter Graph hat immer eine gerade Anzahl an Kanten.
2. Ein einfacher eulerscher Graph mit einer geraden Anzahl an Knoten hat immer eine gerade Anzahl an Kanten.

Lösungen zu den Fragen im Text

- 13 Auf einem Weg kann nur an den Endknoten v_0 und v_k die Anzahl der inzidenten Kanten ungerade sein. Liegt ein Knoten auf dem Weg (z.B. v_i), so gibt es immer eine Kante, die in den Knoten führt (hier e_{i-1}), und eine, die aus ihm herausführt (hier e_i).

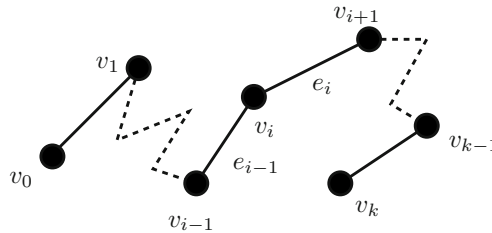


Abb. 4.17: Nur v_0 und v_k haben einen ungeraden Grad.

- 14 Da der Graph zusammenhängend ist, existiert ein solcher Weg.
- 15 Wir wollen überprüfen, ob die Kante $e = (u, v)$ eine Brücke im Graphen G' ist. Sie ist keine Brücke, wenn sie auf einem Kreis liegt. Ein Kreis in G' , der die Kante e enthält, entspricht einem (u, v) -Weg in $G' - e$. Falls wir also in $G' - e$ einen (u, v) -Weg finden, zum Beispiel mit der Tiefen- oder Breitensuche, ist die Kante e keine Brücke und kann als Nächstes gewählt werden. Gibt es hingegen keinen (u, v) -Weg in $G' - e$, dann ist e eine Brücke.
- 16 Wenn der Graph einen Euler-Weg enthält, dann können höchstens die beiden Endknoten ungeraden Grad haben. Dieses Argument haben wir schon einige Male benutzt. Andersherum gilt: Wir können eine extra Kante zwischen den beiden Knoten ungeraden Grades einfügen. Damit ist der Graph eulersch und es existiert eine Euler-Tour. Aus dieser Euler-Tour können wir nun einfach die extra Kante wieder löschen und erhalten einen Euler-Weg.
- 17 Nein. Nehmen wir an, der Graph besteht aus der disjunkten Vereinigung von drei C_4 -Kreisen und einer Kante (Abb. 4.18). Würde der Satz gelten, dann könnte man den so entstandenen Graphen mit einem Weg komplett durchlaufen. Da wir jedoch vier Zusammenhangskomponenten haben, sind dazu mindestens vier Wege notwendig. Wenn der Graph nicht zusammenhängend ist, müssen wir jede einzelne Komponente so gut wie möglich zeichnen. Die Summe ergibt dann eine minimale Zerlegung in Wege.

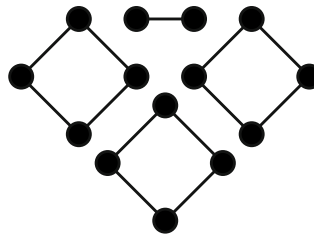


Abb. 4.18: Um alle Kanten zu besuchen, sind mindestens vier Wege notwendig, obwohl der Graph nur zwei Knoten ungeraden Grades hat.

5 Noch zwei Rundreise-Probleme

Übersicht

| | | |
|-----|---|----|
| 5.1 | Hamilton und das Icosian-Spiel | 71 |
| 5.2 | Das <i>Travelling-Salesman</i> -Problem | 78 |
| 5.3 | Komplexitätstheorie | 84 |
| 5.4 | Aufgaben | 86 |

5.1 Hamilton und das Icosian-Spiel

Im Jahr 1859 verkaufte Sir William Rowan Hamilton die Rechte für das „Icosian-Spiel“ bzw. das Spiel „Eine Reise um die Welt“ für damals 25 Pfund an den Spielehersteller John Jaques. Das Spiel ist schnell erklärt: Auf einem Spielplan sind 20 Städte (Brüssel, Canton, Delhi, ..., Xerez und Zanzibar) angeordnet (Abb. 5.1), die über ein Straßennetz miteinander verbunden sind. Ziel des Spiels ist es, die Zahlen 1 bis 20 den Städten so zuzuordnen, dass eine Rundreise entsteht. Können Sie eine solche Rundreise finden?

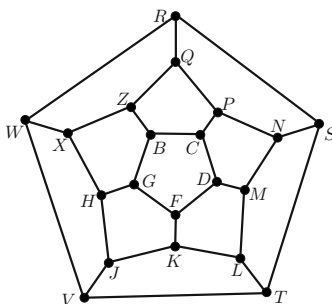


Abb. 5.1: Das Spielbrett des Icosian-Spieles.

Das Spiel kann sogar noch etwas erweitert werden, indem der erste Spieler die ersten fünf Städte der Tour bestimmen darf und der zweite Spieler diese nun beenden muss. Die Originalanleitung liest sich wie folgt:

„In this new Game (invented by Sir WILLIAM ROWAN HAMILTON, LL.D., & c., of Dublin, and by him named Icosian, from a Greek word signifying „twenty“) a player is to place the whole or part of a set of twenty numbered pieces or men upon the points or in the holes of a board, represented by the diagramm above (Abb. 5.1) drawn in such a manner as always to proceed along the lines of the figure, and also to fulfil certain other conditions, which may in various ways be assigned by another player. Ingenuity and skill may thus be exercised in proposing as well as in resolving problems of the game. For example, the first of the two players may place the first five pieces in any five consecutive holes, and then require the second player to place the remaining fifteen men consecutively in such a manner that the succession may be cyclical, that is, so that No. 20 may be adjacent to No. 1; and it is always possible to answer any question of this kind. Thus, if B C D F G be the five given initial points, it is allowed to complete the succession by following the alphabetical order of the twenty consonants, as suggested by the diagramm itself; but after placing the piece No. 6 in the hole H, as before, it is also allowed (by the supposed conditions) to put No. 7 in X insted of J, and then to conclude with the succession, W R S T V J K L M N P Q Z. Other Examples of Icosian Problems, with solutions of some of them, will be found in the following page.“

Das Spiel konnte sich allerdings nie durchsetzen und wurde zum Ladenhüter. Trotz dieses Misserfolgs werden zu Ehren des Erfinders Kreise, die jeden Knoten genau einmal durchlaufen, als Hamilton-Kreise bezeichnet.

Wie Sie sich sicher schon denken können, lässt sich das Icosian-Spiel auch in der Sprache der Graphentheorie formulieren. Gegeben ist dann ein Graph (im Spiel der Spielplan mit den Städten und Straßen) und gesucht ist ein Kreis, der jeden Knoten genau einmal durchläuft.

Definition. Ein Graph heißt **hamiltonsch** oder **Hamilton-Graph**, wenn in ihm ein Kreis existiert, der jeden Knoten genau einmal enthält. Ein solcher Kreis heißt auch **Hamilton-Kreis**.

Wir hatten im letzten Kapitel schon Euler-Kreise kennen gelernt. Dabei sollte jede Kante genau einmal im Kreis vorkommen. Bei Hamilton-Kreisen wollen wir stattdessen jeden Knoten nur einmal besuchen.

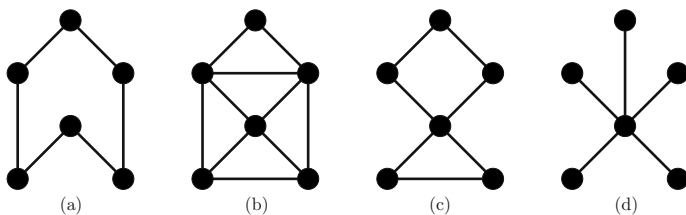


Abb. 5.2: Der Graph in (a) ist eulersch und hamiltonsch; der in (b) nur hamiltonsch und der in (c) nur eulersch. Der Baum in (d) enthält keinen Kreis.

Graphen können beide Eigenschaften haben (d.h. eulersch und hamiltonsch sein), nur eine der beiden oder sogar keine der beiden Eigenschaften erfüllen. Beispiele dafür lassen sich leicht konstruieren (Abb. 5.2). Gibt es für alle vier Fälle auch Beispiele mit drei oder vier Knoten?



Für Euler-Graphen hatten wir eine einfache Charakterisierung gefunden: Der Grad eines jeden Knotens muss gerade sein. Bei hamiltonschen Graphen ist dies leider nicht so einfach. Wir wollen uns als Erstes einige Bedingungen überlegen, die jeder hamiltonsche Graph erfüllen muss.

Lemma 5.1. *Falls ein Graph hamiltonsch ist, dann ist er auch zusammenhängend.*

Allerdings ist nicht jeder zusammenhängende Graph automatisch hamiltonsch. Ein gutes Gegenbeispiel dazu liefern Bäume. Denn ein Graph ist ja genau dann ein Baum, wenn er kreisfrei und zusammenhängend ist. Demnach kann er insbesondere keinen Hamilton-Kreis enthalten.

Lemma 5.2. *Ein hamiltonscher Graph enthält keinen Knoten mit Grad 1.*

Ein Kreis muss in jedem Knoten eine Kante zum „Reingehen“ und eine zum „Rausgehen“ haben. Wenn ein Knoten nur den Grad 1 hat, bleibt die Tour dort „stecken“. Die andere Richtung gilt aber wieder nicht. Dazu kann man sich einfach zwei an einem Knoten v „zusammengeklebte“ Kreise vorstellen (Abb. 5.3). Kein Knoten hat einen Grad von 1. Betrachten wir die beiden Teilmengen $V_1 = \{a, b, c\}$ und $V_2 = \{d, e, f\}$, dann erreicht man die Menge V_1 von der Menge V_2 aus nur über den Knoten v . Ein Kreis, der Knoten aus beiden Mengen beinhaltet, muss mindestens einmal von der Menge V_1 in die Menge V_2 und einmal von der Menge V_2 in die Menge V_1 gehen. Damit enthält jeder Kreis, der alle Knoten besucht, den Knoten v mindestens zweimal. Das widerspricht der Definition eines Hamilton-Kreises.

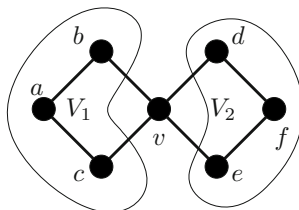


Abb. 5.3: Die beiden C_4 -Kreise sind im Knoten v „zusammengeklebt“.

Allgemein kann man sich auch Folgendes überlegen: Nehmen wir an, dass nach dem Löschen von mehreren Knoten zwei Zusammenhangskomponenten Z_1 und Z_2 entstehen. Wie viele Knoten müssen das dann mindestens gewesen sein, falls der Graph hamiltonsch war? Die Antwort ist klar: Es werden mindestens zwei „ver-

bindende“ Knoten benötigt: Zum einen müssen wir von Z_1 über einen gelöschten Knoten nach Z_2 kommen; und zum anderen über einen anderen gelöschten Knoten wieder von Z_2 zu Z_1 (Abb. 5.4).

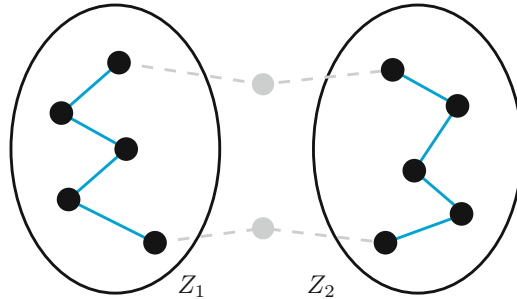


Abb. 5.4: Nach dem Löschen von einigen Knoten sind zwei Zusammenhangskomponenten entstanden. Die blauen Kanten sollen die Reste des Hamilton-Kreises in den jeweiligen Komponenten darstellen. Um die beiden Teile wieder zusammenzuführen, müssen mindestens zwei Knoten wieder in den Graphen eingefügt werden.

Was würde passieren, wenn durch das Löschen drei Zusammenhangskomponenten entstanden wären? Wie viele Knoten wären dann gelöscht worden? Diese Fragen kann man immer weiterführen und kommt so zu folgendem Satz.

Satz 5.3. *Für jeden hamiltonschen Graphen gilt: Wenn k Knoten aus dem Graphen gelöscht werden, zerfällt der Graph in höchstens k Zusammenhangskomponenten.*

Der Graph aus 5.5 erfüllt die obige Bedingung nicht. Löscht man die Knoten a und b , so entstehen drei Zusammenhangskomponenten. Er ist somit nicht hamiltonsch.

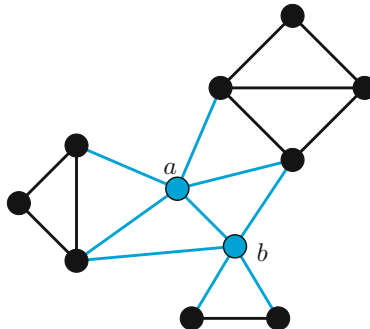


Abb. 5.5: Der Graph kann nicht hamiltonsch sein.

Beweis von Satz 5.3. Betrachten wir einen Hamilton-Kreis $C = v_1v_2v_3 \dots v_nv_1$ mit den Knoten v_1, \dots, v_n in einem Graphen G . Wir zeichnen als Erstes alle Knoten in einem Kreis entsprechend der Reihenfolge des Hamilton-Kreises C (Abb.

5.6 (a) nach (b)). Dabei lassen wir vorerst alle Kanten aus G mit Ausnahme der Kreiskanten weg.

Als Nächstes löschen wir k Knoten aus der Darstellung (Abb. 5.6 (c)). Warum können dabei nicht mehr als k Zusammenhangskomponenten entstehen? Jetzt fehlen nur noch die restlichen Kanten von G , um eine Zeichnung von G ohne die gelöschten k Knoten zu erhalten. Durch das Hinzufügen dieser Kanten werden möglicherweise einige Zusammenhangskomponenten wieder verbunden, aber es entstehen keinesfalls mehr Komponenten als zuvor (Abb. 5.6 (d)). Damit haben wir die Aussage des Satzes bewiesen. \square

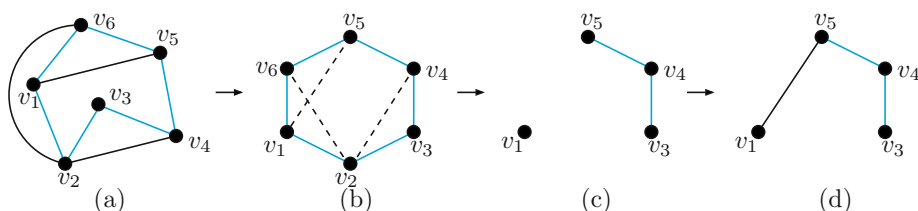


Abb. 5.6: Für jeden hamiltonschen Graphen gibt es eine Darstellung, in der alle Knoten auf einem Kreis liegen und nur Kanten im Inneren verlaufen.

Leider reicht auch diese Bedingung nicht aus, um die Existenz eines Hamilton-Kreises zu garantieren, wie beispielsweise der Petersen-Graph zeigt. Durch das Löschen von k Knoten entstehen zwar niemals mehr als k Zusammenhangskomponenten, trotzdem kann man in diesem Graphen keinen Hamilton-Kreis finden (Abb. 5.7).¹⁸

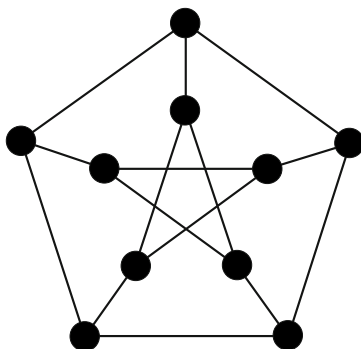


Abb. 5.7: Der Petersen-Graph erfüllt die Bedingung aus Satz 5.3, ist jedoch nicht hamiltonsch.

Viele Mathematiker haben sich um weitere Charakteristika von hamiltonschen Graphen bemüht. Einen Anhaltspunkt für die Existenz eines Hamilton-Kreises liefert die Anzahl der Kanten. Viele Kanten alleine reichen jedoch noch nicht aus. Ein Graph mit n Knoten, in dem $n-1$ Knoten einen vollständigen Graphen bilden und der letzte Knoten isoliert ist, hat bestimmt viele Kanten (wie viele?). Aber es kann natürlich keinen Hamilton-Kreis darin geben.



Um diese Situation zu vermeiden, müssen wir für eine gute Verteilung der Kanten sorgen. Ein Maß dafür liefert der Minimalgrad eines Graphen. Und tatsächlich hat im Jahr 1952 Dirac ein Kriterium abhängig vom Minimalgrad eines Graphen entdeckt, das besagt, in welchen Fällen ein Graph definitiv einen Hamilton-Kreis enthält.

Satz 5.4 (Dirac 1952). *Sei G ein einfacher Graph mit mindestens drei Knoten, für den außerdem $\delta(G) \geq 0,5 \cdot n$ gilt. Dann enthält der Graph einen Hamilton-Kreis.*

Die Forderung, dass der Graph mindestens drei Knoten enthalten muss, ist notwendig. Der K_2 (eine einfache Kante) erfüllt nämlich die obige Bedingung an den Minimalgrad, enthält allerdings keinen Kreis.

Beweis Satz von Dirac. Wir beweisen den Satz von Dirac über eine Widerspruchsnahme: Es existiere ein Graph G , der die zwei Eigenschaften erfüllt:

1. $\delta(G) \geq 0,5 \cdot n$
2. G ist nicht hamiltonsch.

Nun erweitern wir den Graphen G so lange um Kanten, bis das Hinzufügen einer jeden weiteren Kante (u, v) dafür sorgt, dass der neue Graph hamiltonsch ist. Man spricht auch von einem „größten Verbrecher“, der die gewünschte Aussage nicht erfüllt (Abb. 5.8). Dieser sei in unserem Fall wieder G .

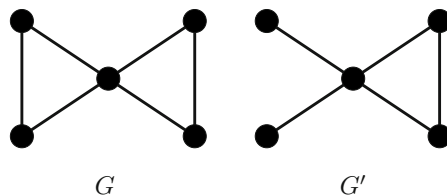


Abb. 5.8: Jede neue Kante in G führt zu einem Hamilton-Kreis. In G' ist das nicht der Fall.

Seien u und v zwei nicht benachbarte Knoten aus G . Der Graph $G' = G + (u, v)$ enthält einen hamiltonschen Kreis, da G ja ein größter Verbrecher ist. Dann muss dieser Kreis C die Kante (u, v) enthalten, da sonst in G ein Hamilton-Kreis existiert. Der Kreis C besucht jeden Knoten genau einmal und verbindet die beiden Knoten u und v direkt miteinander. Folglich gibt es also auch einen Weg v_1, \dots, v_n von $u = v_1$ nach $v = v_n$, der alle Knoten genau einmal enthält (der Kreis C ohne die Kante (u, v)). Aus dieser Tatsache werden wir nun folgern, dass in G doch ein Hamilton-Kreis vorhanden ist. Dazu brauchen wir folgende Überlegung: Angenommen, es folgt ein Nachbar v_{i+1} von u direkt auf einen Nachbarn v_i von v . Dann ist der Weg $u, v_{i+1}, v_{i+2}, \dots, v, v_i, \dots, v_2, u$ ein Hamilton-Kreis und wir erhalten einen Widerspruch (Abb. 5.9).

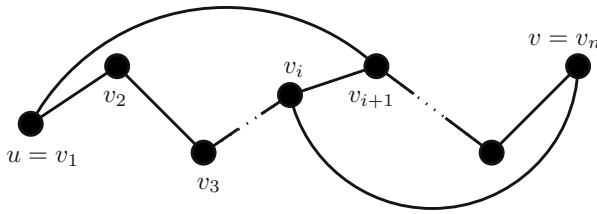


Abb. 5.9: Der Weg v_1, \dots, v_n lässt sich in einen Hamilton-Kreis umformen.

Wir müssen also nur noch zeigen, dass ein solches Knotenpaar $\{v_i, v_{i+1}\}$ existiert. Dazu werden wir nach einem Index i suchen, sodass das Knotenpaar $\{v_i, v_{i+1}\}$ die gewünschten Bedingungen erfüllt. Sei $S = \{i \mid u \text{ ist benachbart zu } v_{i+1}\}$ und $T = \{i \mid v \text{ ist benachbart zu } v_i\}$ (Abb. 5.10). Da $\delta(G) \geq 0,5 \cdot n$, gilt $|S| = d(u) \geq 0,5n$ und auch $|T| = d(v) \geq 0,5n$ (beide Knoten haben mehr als $0,5 \cdot n$ Nachbarn).

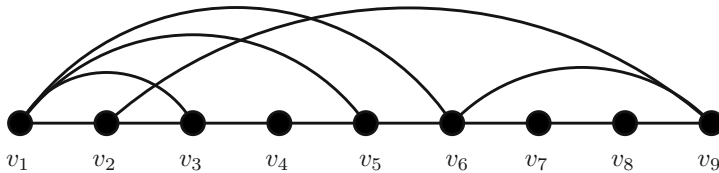


Abb. 5.10: Die Menge S besteht aus den Indizes $\{2, 4, 5\}$ und die Menge $T = \{2, 6\}$.

In $S \cap T$ sind alle gesuchten Indizes zu finden. Wir wollen $S \cap T \neq \emptyset$ zeigen. Aus der Mengenlehre wissen wir, dass $|A| + |B| = |A \cup B| + |A \cap B|$ gilt (Abb. 5.11).

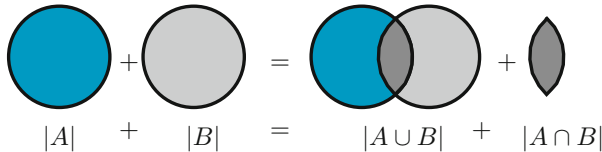


Abb. 5.11: Wenn wir die Menge $A \cup B$ betrachten, sind alle Elemente aus dem Schnitt nur einmal vorhanden. In den Mengen A und B kommen sie jedoch doppelt vor.

Wendet man die Formel umgedreht auf S und T an, so erhalten wir

$$|S \cup T| + |S \cap T| = |S| + |T| = d(u) + d(v) \geq n.$$

Der Index n , d.h. der Index, der zum Knoten v gehört, ist weder in T (da wir nur einfache Graphen betrachten) noch in S (da u nicht benachbart mit v ist) enthalten. Damit gilt aber $|S \cup T| < n$ und somit $|S \cap T| \geq 1$, sodass der von uns gesuchte Index existiert. Mit anderen Worten: Wir haben bewiesen, dass es keinen einfachen Graphen mit Minimalgrad $\delta(G) \geq 0,5 \cdot n$ geben kann, der nicht hamiltonsch ist. \square

Mit dem Satz können wir also für bestimmte Graphen sofort sagen, ob sie einen Hamilton-Kreis enthalten. Aber nicht jeder hamiltonsche Graph muss einen großen



Minimalgrad haben. Bei welchen Graphen ist das nicht der Fall? Ein allgemeines, einfaches Kriterium, an dem ein hamiltonscher Graph erkannt werden kann, gibt es leider noch nicht.

5.2 Das *Travelling-Salesman-Problem*

Eines der vielleicht bekanntesten Optimierungsprobleme ist das Travelling-Salesman-Problem, das auch als Handlungsreisenden-Problem bezeichnet wird. Die Herausforderung lässt sich folgendermaßen beschreiben: Ein Kaufmann möchte n unterschiedliche Städte bereisen und dabei an jedem Ort mindestens einmal Station machen. Welches ist die kürzeste Tour, die dies erfüllt?

Definition (*Travelling-Salesman-Problem*, TSP). Gegeben sei ein vollständiger Graph K_n mit positiven Gewichten $c(e)$ auf den einzelnen Kanten e . Gesucht ist ein Hamilton-Kreis C in K_n mit minimalem Gewicht $c(C)$, wobei gilt

$$c(C) = \sum_{e \in C} c(e).$$

Zur Erinnerung: Ein Graph heißt **vollständig**, wenn zwischen je zwei Knoten immer eine Kante existiert. Das *Travelling-Salesman-Problem* tritt nicht nur beim Erstellen von Reiserouten auf, sondern auch zum Beispiel bei der Fertigung von Leiterplatten in der Elektrotechnik.

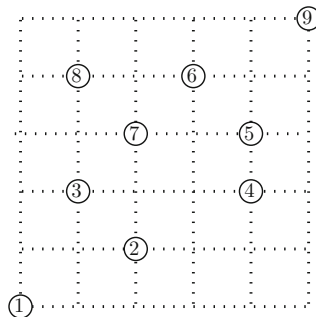


Abb. 5.12: Hier sieht man ein einfaches Bohrproblem mit 9 Löchern.

Bei der Produktion von Leiterplatten werden unterschiedliche Bauteile auf der Platte festgesteckt. Das Bohren eines Lochs erfordert immer gleich viel Zeit. Die einzige Möglichkeit zur Beschleunigung dieses Produktionsschritts besteht deshalb darin, den Bohrer während des Bohrvorgangs und beim Zurücksetzen zur Startposition entlang eines möglichst kurzen Wegs laufen zu lassen. Was hat dies jetzt mit dem *Travelling-Salesman-Problem* zu tun?

Das Problem kann man graphentheoretisch so modellieren, dass die zu bohrenden Löcher als Knoten eines vollständigen Graphen angesehen werden. Die Kanten

werden nun mit der tatsächlichen Entfernung der Löcher gewichtet. Üblicherweise erreicht man die Richtungssteuerung von Bohrern in der Leiterplattenindustrie durch zwei Motoren, einen für die rechts-links-Bewegung (x -Richtung) und einen für die vorne-hinten-Bewegung (y -Richtung). Damit ist zu zwei Löcherpositionen (x_1, y_1) und (x_2, y_2) gar nicht der (mit einem Metermaß messbare) **euklidische Abstand**

$$d_{\text{eukl}} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2},$$

sondern der sogenannte **Maximum-Abstand**

$$d_{\text{max}} = \max\{|x_2 - x_1|, |y_2 - y_1|\}$$

für das Problem relevant. Beispielsweise braucht der Bohrer von Position 3 zu 4 genauso lange wie für den Weg von 1 zu 7 (bei letzterem ist die y -Richtung die maßgebliche; bis der y -Motor die drei Längeneinheiten abgefahren hat, ist der x -Motor längst fertig). In der Abbildung 5.13 ist der entsprechende vollständige Graph mit den Kantengewichten für die Knoten 1 und 6 dargestellt.

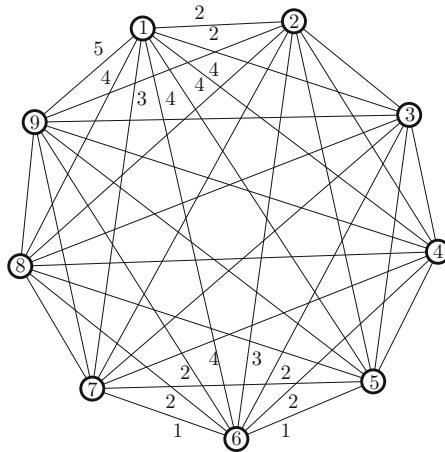


Abb. 5.13: Die Kantengewichte entsprechen dem Maximum-Abstand und nicht dem euklidischen.

Gesucht ist nun eine Tour für den Bohrer, die möglichst kurz ist und an jedem Loch mindestens einmal vorbeikommt. Graphentheoretisch gesprochen suchen wir einen Hamilton-Kreis mit dem kleinsten Gewicht. Das Gewicht eines Kreises besteht dabei aus der Summe seiner Kantengewichte. Wie könnte eine möglichst gute Rundreise auf der Leiterplatte in der Abbildung 5.12 aussehen?

Es ist also durchaus wichtig für die Praxis, Algorithmen zu entwickeln, die das *Travelling-Salesman-Problem* lösen. Als ersten algorithmischen Ansatz kann man alle Hamilton-Kreise des gegebenen vollständigen Graphen berechnen und aus diesen den billigsten aussuchen. Allerdings gibt es zu viele Hamilton-Kreise in einem vollständigen Graphen: Starten wir von einem beliebigen Knoten v_0 . Dann



kann die Tour als Nächstes einen der $n - 1$ anderen Knoten enthalten. Setzen wir diesen Knoten mit $v_1 \neq v_0$ fest, so bleiben uns für die Wahl des darauf folgenden Knotens noch $n - 2$ Möglichkeiten. Führen wir diese Überlegung fort, erhalten wir demnach $(n - 1)!$ Kreise. Jetzt müssen wir noch die Anzahl der Kreise durch zwei teilen, da wir in unserer Zählweise jeden Kreis in beide Richtungen durchlaufen.

Satz 5.5. *Ein vollständiger Graph enthält genau $0,5 \cdot (n - 1)!$ Hamilton-Kreise.*

Da $0,5 \cdot (n - 1)!$ sehr schnell wächst, ist das Verfahren, alle Kreise mit ihren Gewichten zu berechnen, in der Praxis ungeeignet. Es werden andere Algorithmen benötigt, die vielleicht nicht die optimale, aber doch wenigstens eine ziemlich gute Tour liefern. Solche Algorithmen nennt man auch *Heuristiken*. Eine erste Möglichkeit ist, möglichst gierig (engl. *greedy*) vorzugehen, d.h. für die Tour möglichst billige Kanten zu wählen. Die Vorgehensweise führt zu folgendem *Greedy-Algorithmus* (Alg. 5.1):

Algorithmus 5.1 *Greedy-TSP-Algorithmus*

Input: Ein vollständiger gewichteter Graph.

Output: Ein Hamilton-Kreis.

Schritt 1: Wählen Sie eine noch nicht gestrichene und noch nicht markierte Kante e minimalen Gewichts.

Schritt 2: Falls sich die bisher markierten Kanten zusammen mit e zu einer Tour ergänzen lassen, so markieren Sie auch e . Ansonsten streichen Sie e .

Schritt 3: Falls die markierten Kanten einen Hamilton-Kreis bilden: STOPP. Ansonsten: Gehen Sie zu Schritt 1.

Welche Tour berechnet der *Greedy-TSP-Algorithmus* in dem Graphen aus der Abbildung 5.14?

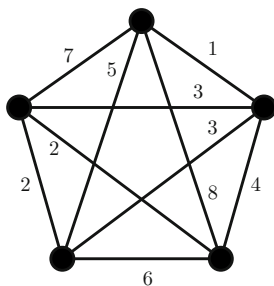


Abb. 5.14: Der K_5 mit Kantengewichten.

Wie haben Sie in Schritt 2 entschieden, ob sich die Kanten zu einer Tour ergänzen lassen? Was ist, wenn in Schritt 1 zwei Kanten zur Auswahl stehen? Kann die Wahl einer der Kanten zu einer starken Kostenveränderung bei der Lösung führen?¹⁹ Bei dem Graphen der Abbildung 5.15 ergibt der *Greedy-TSP-Algorithmus* eine Tour mit einem Gesamtgewicht 11, während die optimale Tour die Länge 9 hat. Der Unterschied von zwei hört sich gar nicht so schlecht an. Auch wenn wir uns das Verhältnis „Gewicht der ausgewählten Tour : Gewicht der optimalen Tour“ anschauen (hier $\frac{11}{9}$), liegt dieses nahe bei eins, dem optimalen Wert. Ist das aber immer so oder kann es auch passieren, dass dieses Verhältnis beliebig groß werden kann?

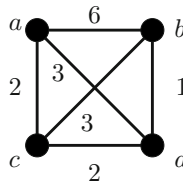


Abb. 5.15: Der *Greedy-TSP-Algorithmus* berechnet eine Tour der Länge 11. Wie sieht sie aus? Welcher Hamilton-Kreis ist eine optimale Tour?

Ein geschickterer Algorithmus als der *Greedy-Algorithmus* ist der Doppelte-Baum-Algorithmus von Rosenkrantz, Stearns und Lewis aus dem Jahr 1977, vorausgesetzt die Kantengewichte erfüllen die Dreiecksungleichung. Die [Dreiecksungleichung](#) garantiert im vollständigen Graphen, dass für alle Knoten u, v und w gilt:

$$c(u, v) \leq c(u, w) + c(w, v).$$

Man könnte auch sagen, der direkte Weg von u nach v ist auf jeden Fall kürzer als der Umweg über einen weiteren Knoten w (Abb. 5.16).

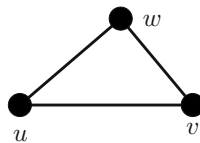


Abb. 5.16: In jedem Dreieck ist die Kante zwischen u und v kürzer als die Strecke von u nach v über den Knoten w .

Für den euklidischen Abstand ist die Bedingung erfüllt. Wie sieht es für den Maximum-Abstand aus? Kann eine Kantengewichtung, die die Dreiecksungleichung erfüllt, auch negative Werte annehmen?²⁰



Der Doppelte-Baum-Algorithmus funktioniert nun folgendermaßen (Alg. 5.2):

Algorithmus 5.2 Doppelter-Baum-Algorithmus

Input: Ein vollständiger Graph K_n mit Kantengewichten $c(e)$, die die Dreiecksungleichung erfüllen.

Output: Ein Hamilton-Kreis.

Schritt 1: Konstruieren Sie einen minimal spannenden Baum T von K_n .

Schritt 2: Verdoppeln Sie alle Kanten von T (damit erhalten wir einen eulerschen Graphen T_d).

Schritt 3: Berechnen Sie eine Euler-Tour in T_d .

Schritt 4: Durchlaufen Sie die Euler-Tour vom Knoten 1 aus. Falls ein Knoten schon besucht wurde, nehmen Sie die Abkürzung zum nächsten unbesuchten Knoten auf der Tour.

Der Algorithmus benutzt alle Techniken, die wir bisher kennen gelernt haben, um eine gute Tour zu berechnen. Gehen wir den Algorithmus nun an einem Beispiel durch.

Beispiel. Wir nehmen an, dass die Punkte in der Ebene die Knoten unseres Graphen sind (Abb. 5.2(a)). Der euklidische Abstand zwischen zwei Knoten soll dieses Mal auch die Gewichtung der Kanten bestimmen. Damit das Beispiel übersichtlich bleibt, lassen wir das Gewicht und auch die Kanten zunächst weg.

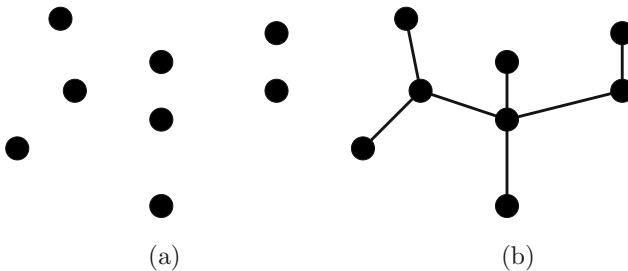


Abb. 5.17: Die Entfernung zwischen zwei Knoten wird durch den euklidischen Abstand definiert. In Teil (b) ist dann ein minimal spannender Baum abgebildet.

Als Erstes berechnen wir einen minimal spannenden Baum T (Abb. 5.17 (b)). Welche Algorithmen können wir hierfür verwenden? Wie können Sie überprüfen, ob der gegebene Baum optimal ist? Als Nächstes gibt uns der Algorithmus vor, die einzelnen Kanten zu verdoppeln. In diesem neuen Graphen T_d bestimmen wir nun eine Euler-Tour (Abb. 5.18 (c)). Wieso existiert in jedem Fall eine solche Tour? Können nicht auch Knoten ungeraden Grades in T_d auftauchen?



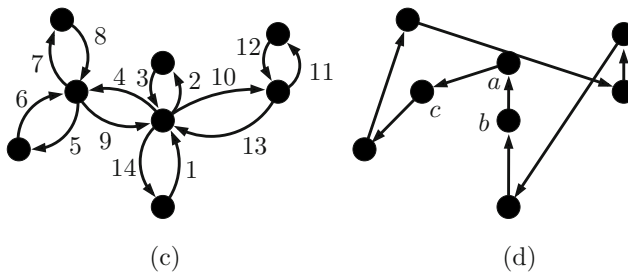


Abb. 5.18: Die Zahlen an den Kanten geben die Euler-Tour in T_d wieder (Teil (c)). In Teil (d) ist dann die berechnete Tour dargestellt. Ist sie optimal?

Als letzten Schritt bestimmen wir noch den Weg entlang der Euler-Tour und nutzen, falls möglich, Abkürzungen. Hier in diesem Fall können wir statt des Rückwegs von Knoten a zu b direkt zu c weitergehen (Abb. 5.18(d)). ■

Natürlich stellt sich auch hier die Frage nach der Genauigkeit des Algorithmus. Dieser liefert in unserem Beispiel zwar keine optimale Lösung, aber er berechnet immer eine, die maximal doppelt so lang wie die optimale Tour ist.

Satz 5.6. Sei K_n ein vollständiger Graph mit positiven Kantengewichten $c(e)$ für jede Kante e , wobei die Gewichte die Dreiecksungleichung erfüllen. Sei T' die Lösung des Doppelten-Baum-Algorithmus und OPT eine optimale Lösung. Dann gilt

$$c(T') \leq 2 \cdot c(OPT).$$

Beweis: Die Tour T' kann höchstens doppelt so lang sein, wie der Wert des minimal spannenden Baums T , der im Algorithmus berechnet wurde, d.h.

$$c(T') \leq 2 \cdot c(T).$$

Damit haben wir eine obere Schranke an die Kosten von T' . Als Nächstes suchen wir noch eine untere Schranke an die Kosten von OPT . Dabei hilft die folgende Beobachtung: Löschen wir eine Kante aus der optimalen Tour, erhalten wir einen spannenden Baum. Dieser Baum ist nicht billiger als der minimal spannende Baum T . Also gilt $c(T) \leq c(OPT)$. Daraus ergibt sich dann

$$c(T') \leq 2 \cdot c(T) \leq 2 \cdot c(OPT).$$

□

Damit berechnet der Doppelte-Baum-Algorithmus immer eine Tour mit dem Verhältnis $c(T')/OPT \leq 2$. Das entspricht noch nicht ganz unserem Ziel, eine optimale Lösung zu bestimmen. Tatsächlich ist zurzeit kein Algorithmus bekannt, der eine optimale Tour in annehmbarer Zeit berechnet. Was „annehmbare“ Zeit bedeutet und welche weiteren Auswirkungen das Problem hat, werden Sie im Folgenden Abschnitt sehen.

5.3 Komplexitätstheorie

Im Bereich der kombinatorischen Optimierung beschäftigt man sich hauptsächlich mit zwei Problemklassen: den Entscheidungsproblemen und den Optimierungsproblemen. Ein **Entscheidungsproblem** ist ein Problem, bei dem die Antwort entweder „ja“ oder „nein“ lautet. Ein Beispiel hierfür ist die Frage: Existiert in dem gegebenen Graphen ein Hamilton-Kreis? Bei einem **Optimierungsproblem** hingegen wird die beste Lösung unter allen korrekten Lösungen ausgesucht, wie das beim *Travelling-Salesman-Problem* der Fall ist. Wir wollen uns im Folgenden auf Entscheidungsprobleme konzentrieren. Bevor wir uns mit der Aufteilung der Entscheidungsprobleme in unterschiedliche Kategorien beschäftigen, müssen wir unser Begriffsrepertoire bei Algorithmen noch etwas erweitern.

5.3.1 Effiziente und ineffiziente Algorithmen

Ein **Algorithmus** ist eine Handlungsvorschrift zur Lösung eines Problems bzw. einer Kategorie von Problemen. In den meisten Fällen soll sich ein Algorithmus auch immer als Computerprogramm umsetzen lassen. In der Praxis wird die Laufzeit eines solchen Algorithmus (vom Start- bis zum Endpunkt) einfach per Stoppuhr gemessen. In der Theorie will man genauer untersuchen, wie lange der Algorithmus für alle unterschiedlichen Problemgrößen braucht. Dazu zerlegt man den Algorithmus in sogenannte **elementare Rechenschritte**, wie die Addition, das Schreiben auf einen Speicherplatz oder den Zugriff auf einen Speicherplatz. Die **Laufzeit** eines Algorithmus ist dann die Anzahl der benötigten elementaren Rechenschritte.

Natürlich hängt die Laufzeit immer von der betrachteten Instanz ab, in diesem Fall der Inputgröße. Soll der Hierholzer-Algorithmus zur Berechnung einer Euler-Tour auf einem Graphen mit vielen Knoten und Kanten laufen, dann dauert das länger, als wenn wir ihn auf den K_3 anwenden. Die Laufzeit eines Algorithmus wird deswegen immer in Abhängigkeit der Inputgröße angegeben. Im Fall der Graphentheorie geht man davon aus, dass die Anzahl der Knoten n und die Anzahl der Kanten m den Input des Graphen bestimmen. Eine exakte Berechnung der Laufzeit, wie „der Algorithmus benötigt $34n^2 + 5m$ Rechenoperationen“, ist meist kompliziert und nicht so interessant. Stattdessen schätzt man die Laufzeit grob ab. Gilt zum Beispiel, dass die Laufzeit $\ell_A(n, m)$ eines Algorithmus A mit der Inputgröße n, m abschätzbar ist durch

$$\ell_A(n, m) \leq an^2 + bm + c$$

mit beliebigen Konstanten a, b, c , so interessiert uns nur der höchste Exponent. Wir sprechen bei A von einer quadratischen Laufzeit wegen n^2 . Einen Algorithmus, dessen Laufzeit über ein beliebiges Polynom abschätzbar ist, bezeichnet man als Algorithmus mit **polynomieller** Laufzeit. Solche Algorithmen heißen auch **ef-**

effiziente oder gute Algorithmen. Damit haben wir ein Kriterium zum Vergleich unterschiedlicher Algorithmen gefunden.

Wieso wir effiziente Algorithmen als gut bezeichnen, sieht man an dem folgenden Beispiel. Nehmen wir an, uns ist ein nicht effizienter Algorithmus mit einer Laufzeit von $n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1 = n!$ gegeben, und wir vergleichen ihn mit Algorithmen, die eine Laufzeit von n , n^2 und n^4 haben. Dann ergibt sich folgende Anzahl elementarer Rechenschritte für die unterschiedlichen Inputgrößen:

| n | Laufzeit $n!$ | Laufzeit n | Laufzeit n^2 | Laufzeit n^4 |
|-----|--|--------------|----------------|----------------|
| 3 | 6 | 3 | 9 | 81 |
| 5 | 120 | 5 | 25 | 625 |
| 10 | 3.628.800 | 10 | 100 | 10.000 |
| 20 | $= 2.432.902.008.176.640.000$ $\approx 2,4 \cdot 10^{18}$ | 20 | 400 | 160.000 |

Tab. 5.1: Unterschiedliche Rechenoperationen für einen Input der Größe n . Im Vergleich wird das Alter des Universums auf $13,7 \cdot 10^9$ Jahre geschätzt.

Die Anzahl der Rechenoperationen explodiert geradezu. Die Laufzeit von Algorithmen hilft uns jetzt Entscheidungsprobleme zu klassifizieren.

5.3.2 Die Klasse P und die Klasse NP

Wir hatten gesagt, dass Entscheidungsprobleme mit Ja oder Nein zu beantworten sind. Zwei Entscheidungsprobleme, die wir bereits kennen gelernt haben, sind die folgenden:

- Enthält der Graph G einen Hamilton-Kreis?
- Ist der Graph G eulersch?

Zur **Klasse P** gehören nun alle Entscheidungsprobleme, für die es einen effizienten Algorithmus gibt, der eine Ja- oder Nein-Antwort liefert. Das zweite Problem liegt in dieser Klasse. Auch, ob ein Graph G bipartit ist, kann in polynomieller Zeit entschieden werden. Das hatten wir in Abschnitt 2.3.1 gezeigt. Einige weitere Probleme aus der Klasse **P** werden wir noch kennen lernen.

Wie sieht es aber mit dem Problem aus, ob G einen Hamilton-Kreis enthält oder nicht. Zu dessen Lösung kennt man bisher noch keinen polynomiellen Algorithmus. Immerhin können wir in einem gegebenen Graphen schnell überprüfen, ob eine gefundene Lösung wirklich eine ist: Ist die Lösung ein einfacher Kreis und enthält sie jeden Knoten genau einmal, ist der Graph hamiltonsch. Auf diese Weise definiert man nun die Klasse **NP**. In der **Klasse NP** sind alle Entscheidungsprobleme enthalten, für die ein schneller Beweis für die Korrektheit einer Ja-Antwort geliefert werden kann, falls die Antwort Ja lautet.

Leicht zu sehen ist, dass die Klasse \mathbf{P} in der Klasse \mathbf{NP} enthalten ist. Die wichtigste Frage in der Komplexitätstheorie ist:

Gilt $\mathbf{P} = \mathbf{NP}$?

Das „ $\mathbf{P} = \mathbf{NP}$ “-Problem ist eines der *Millenium Prize Problems*, die das Clay Mathematics Institute definiert hat (www.claymath.org/millenium). Für die Lösung des Problems winkt ein Preisgeld von einer Million Dollar. Für die Mathematik bedeutet die Lösung das Folgende:

Falls $\mathbf{P} \neq \mathbf{NP}$ gilt, werden wir Mathematiker bis ans Ende unserer Tage gebraucht. Für viele Probleme aus der Praxis ist nämlich noch kein polynomieller Algorithmus bekannt. Gleichzeitig drängen die Praktiker aber immer mehr darauf, ihre Entscheidungen aufgrund von mathematischen Lösungen zu treffen.

Falls $\mathbf{P} = \mathbf{NP}$ gilt, sind die Mathematiker jedoch auch nicht schlagartig beschäftigungslos. Es ist keinesfalls sicher, dass sofort effiziente Algorithmen für alle Probleme aus \mathbf{NP} bekannt sind. Das hängt davon ab, ob der Beweis konstruktiv ist, d.h. ein Werkzeug für polynomielle Algorithmen bereitstellt, oder nicht. Wäre der Beweis konstruktiv, so hätte dies für die Kryptographie weitreichende Folgen, da damit alle Verschlüsselungssysteme leicht zu knacken wären. Auf der anderen Seite könnten viele schwierige Probleme in kurzer Zeit gelöst werden.

Wir wollen an dieser Stelle den Bereich der Komplexitätstheorie verlassen und uns anderen Problemen aus der Graphentheorie und der Kombinatorischen Optimierung widmen.

5.4 Aufgaben

Aufgabe 5.1. Diplomaten aus 6 Ländern sollen zu einer Konferenz an einem runden Tisch Platz nehmen (Abb. 5.19). Allerdings gibt es einige Konflikte unter ihnen und zwar zwischen den Ländern A und B , zwischen C und B , zwischen E und B , zwischen C und D sowie zwischen E und F . Deswegen sollen ihre Vertreter nicht direkt nebeneinander sitzen. Kann der Protokollchef die Diplomaten so platzieren, dass diese Bedingung erfüllt ist?

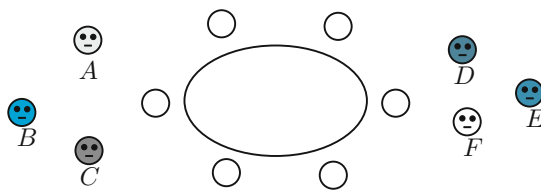


Abb. 5.19: Nicht jeder Diplomat möchte neben jedem sitzen.

Aufgabe 5.2. Von den folgenden Graphen (Abb. 5.20) sind vier hamiltonsch. Welche sind es und warum gibt es bei den anderen keinen Hamilton-Kreis?

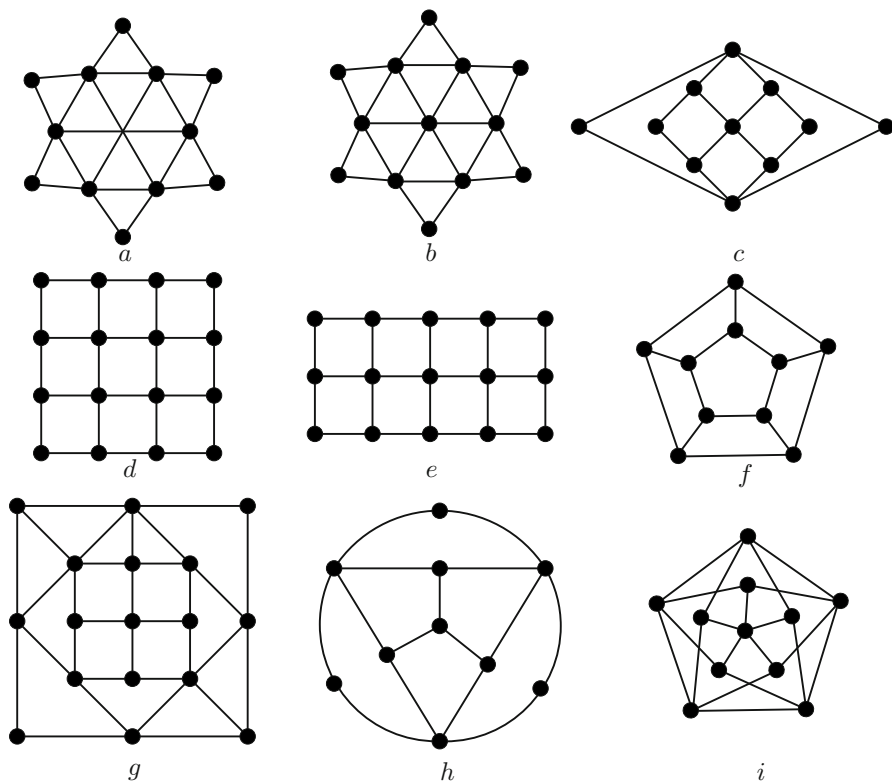


Abb. 5.20: In welchen Graphen gibt es einen Hamilton-Kreis?

Aufgabe 5.3. Eine Maus sitzt vor einem würfelförmigen Stück Käse mit den Maßen $3 \times 3 \times 3$, der wiederum aus lauter kleinen $1 \times 1 \times 1$ -Würfeln besteht (Abb. 5.21). Kann sie sich durch alle Würfel hindurchfressen und in der Mitte aufhören? Dabei durchquert sie die Würfel nur an den Grenzflächen, nicht an den Kanten. Was passiert, wenn sie an einer anderen Stelle anfängt?

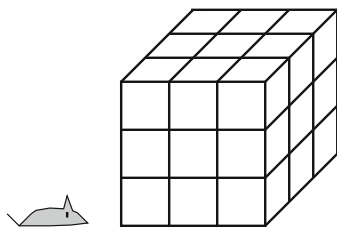


Abb. 5.21: Die Maus möchte sich durch alle Würfel essen und in der Mitte enden. Ist das möglich?

Aufgabe 5.4. In welchen Graphen ist eine eulersche Tour zugleich ein hamiltonscher Kreis?

Aufgabe 5.5. Für welche Werte a und b ist der vollständige bipartite Graph $K_{a,b}$ hamiltonsch?

Aufgabe 5.6. Kann man auf einem 4×4 -Schachbrett den Springer so ziehen, dass er auf jedes Feld einmal gesetzt wird und am Ende wieder da ankommt, wo er gestartet ist? Ein Springer darf immer zwei nach oben bzw. zwei nach rechts und dann eins nach rechts bzw. eins nach oben ziehen (Abb. 5.22).

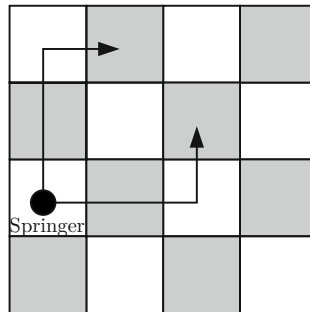


Abb. 5.22: Von dieser Position aus kann der Springer entlang der beiden eingezeichneten Pfeile ziehen.

Wie sieht es mit einem 5×5 - oder 6×6 - Schachbrett aus?

Aufgabe 5.7. Konstruieren Sie zwei einfache zusammenhängende Graphen mit derselben Knotenanzahl und Gradfolge, wobei nur einer von beiden hamiltonsch ist.

Aufgabe 5.8. Beweisen Sie: Ein einfacher zusammenhängender 6-regulärer Graph mit 11 Knoten enthält immer einen Hamilton-Kreis.

Aufgabe 5.9. Sei $G = (V, E)$ ein Graph. Sein Komplementärgraph $\overline{G} = (V, \overline{E})$ enthält alle Kanten, die nicht in E enthalten sind. Zeigen Sie, dass der Komplementärgraph \overline{C}_n zum Kreisgraphen C_n für $n \geq 5$ einen Hamilton-Kreis enthält.

Lösungen zu den Fragen im Text

18 **Satz.** Der Petersen-Graph enthält keinen Hamilton-Kreis.

Beweis: Falls im Petersen-Graph ein Hamilton-Kreis existiert, so enthält jeder Schnitt eine gerade Anzahl an Kanten des Kreises. Betrachten Sie nun die äußeren Knoten und den von ihnen induzierten Schnitt (Abb. 5.23). In diesem Fall gibt es zwei Möglichkeiten, wie viele Kanten in dem Schnitt liegen: Entweder es sind zwei oder vier Kanten.

Fall 1: Zwei Kanten liegen im Schnitt. Dann müssen die zwei Kanten zu den äußeren Knoten nebeneinander liegen. In der Abbildung 5.23 sind sie und die Tour durch die

äußeren Knoten blau markiert. Jedoch kann man keinen Weg finden, der alle Knoten im Inneren enthält und dessen Endknoten an den beiden Schnittkanten inzidiert.

Fall 2: Vier Kanten liegen im Schnitt. Liegen vier Kanten im Schnitt, so sind sie wie in der Abbildung 5.23 angeordnet. Auch hier kann die angefangene Tour zu keinem Hamilton-Kreis erweitert werden. \square

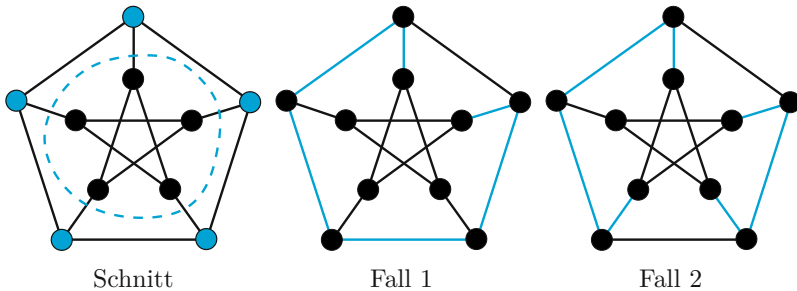


Abb. 5.23: Über jeden Schnitt führt eine gerade Anzahl an Kanten des Hamilton-Kreises.

- 19 Wir nehmen an, dass der Algorithmus mit der Kante e_1 startet. Wählt er anschließend die Kante e_2 , so berechnet der Algorithmus eine Tour mit Kosten 8. Hätte er hingegen als Nächste die Kante e_3 gewählt, so hätte sich eine Tour mit Kosten 4 ergeben. Die Länge der Tour kann also wesentlich davon abhängen, welche Kante mit gleichem Gewicht gewählt wird.

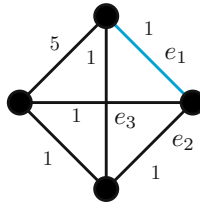


Abb. 5.24: Je nachdem ob e_2 oder e_3 gewählt wird, berechnet der Algorithmus die optimale oder eine doppelt so lange Tour.

- 20 **Satz.** Für einen vollständigen Graphen seien Kantengewichte c gegeben. Falls c die Dreiecksungleichung erfüllt, dann gilt $c(e) \geq 0$ für jede Kante $e \in E$.

Beweis: Angenommen für die Kante $e = (u, v)$ gilt $c(e) < 0$. Betrachten Sie einen beliebigen weiteren Knoten w . Weiter sei $c((v, w)) \leq c((u, v))$. Nach der Dreiecksungleichung gilt

$$c((u, w)) \leq c((v, w)) + \underbrace{c((u, v))}_{< 0} < c((v, w)).$$

Das ist aber ein Widerspruch zu $c((v, w)) \leq c((u, v))$. \square

6 Planarität

Übersicht

| | | |
|-----|--|-----|
| 6.1 | Gas-Wasser-Strom und Planarität | 91 |
| 6.2 | Outerplanare Graphen | 95 |
| 6.3 | Die Euler-Formel | 98 |
| 6.4 | Die Graphen $K_{3,3}$, K_5 und Kuratowski | 104 |
| 6.5 | Aufgaben | 108 |

6.1 Gas-Wasser-Strom und Planarität

Im Jahr 1917 veröffentlichte H. E. Dudeney in dem Buch „*Amusements in Mathematics*“ das Gas-Wasser-Strom-Problem: Drei Häuser sollen mit Gas, Wasser und Strom versorgt werden. Aus Sicherheitsgründen dürfen sich die Leitungen von den Strom-, Wasser- und Gaswerken zu den Häusern nicht kreuzen (Abb. 6.1). Ist dies möglich?

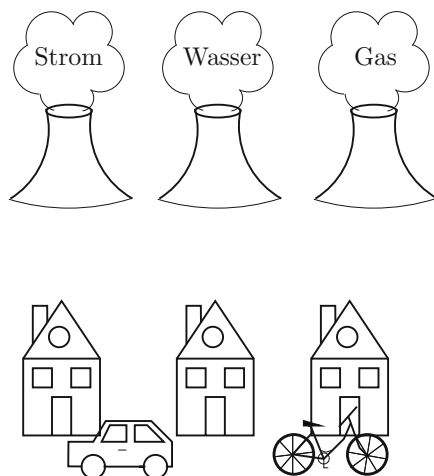


Abb. 6.1: Können die Leitungen so gelegt werden, dass kein Risiko für die Bewohner entsteht?

Dudeney charakterisiert das Problem als „[as] old as hills, ... much older than electric lighting, or even gas“, allerdings gibt er keine älteren Quellen an.

Aber nicht nur in der Bauindustrie ist man an der Verlegung kreuzungsfreier Leitungen interessiert. Genauso wichtig ist es, z.B. auf einem Chip die Leitungen zwischen elektronischen Bauteilen (Widerstände, Kondensatoren oder Transformatoren) so anzuordnen, dass diese sich nicht überschneiden und einen Kurzschluss hervorrufen.

Überträgt man die Problemstellung in die Graphentheorie, so stellt sich die Frage nach den Darstellungsmöglichkeiten eines Graphen. In diesem Zusammenhang sprechen wir von einer **Darstellung** des Graphen auf einem Blatt Papier oder allgemeiner im \mathbb{R}^2 , wenn die Knoten nicht übereinander liegen und die Kanten außer ihrem Start- und Endknoten keine weiteren Knoten durchqueren. Eine spezielle Darstellung ist nun diejenige, in der sich keine Kanten kreuzen. Besitzt ein Graph eine solche Darstellung, so bezeichnen wir ihn als planar.

Definition. Ein Graph heißt **eben** oder **planar dargestellt**, wenn er ohne Kantenkreuzungen in der Ebene, d.h. im \mathbb{R}^2 gezeichnet ist. Ein Graph heißt **planar**, wenn er eine planare Darstellung hat.

An sich gehören die Begriffe „Kantenkreuzung“, „übereinander liegen“ und „durchqueren“ in den Bereich der Topologie und müssten formal definiert werden. Für unsere Zwecke ist dies jedoch nicht wirklich notwendig. Stattdessen wollen wir uns klar machen, was der Unterschied zwischen planar und planar dargestellt ist. Anhand der zwei Zeichnungen des K_4 (Abb. 6.2) lässt sich dies leicht nachvollziehen. In der ersten Darstellung schneiden sich die Kanten in der Mitte. Trotzdem ist der Graph planar, da er eine ebene Darstellung besitzt (Abb. 6.2 (b)). Ein weiteres Beispiel bildet der Würfelgraph, den wir schon in Kapitel ?? kennen gelernt haben. Auch hier gibt es sowohl ebene als auch nicht ebene Darstellungen. Der Würfelgraph ist also ebenfalls planar.

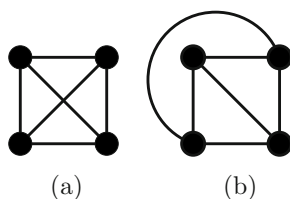


Abb. 6.2: Der K_4 ist planar, aber nicht jede Darstellung ist eben.

Bisher hatten wir uns bei einem Graphen hauptsächlich um die Knoten und Kanten gekümmert. Hat man nun planare Darstellungen, so kann man auch von Flächen reden. Grob gesprochen sind die **Flächen** eines Graphen die Teile der Ebene, die übrig bleiben, wenn die Knoten und Kanten aus der Ebene gelöscht werden (Abb. 6.3).

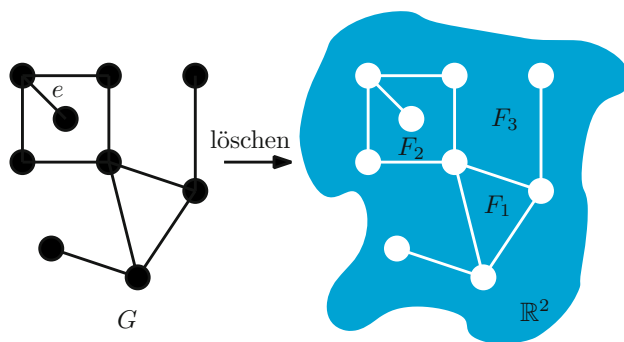


Abb. 6.3: Durch das Löschen von G entstehen die drei Flächen F_1 , F_2 und F_3 .

Die Abbildung 6.3 zeigt die drei Flächen des dargestellten Graphen. Die Fläche F_3 unterscheidet sich von den anderen darin, dass sie keine äußere Begrenzung oder Umrahmung hat. Eine solche Fläche wird auch als **äußere Fläche** bezeichnet. Jede Darstellung hat genau eine äußere Fläche. Die anderen Flächen werden **innere Flächen** genannt und sind immer durch einen Kreis im Graphen begrenzt. Besitzt jeder Graph eine innere Fläche?²¹

In zwei planaren Darstellungen eines Graphen sieht die äußere Fläche nicht immer gleich aus. Tatsächlich kann jede beliebige innere Fläche zur äußeren gemacht werden. Vielleicht hilft Ihnen dazu die folgende Vorstellung: Eine Darstellung des Graphen kann man auf einen Luftballon übertragen. Soll jetzt eine Fläche zur äußeren Fläche werden, so schneidet man ein Loch in diese Fläche, zieht die Enden nach außen und drückt alles flach auf den Boden. Hat man die Enden des Lochs weit genug auseinandergedrückt, so ist wieder eine planare Darstellung entstanden. Wie sieht eine planare Darstellung des Graphen in 6.4 aus, die die Fläche A als äußere Fläche hat?

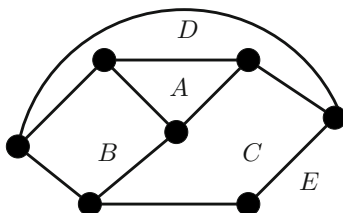


Abb. 6.4: Die äußere Fläche in dieser Darstellung ist E . Aber jede andere Fläche kann zur äußeren Fläche werden.

Während für Knoten die Anzahl inzidenter Kanten wichtig ist, wird die Fläche durch die Anzahl der sie umschließenden Kanten näher bestimmt.

Definition. Der **Grad** $d(F)$ oder die **Länge** $f(F)$ einer Fläche F ist die Anzahl der sie begrenzenden Kanten, wobei in das Gebiet ragende Kanten doppelt gezählt werden.

Nach Definition hat die Fläche F_1 aus der Abbildung 6.3 die Länge 3. Für die Fläche F_2 erhalten wir $f(F_2) = 6$, da die Kante e doppelt gezählt wird. Aber wie sieht es mit F_3 aus?

Auch für die äußere Fläche ist der Grad definiert. Kanten, die in den Raum ragen, werden dabei wieder doppelt gezählt. Der Grad von F_3 beträgt damit 11. Man kann sich die Berechnung so vorstellen, als würde man einmal um die Darstellung herumlaufen und dabei zählen, wie viele Kanten man besucht (Abb. 6.5).

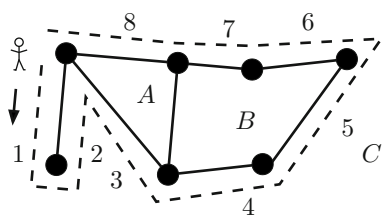


Abb. 6.5: Die Länge einer Fläche kann durch ihr „Ablaufen“ leicht bestimmt werden. Die Zahlen geben an, wie nach und nach alle Kanten gezählt werden.

Damit haben wir die wichtigsten Begriffe der Planarität eingeführt und können sie an einigen Beispielen ausprobieren. Welche der folgenden Graphen (Abb. 6.6) sind denn nun planar? Und wie viele Flächen haben sie? Welchen Grad hat die äußere Fläche?

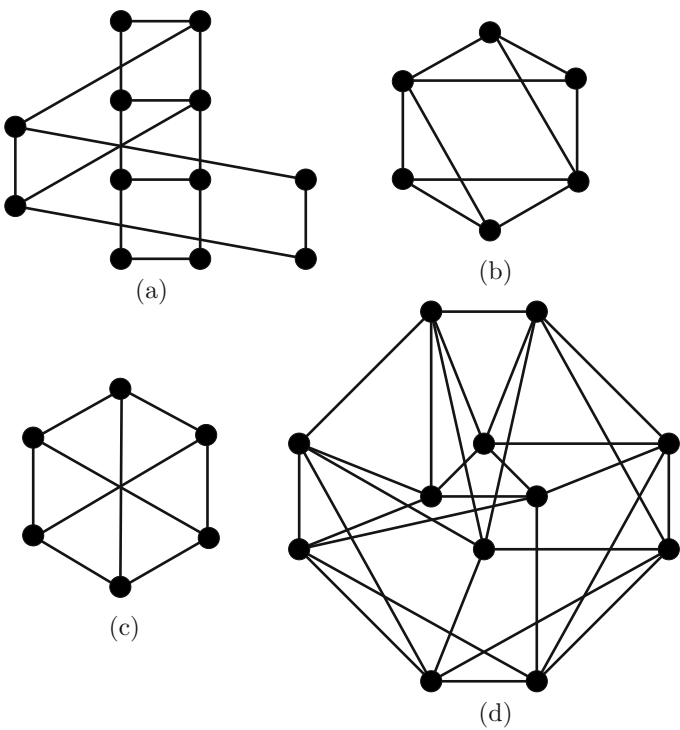


Abb. 6.6: Welche Graphen sind planar und wie sehen ihre Flächen aus?

Vielleicht fallen Ihnen auch schon Klassen von planaren Graphen ein. Mit einer speziellen Klasse beschäftigen wir uns im nächsten Abschnitt.

6.2 Outerplanare Graphen

Outerplanare Graphen bilden eine spezielle Graphenklasse innerhalb der planaren Graphen und sind wieder über eine Besonderheit ihrer Darstellung definiert.

Definition. Ein Graph ist **outerplanar**, wenn er eine planare Darstellung hat, in der alle Knoten inzident zur äußeren Fläche sind. Eine solche Darstellung bezeichnet man auch als **outerplanare Darstellung**.

Die Definition hört sich komplizierter an, als sie ist. Die Abbildung 6.7 zeigt zwei Darstellungen outerplanarer Graphen.

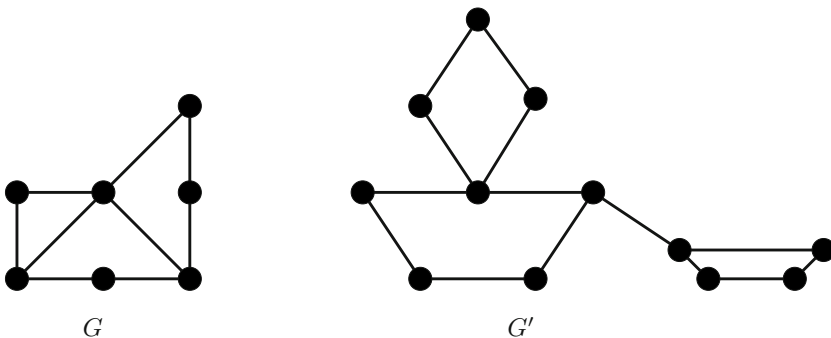


Abb. 6.7: Bei outerplanaren Graphen liegen alle Knoten an der äußeren Fläche.

Bei einer outerplanaren Darstellung ist es immer möglich, einen extra Knoten, der mit allen anderen Knoten verbunden ist, so in die Zeichnung einzufügen, dass sich auch weiterhin keine Kanten kreuzen. Wie könnte das bei den beiden Graphen in 6.7 aussehen?

Diese Beobachtung wollen wir in einer ersten Charakterisierung von outerplanaren Graphen festhalten:

Lemma 6.1. Sei $G = (V, E)$ ein Graph. Der Graph $G' = (V', E')$ entstehe aus G , indem ihm ein weiterer Knoten hinzugefügt wird, der mit jedem Knoten aus V benachbart ist, d.h. $V' = V \cup \{v\}$ und $E' = E \cup \{(u, v) \mid u \in V\}$. Dann ist G genau dann outerplanar, wenn G' planar ist.

Beweis: Bei der Hin-Richtung muss man sich nur überlegen, wie man den extra Knoten v in eine outerplanare Darstellung so einbettet, dass der Graph planar bleibt. Wie kann das am geschicktesten gemacht werden?



Die andere Richtung ist auch nicht kompliziert. Wir betrachten dazu eine planare Darstellung von G' . Da v mit allen Knoten verbunden ist, müssen diese „in einem Kreis“ um ihn herum liegen (Abb. 6.8). Löschen wir nun den Knoten v und alle zu ihm inzidenten Kanten, so liegen alle Knoten an einer Fläche. Für eine outerplanare Darstellung von G reicht es, die Darstellung so umzumodellieren, dass diese Fläche zur äußeren Fläche wird. \square

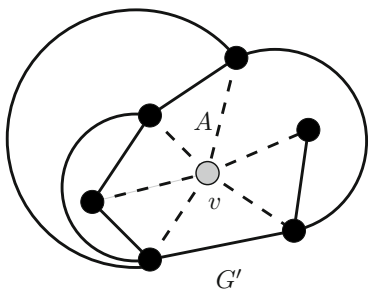


Abb. 6.8: Wird der Knoten v mit seinen Kanten aus der Darstellung von G' gelöscht, so liegen alle Knoten an der Fläche A . Diese kann dann zur äußeren Fläche umgewandelt werden.

In vielen Fällen sind die Graphen interessant, die besonders viele Kanten enthalten. Wir hatten es ja schon häufiger mit vollständigen Graphen zu tun. Einen outerplanaren Graphen bezeichnen wir in diesem Sinn als **maximal**, wenn keine Kante mehr hinzugefügt werden kann, ohne die Eigenschaft der Outerplanarität zu zerstören (Abb. 6.9).

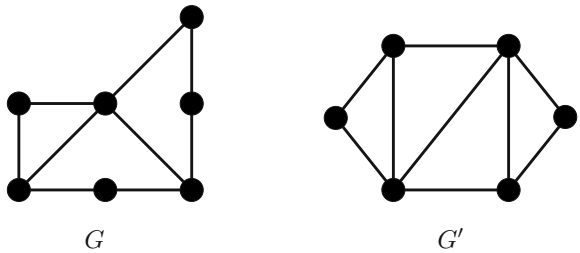


Abb. 6.9: In welche Darstellung können noch Kanten hinzugefügt werden? Welcher Graph ist maximal outerplanar?

Für maximal outerplanare Graphen gilt nun Folgendes:

Satz 6.2. *Alle Knoten eines maximal outerplanaren Graphen liegen auf einem einfachen Kreis, falls $n \geq 3$.*

Beweis: Wir können davon ausgehen, dass ein maximal outerplanarer Graph zusammenhängend ist, da ansonsten zwischen zwei Zusammenhangskomponenten

eine weitere Kante hinzugefügt werden könnte, sodass die Darstellung outerplanar bliebe.

Betrachten wir nun eine outerplanare Darstellung unseres Graphen genauer: Wie schon zum Zählen des Grades der äußeren Fläche konstruieren wir nun einen Weg p , der entlang dieser verläuft. Angenommen, wir kommen bei einem Knoten v zum zweiten Mal vorbei und es seien u und w diejenigen Knoten, die direkt vor und nach v auf dem Weg p liegen (Abb. 6.10). Dann können wir in die Darstellung noch die Kante (u, w) hinzufügen, sodass sie weiterhin outerplanar bleibt. Aufgrund dieses Widerspruchs zur Maximalität des Graphen muss p ein einfacher Kreis sein, der jeden Knoten genau einmal besucht. Damit liegen alle Knoten auf einem einfachen Kreis. \square

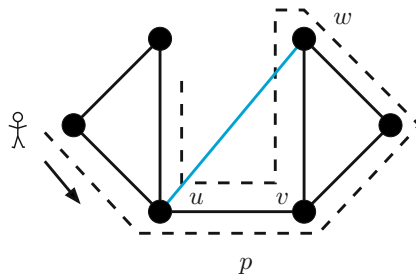


Abb. 6.10: Der Knoten v wird von p zweimal besucht. Also kann die blaue Kante zwischen u und w noch in den Graphen eingefügt werden, ohne die outerplanare Darstellung zu zerstören.

Einfache Kreise, auf denen alle Knoten liegen, hatten wir schon im letzten Kapitel kennen gelernt. Satz 6.2 kann also auch folgendermaßen formuliert werden: Jeder maximal outerplanare Graph mit mehr als drei Knoten enthält einen Hamilton-Kreis. Wieso gilt die Aussage erst ab drei Knoten?

Mit der Aussage von Satz 6.2 lässt sich nun beweisen, dass in jedem outerplanaren Graphen ein Knoten existiert, der höchstens mit zwei Knoten benachbart ist.

Satz 6.3. *In jedem einfachen outerplanaren Graphen existiert ein Knoten v mit $d(v) \leq 2$.*

Beweis: Sei G ein outerplanarer Graph. Fügen Sie dem Graphen so lange Kanten hinzu, bis alle Knoten auf einem einfachen Kreis liegen; dabei soll der Graph jedoch outerplanar bleiben. Der Knotengrad eines jeden Knotens vergrößert sich dadurch höchstens. Wir betrachten nun eine outerplanare Darstellung des neuen Graphen, wobei alle Knoten auf dem Kreis C liegen. Anschließend definieren wir den Abstand zwischen zwei Knoten x und y als die Anzahl der Kanten auf der kürzeren Hälfte, die zwischen x und y auf dem Kreis C liegen (Abb. 6.11).



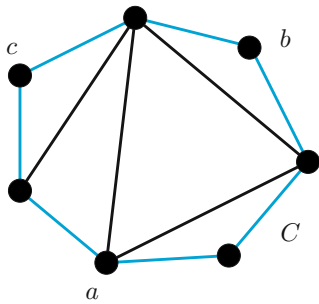


Abb. 6.11: Der Abstand zwischen a und b beträgt drei, zwischen a und c zwei.

Sei nun (u, v) eine Kante, die nicht auf C liegt und für die der Abstand zwischen u und v minimal ist (Abb. 6.12). Da der Graph einfach ist, befindet sich mindestens ein Knoten zwischen u und v . Nehmen wir an, dass dieser Knoten nicht einen Grad 2 hat. Aus der Outerplanarität des Graphen ergibt sich, dass der betreffende Knoten zwischen u und v mit einem weiteren verbunden ist. Ihr Abstand ist dabei kleiner als der zwischen u und v . Dies stellt aber einen Widerspruch zur Wahl der Kante (u, v) dar. \square

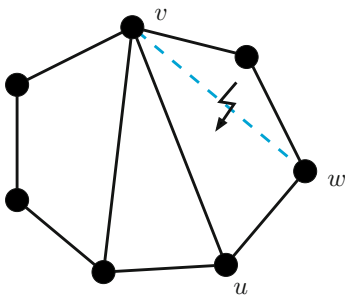


Abb. 6.12: Betrachtet man alle Kanten $(x, y) \in E \setminus C$ und wählt die mit dem geringsten Abstand (hier (u, v)), so hat jeder Knoten zwischen u und v einen Grad von zwei. Die blaue Kante (w, v) kann nicht existieren.

Damit haben wir schon zwei Eigenschaften von outerplanaren Graphen kennen gelernt. Eine weitere Charakterisierung aller outerplanaren Graphen erhalten wir in Abschnitt 6.4. Doch zuvor beweisen wir die berühmte Euler-Formel, die uns die Herleitung einiger Eigenschaften planarer Graphen ermöglicht.

6.3 Die Euler-Formel

Am 14. November 1750 schrieb Euler in einer kuriosen Sprachmischung deutscher und lateinischer Wörter seinem ehemaligen Kollegen Christian Goldbach in St. Petersburg:

„Folgende Proposition aber kann ich nicht recht rigorose demonstrieren:

6. In omni solido hedris planis incluso aggregatum (konvexer Polyeder) ex numero hedrarum H (Flächen) et numero angulorum solidorum S (Ecken) binario superat numerum acierum A (Kanten), seu est $H + S = A + 2$.“

Übersetzt bedeutet das so viel wie: In einem konvexen Polyeder gilt

$$H + S = A + 2,$$

wobei H die Anzahl der Flächen, S die Anzahl der Ecken und A die Anzahl der Kanten angibt. Ein Polyeder ist ein **Körper**, der ausschließlich von geraden Flächen begrenzt wird, wie das zum Beispiel beim Würfel der Fall ist. Er heißt **konvex**, wenn die gradlinige Verbindung zwischen je zwei Punkten des Polyeders ebenfalls im Polyeder liegt. Jeder dieser Körper kann als Graph aufgefasst werden. Dabei werden die Ecken des Körpers zu Knoten und die Kanten bleiben Kanten (Abb. 6.13).

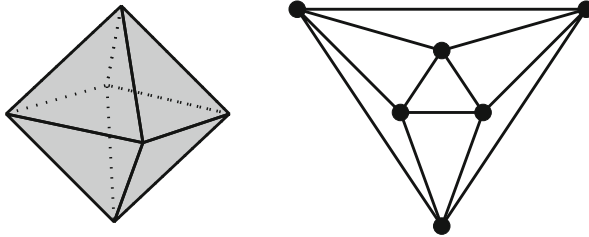


Abb. 6.13: In der Abbildung ist links ein Oktaeder und rechts der dazugehörige planare Graph zu sehen. Wie sehen die Graphen zu den restlichen platonischen Körpern aus?

Für einen konvexen Polyeder erhält man durch diese Umwandlung immer einen planaren Graphen. Deswegen kann der Satz, von dem Euler hier spricht, in die Graphentheorie übertragen werden.

Satz 6.4 (Euler-Formel). *In einem planaren, zusammenhängenden Graphen mit n Knoten, m Kanten und f Flächen gilt*

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

Euler konnte, wie er selber schreibt, die Aussage im Jahr 1750 nicht und zwei Jahre später nur mit Lücken beweisen. Mittlerweile gibt es viele verschiedene Beweise. David Eppstein hat einige auf seiner Webseite veröffentlicht. Wir beschränken uns hier auf einen Induktionsbeweis über die Anzahl der Flächen.

Beweis: Wir führen den Beweis mit Induktion über die Anzahl der Flächen f . Wir beginnen mit dem *Induktionsanfang*. Sei $f = 1$, d.h. die Zeichnung des Graphen G hat nur eine Fläche. Dann darf der Graph nur die äußere Fläche besitzen. Außerdem ist er nach Voraussetzung des Satzes zusammenhängend. Der betrachtete

Graph ist somit ein Baum. Für Bäume wissen wir, dass $m = n - 1$ gilt (Satz 1.8) und damit

$$n - m + f = n - (n - 1) + 1 = 2.$$

Folglich stimmt die Aussage für $f = 1$.

Als Nächstes folgt der *Induktionsschluss*. Sei also G ein planarer, zusammenhängender Graph mit $f \geq 2$ Flächen. Wir nehmen an, dass die Formel für solche Graphen mit weniger Flächen bewiesen ist (*Induktionsvoraussetzung*).

Wegen $f \geq 2$ gibt es einen Kreis in G . Wählen Sie eine Kante e aus einem Kreis und löschen Sie sie (Abb. 6.14). Sei nun $G' = G - e$. Die Kante liegt auf einem Kreis und damit an zwei verschiedenen Flächen. Diese beiden Flächen fallen jetzt in G' zusammen, wodurch G' nicht nur eine Kante, sondern auch eine Fläche weniger hat, d.h. $m' = m - 1$ und $f' = f - 1$. Da G planar ist, ist auch G' planar. Für G' gilt nach Induktionsvoraussetzung (wir haben ja eine Fläche weniger) die Euler-Formel $n' - m' + f' = 2$. Weil die Anzahl der Knoten gleich bleibt, $n' = n$, erhalten wir

$$2 = n' - m' + f' = n - (m - 1) + (f - 1) = n - m + f.$$

Der Beweis der Euler-Formel ist damit abgeschlossen. □

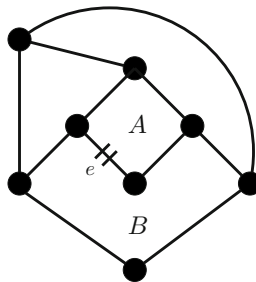


Abb. 6.14: Beim Löschen der Kante e werden die Flächen A und B zu einer Fläche.

Wir hatten den Satz 6.4 so formuliert, dass die Euler-Formel nur für zusammenhängende Graphen stimmt. Betrachten Sie einen Graphen mit n Knoten und keinen Kanten. Dann ist sofort klar, dass die Formel nicht gilt. Es lässt sich aber schnell eine Verallgemeinerung für einen Graphen mit k Zusammenhangskomponenten aufstellen. Wie viele Kanten werden benötigt, um die Zusammenhangskomponenten miteinander zu verbinden. Diese Kanten können ebenfalls planar in die ursprüngliche Darstellung eingebettet werden? Der neue Graph ist zusammenhängend und planar, sodass nun die Euler-Formel gilt. Wie sieht dann die Euler-Formel für nicht zusammenhängende Graphen aus?²²

Die Euler-Formel ist ein wichtiges Hilfsmittel, um Eigenschaften planarer Graphen zu beweisen. Zum Beispiel folgt aus ihr sofort, dass jede planare Darstellung eines Graphen die gleiche Anzahl an Flächen hat. Bisher war nur klar, dass dies für die Anzahl der Knoten und der Kanten gilt. Ist ein planarer Graph dann über die Anzahl der Knoten, Kanten und Flächen eindeutig bestimmt?²³



6.3.1 Erste Anwendungen der Euler-Formel

Als Erstes wollen wir die maximale Anzahl an Kanten eines planaren Graphen in Abhängigkeit seiner Knoten abschätzen. Damit lässt sich zum Beispiel für vollständige Graphen mit mehr als fünf Knoten zeigen, dass sie nicht planar sind.

Satz 6.5. *Für einen planaren Graphen mit mehr als zwei Knoten gilt*

$$m \leq 3n - 6.$$

Beweis: Sei ein planarer Graph G mit einer planaren Darstellung gegeben. Sei f die Anzahl seiner Flächen, m die Anzahl der Kanten und n die Anzahl der Knoten. Wir fügen dem Graphen Kanten hinzu, bis er zusammenhängend ist, dabei aber planar bleibt. Für den neuen Graphen G' gilt nun $n' = n$, $f' \geq f$ und $m' \geq m$. Auf diesen Graphen G' können wir dann später die Euler-Formel anwenden. Jede Fläche F in G' hat nun mindestens eine Länge von drei, d.h. $d(F) \geq 3$. Jede Kante ist entweder in zwei unterschiedlichen Flächen enthalten oder wird bei der Berechnung der Länge einer Fläche doppelt gewertet (Abb. 6.15).

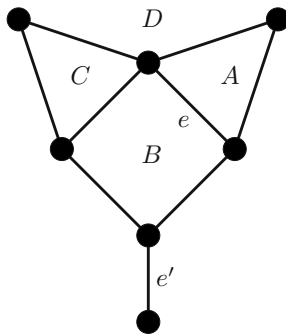


Abb. 6.15: Die Kante e gehört sowohl zur Fläche A als auch zur Fläche B . Die Kante e' hingegen liegt nur in der Fläche D .

Damit gilt

$$\sum_{F \in G'} d(F) = 2m'.$$

Zusammengenommen erhalten wir

$$3f' = \sum_{F \in G'} 3 \leq \sum_{F \in G'} d(F) = 2m'.$$

Setzen wir diese Ungleichung $3f' \leq 2m'$ in die Euler-Formel ein, so gilt

$$\begin{aligned} 3 \cdot 2 &= 3 \cdot (n' - m' + f') \\ &\leq 3n' - 3m' + 2m' \end{aligned}$$

und dies ergibt etwas umgeformt

$$m' \leq 3n' - 6.$$

Da $m' \geq m$ und $n' = n$ gilt, erhalten wir auch für unseren ursprünglichen Graphen G die Formel

$$m \leq m' \leq 3n' - 6 = 3n - 6.$$

□



Damit haben wir die Anzahl der Kanten nach oben gut beschränkt. Aber gibt es überhaupt Graphen, die genau $3n - 6$ Kanten besitzen? Wie sehen diese aus?²⁴ Jetzt könnte man sich auch überlegen, warum ein vollständiger Graph K_n mit $n \geq 5$ nicht planar ist. Wir werden den Beweis auch in Abschnitt 6.4 führen.

Die Hauptidee bei der Bestimmung der oberen Schranke an die Anzahl der Kanten war, den Grad jeder Fläche mit drei abzuschätzen. Angenommen, ein planarer Graph hat keinen Kreis der Länge 3, was zum Beispiel bei bipartiten Graphen der Fall ist. Können wir dann die Anzahl der Kanten besser nach oben abschätzen? Um die obere Schranke so allgemein wie möglich zu halten, definieren wir die **Taillenweite** ℓ eines Graphen als die Länge seines kürzesten Kreises. Als maximale Anzahl seiner Kanten erhalten wir dann:

Satz 6.6. *Sei G ein planarer Graph mit der Taillenweite ℓ , der Anzahl der Knoten n und der Anzahl der Kanten m . Dann gilt*

$$m \leq \frac{\ell}{\ell - 2}(n - 2).$$



Wie kann der Beweis zu Satz 6.5 verändert werden, um diesen Satz zu beweisen?²⁵ Als erste Anwendung des Satzes zeigen wir, dass der Petersen-Graph nicht planar ist. Welche Taillenweite hat er?²⁶

Lemma 6.7. *Der Petersen-Graph (Abb. 6.16) ist nicht planar.*

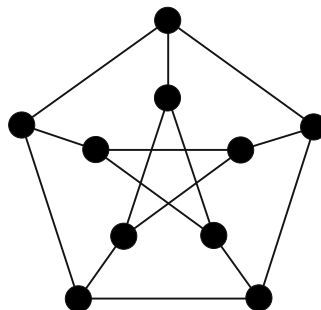


Abb. 6.16: Der Petersen-Graph hat zu viele Kanten, um planar zu sein.

Wir werden uns im nächsten Kapitel mit Knotenfärbungen insbesondere von planaren Graphen beschäftigen. Dazu benötigen wir immer wieder die folgende Aussage:

Lemma 6.8. *Jeder planare Graph hat mindestens einen Knoten mit höchstens fünf Nachbarn.*

Beweis: Angenommen, in einem planaren Graphen haben alle Knoten mindestens sechs Nachbarn, d.h. $d(v) \geq 6$ für alle $v \in V$. Dann gilt nach dem *Handshaking-Lemma*

$$6n \leq \sum_{v \in V} d(v) = 2m \quad \Leftrightarrow \quad 3n \leq m$$

und mit Hilfe des Satzes 6.5

$$3n \leq 3n - 6.$$

Das kann aber nicht stimmen. □

Bisher haben wir nur allgemeine Aussagen über die Anzahl der Flächen und Kanten getroffen. Wie diese Flächen genau aussehen, wissen wir noch nicht. Es gibt aber auch hier einen Zusammenhang zwischen der Anzahl der Flächen und Kanten eines Graphen und den Längen der einzelnen Flächen.

Lemma 6.9. *In jedem planaren Graphen gilt*

$$\sum_{i=1}^{2m} (6-i) \cdot f_i = 6f - 2m,$$

wobei f_i die Anzahl der Flächen mit der Länge i angibt.

In einem planaren Graphen kann eine Fläche maximal eine Länge von $2m$ haben. Deswegen reicht es aus die Werte bis $2m$ aufzusummieren. Welcher Graph hat eine Fläche mit dieser Länge?²⁷



Beweis Lemma 6.9. Die Summe $\sum_{i=1}^{2m} i \cdot f_i$ gibt die Gesamtlänge aller Flächen des Graphen an. Weiter nutzen wir hier erneut die Aussage, dass jede Kante genau zweimal in dieser Summe vorkommt:

$$\sum_{i=1}^{2m} i \cdot f_i = 2m$$

Weiter gilt

$$\sum_{i=1}^{2m} f_i = f.$$

Daraus ergibt sich dann:

$$\begin{aligned} 6f - 2m &= 6 \cdot \sum_{i=1}^{2m} f_i - \sum_{i=1}^{2m} i \cdot f_i \\ &= \sum_{i=1}^{2m} (6 - i) \cdot f_i. \end{aligned}$$

□

Aus der oberen Ungleichung erhalten wir jetzt schnell:

Lemma 6.10. *In jedem planaren zusammenhängenden Graphen mit Minimalgrad $\delta(G) \geq 3$ gilt*

$$\sum_{i=1}^{2m} (6 - i) f_i \geq 12.$$

Für den Beweis reicht es, $6f - 2m \geq 12$ zu zeigen, was sich aus dem *Handshaking-Lemma* 1.1 und der Euler-Formel 6.4 ergibt.²⁸

Aus dem Satz lässt sich einfach folgern, dass in jedem zusammenhängenden planaren Graphen mit $\delta(G) \geq 3$ mindestens eine Fläche mit einer Länge kleiner als 6 existieren muss. Wieso ist das so?²⁹ Auch gilt die folgende Aussage:



Lemma 6.11. *Jeder planare zusammenhängende Graph mit Minimalgrad $\delta(G) \geq 3$, dessen Taillenweite ℓ größer als 4 ist, hat mindestens zwölf Flächen der Länge 5 und somit gilt $\ell = 5$.*

Für den Beweis braucht man sich nur die Ungleichung

$$\sum_{i=1}^{2m} (6 - i) f_i \geq 12$$

genauer anzuschauen und dabei zu überlegen, wann $(6 - i)$ positiv und wann negativ ist. Kennen Sie einen Graphen, der zwölf Flächen hat, die jeweils eine Länge von fünf haben?³⁰

Wir haben jetzt eine ganze Menge Eigenschaften von planaren Graphen kennen gelernt. Mit ihrer vollständigen Charakterisierung beschäftigen wir uns im nächsten Abschnitt.

6.4 Die Graphen $K_{3,3}$, K_5 und Kuratowski

Für zwei spezielle Graphen, den K_5 und den $K_{3,3}$, zeigen wir mit Hilfe der Euler-Formel, dass sie nicht planar sind.

Satz 6.12. *Der K_5 ist nicht planar.*

Beweis: Aus Satz 6.5 wissen wir, dass planare Graphen die Ungleichung

$$m \leq 3n - 6$$

erfüllen. Zählt man jedoch die Knoten und Kanten des K_5 , so gilt $n = 5$ und $m = 10$. Damit hat er zu viele Kanten, um planar zu sein. \square

Auf ähnliche Weise beweisen wir jetzt, dass der $K_{3,3}$ nicht planar ist. Darin besteht dann auch die Lösung des Gas-Wasser-Strom-Problems. Mindestens eines der Häuser muss entweder auf Strom, Wasser oder Gas verzichten, damit die Sicherheitsstandards eingehalten werden können. In der Realität lässt sich das Problem trotzdem praktisch lösen. Fällt Ihnen eine Möglichkeit ein?³¹ Zurück zu unserem Satz und seinem Beweis.



Satz 6.13. *Der $K_{3,3}$ ist nicht planar.*

Beweis: Wir nehmen wieder an, dass der Graph planar ist. Da der $K_{3,3}$ genau 6 Knoten und 9 Kanten besitzt, beträgt die Anzahl der Flächen nach der Euler-Formel 5.

Der $K_{3,3}$ ist bipartit. Also muss die Anzahl der Kanten, die eine Fläche begrenzen, mindestens 4 betragen. Betrachtet man die Gradzahlen $d(F)$ aller Flächen, so gilt

$$\sum_{F \in K_{3,3}} d(F) \geq 4f.$$

Weiter tritt jede Kante bei der Berechnung der Länge aller Flächen genau zweimal auf:

$$\sum_{F \in K_{3,3}} d(F) = 2m.$$

Daraus folgt

$$2m \geq 4f$$

und in diesem Fall

$$2 \cdot 9 = 18 \geq 4 \cdot 5 = 20,$$

ein Widerspruch. \square



An welcher Stelle kann der Beweisgang mit Hilfe von Satz 6.6 verkürzt werden? Jetzt haben wir mit einigem Rechenaufwand bei zwei Graphen ihre Planarität ausgeschlossen. An sich interessieren uns jedoch feststehende Kriterien, nach denen wir beurteilen können, ob ein beliebiger Graph planar ist oder nicht. Dazu noch ein paar Vorüberlegungen: Was passiert, wenn wir eine Kante oder sogar einen Knoten aus einem Graphen löschen? Ist der neue Graph nicht planar, so ist es auch nicht der ursprüngliche. Gleiches gilt, wenn wir zwei durch eine Kante verbundene

Knoten nehmen, die Knoten zusammenziehen und sie jetzt als „einen“ Knoten betrachten (Abb. 6.17). Diese Operation wird auch **Kontraktion** genannt. Ist der neue Graph nach der Kontraktion nicht planar, dann war es der alte auch nicht.

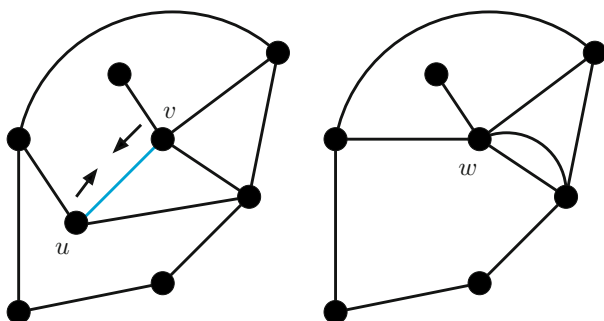


Abb. 6.17: Kontrahiert man die Kante (u, v) , so entsteht der neue Knoten w .

Graphen, die durch die beiden Operationen, also das Löschen einer Kante oder eines Knotens und durch Kontraktion einer Kante, aus einem gegebenen Graphen G entstehen, werden als **Minoren** von G bezeichnet. Mit Hilfe dieser Minoren, dem K_5 und dem $K_{3,3}$ lassen sich nun alle planaren Graphen charakterisieren. Erstmals gezeigt hat dies 1930 der polnische Mathematiker Kasimir Kuratowski.

Satz 6.14 (Kuratowski 1930; Wagner 1937). *Ein Graph ist genau dann planar, wenn er weder einen K_5 noch einen $K_{3,3}$ als Minor enthält.*



Welcher Minor steckt in dem Petersen-Graphen? Wie sieht es mit den folgenden Graphen aus (Abb. 6.18)? Sind sie planar oder enthalten sie einen der beiden Graphen als Minor?

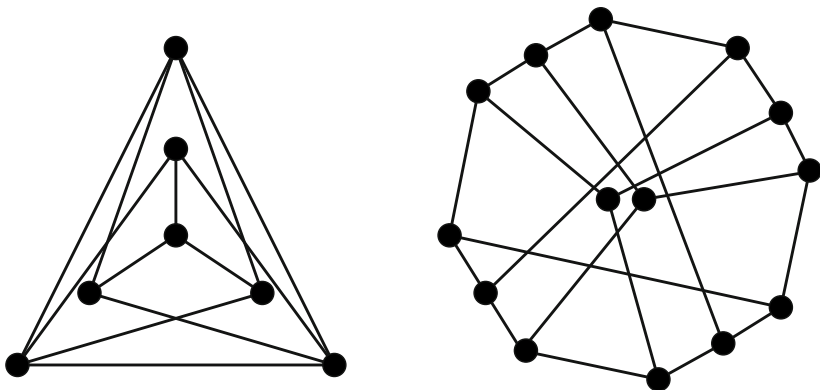


Abb. 6.18: Enthalten die beiden Graphen den $K_{3,3}$ oder den K_5 als Minor?

Der Beweis zum Satz von Kuratowski 6.14 kann hier wegen seiner Länge nicht geführt werden. Aber wir können seine Aussage nutzen, um ebenfalls eine Charakterisierung von outerplanaren Graphen anzugeben.

Satz 6.15 (Chartand and Harary 1967). *Ein Graph ist genau dann outerplanar, wenn er keinen K_4 oder keinen $K_{2,3}$ als Minor enthält.*

Beweis: Sei G ein Graph und G' der Graph, der aus G durch das Hinzufügen eines weiteren Knotens (wir bezeichnen ihn mit v'), der mit allen Knoten von G benachbart ist, entsteht. Diese Art der Konstruktion hatten wir schon in Lemma 6.1 gesehen. Dann gilt nach diesem Lemma: Der Graph G ist genau dann outerplanar, wenn G' planar ist. Und G' ist genau dann planar, wenn er keinen K_5 und $K_{3,3}$ als Minor enthält (Satz von Kuratowski). Abschließend brauchen wir noch die Eigenschaft, dass ein Minor von G' , der nicht den Knoten v' enthält, auch ein Minor von G ist. Der Beweis hierzu ist eher technisch. Deswegen gehen wir davon aus, dass die Aussage stimmt.

Mit diesen beiden Vorüberlegungen können wir jetzt die Verneinung des oben genannten Satzes leicht zeigen (Der Graph G ist genau dann nicht outerplanar, wenn er den K_4 oder den $K_{2,3}$ als Minor enthält):

Sei G nicht outerplanar. Dann ist G' auch nicht planar und enthält somit den K_5 oder den $K_{3,3}$ als Minor (Satz von Kuratowski 6.14).

Fall 1: Der Graph G' enthält den K_5 als Minor. Danach wählen wir den Knoten v' , falls er in dem Minor enthalten ist, oder ansonsten irgendeinen anderen Knoten v aus dem Minor. Nachdem wir den gewählten Knoten aus dem Minor mit all seinen Kanten gelöscht haben, so erhalten wir den K_4 als Minor von G' , der insbesondere nicht den Knoten v' enthält und somit auch ein Minor von G ist.

Fall 2: Der Graph G' enthält den $K_{3,3}$ als Minor. Ist dann der $K_{3,2}$ ein Minor von G ? Reicht das aus für den Beweis? Damit haben wir die erste Richtung bewiesen. Seien nun der K_4 oder $K_{2,3}$ Minoren von G . Dann enthält der Graph G' den K_5 oder den $K_{3,3}$ als Minor, da der Knoten v' mit allen Knoten aus G verbunden ist. (Somit auch mit denen, die den Minor K_4 oder $K_{2,3}$ bilden.) Dann ist G' nicht planar und somit G nicht outerplanar.

Damit ist der Satz bewiesen. □

Im Jahr 1974 veröffentlichten John Hopcroft und Robert Tarjan den ersten effizienten Algorithmus, um einen Graphen auf seine Planarität zu testen. Mit Hilfe von speziellen Datenstrukturen ist dies mittlerweile in linearer Zeit möglich. Wir verlassen jetzt jedoch dieses Thema und beschäftigen uns mit der Knotenfärbung von Graphen.



6.5 Aufgaben

Aufgabe 6.1. Sei n die Anzahl der Knoten, m die Anzahl der Kanten und f die Anzahl der Flächen eines planaren zusammenhängenden Graphen. Welchen Wert haben die fehlenden Parameter in der folgenden Tabelle? Wie sieht ein Graph aus, der die Parameter erfüllt?

| n | m | f |
|-----|-----|-----|
| 12 | 11 | |
| 6 | | 7 |
| | 11 | 3 |

Aufgabe 6.2. Wie viele Kanten müssen aus dem K_5 bzw. dem $K_{3,3}$ gelöscht werden, damit er planar wird?

Aufgabe 6.3. Existiert ein planarer Graph mit 3, 4 oder 5 Knoten, der nicht outerplanar ist?

Aufgabe 6.4. Zeigen Sie, dass der vollständige bipartite Graph $K_{2,6}$ planar ist? Ist auch der $K_{2,n}$ planar? Für welche Werte von n und m (mit $n \leq m$) ist der $K_{m,n}$ planar?

Aufgabe 6.5. Ein Baum hat nur eine Fläche. Ist jeder Graph ein Baum, der nur eine Fläche besitzt?

Aufgabe 6.6. Ist der Minor eines eulerschen Graphen wieder eulersch? Wie sieht es bei hamiltonschen Graphen aus? Was gilt, wenn nur die Kontraktion von Kanten nicht das Löschen von Kanten zulässt?

Aufgabe 6.7. Gibt es ein Tupel n, m, f mit $n \geq 0$, $m \geq 0$ und $f \geq 0$, das die Euler-Formel erfüllt, zu dem es allerdings keinen einfachen planaren Graphen gibt?

Aufgabe 6.8. Gibt es einen planaren Graphen, der genau einen Knoten vom Grad kleiner als 6 hat?

Aufgabe 6.9. Sei G ein zusammenhängender, 3-regulärer (das heißt für alle Knoten $v \in V$ gilt $d(v) = 3$), planarer Graph, sodass jeder Knoten genau an zwei Flächen der Länge 4 und einer Fläche der Länge 6 liegt. Bestimmen Sie die Anzahl der Knoten, Kanten und Flächen von G und zeichnen Sie den Graphen.

Aufgabe 6.10. Sei G ein k -regulärer planarer zusammenhängender Graph. Zeigen Sie, dass $k \leq 5$ gilt.

Aufgabe 6.11. Sei G ein zusammenhängender, einfacher, planarer Graph mit $m = 3n - 6$. Zeigen Sie, dass in jeder planaren Darstellung von G die Länge der Flächen drei ist.

Aufgabe 6.12. Sei G ein zusammenhängender einfacher planarer Graph mit 53 Flächen. Zeigen Sie: Falls es eine planare Darstellung von G gibt, in der jede Fläche mindestens eine Länge von fünf hat, dann hat der Graph mindestens 82 Knoten.

Aufgabe 6.13. Wie viele Kanten müssen mindestens aus dem K_n gelöscht werden, damit der entstandene Graph planar ist?

Lösungen zu den Fragen im Text

- 21 Nein. Bäume haben nur die äußere Fläche.
 22 Um einen Graphen bestehend aus k Zusammenhangskomponenten in einen zusammenhängenden Graphen umzuwandeln, muss man mindestens $k - 1$ Kanten hinzufügen. Die Euler-Formel lautet somit

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

- 23 Nein, vier Knoten, vier Kanten und zwei Flächen können zu mindestens zwei nicht identischen planaren Graphen angeordnet werden.
 24 Ein planarer Graph hat genau dann $3n - 6$ Kanten, wenn jede Fläche eine Länge von 3 hat.
 25 Anstatt

$$3f \leq 2m$$

gilt jetzt

$$\ell f \leq 2m,$$

da weiterhin jede Kante „doppelt“ gezählt wird. Der Rest verläuft analog.

- 26 Der Petersen-Graph hat eine Tailenweite von 5. Dann hat er aber zu viele Kanten nach Satz 6.6.
 27 Bäume: Die äußere Fläche zählt jede Kante zweimal.
 28 Es gilt: $3n \leq 2m$. Eingesetzt in die Euler-Formel erhalten wir

$$\begin{aligned} 12 &= 6n - 6m + 6f \\ &\leq 4m - 6m + 6f \\ &= 6f - 2m. \end{aligned}$$

- 29 Weil sonst die linke Seite negativ bzw. gleich Null wäre.
 30 Der Dodekaeder hat 12 Flächen à 5 Kanten.
 31 Ein Hausbesitzer könnte dem Wasserwerk erlauben, eine der Leitungen durch seinen Keller legen zu lassen. Oder man lässt die Leitungen in unterschiedlichen Erdschichten verlegen. Aber das ist natürlich etwas geschummelt.

7 Knotenfärbung

Übersicht

| | | |
|-----|-------------------------------|-----|
| 7.1 | Die chromatische Zahl | 111 |
| 7.2 | Das Vier-Farben-Problem | 115 |
| 7.3 | Aufgaben | 121 |

7.1 Die chromatische Zahl

Das Abitur steht auch in Herzesberg vor der Tür. Der Direktor der Schule St. Johannes ist deswegen damit beschäftigt, die Tage festzulegen, an denen die Abschlussklausuren geschrieben werden sollen. Natürlich können die Prüfungen in zwei Kursen, die von gleichen Schülern besucht werden, nicht am selben Tag durchgeführt werden. Dem Direktor liegt eine Liste von Kursen und gemeinsamen Teilnehmern vor (Tab. 7.1). Wie viele Tage muss er für den Prüfungszeitraum mindestens vorsehen? Da die anderen Schüler in dieser Zeit frei haben, will er die Prüfungsphase so kurz wie möglich halten. Seine Aufgabe besteht also darin, die Kurse so zu gruppieren, dass die Klausuren einer Gruppe, die am selben Tag geschrieben werden, keine gemeinsamen Teilnehmer haben.

| | Bio | Mat | Gesch | Deu | Eng | Lat | Phy |
|-------|-----|-----|-------|-----|-----|-----|-----|
| Bio | x | | | | x | | x |
| Mat | | x | | x | | | x |
| Gesch | | | x | x | | x | |
| Deu | | x | x | x | x | x | |
| Eng | x | | | x | x | | |
| Lat | | | x | x | | x | |
| Phy | x | x | | | | | x |

Tab. 7.1: In der Liste sind alle Kurse mit gemeinsamen Teilnehmern durch ein x markiert.

Das Problem lässt sich gut mit Konfliktgraphen formulieren: Die Knoten entsprechen dabei den Kursen und ein Konflikt (gemeinsame Teilnehmer) wird durch eine

Kante dargestellt. Eine Gruppe in diesem Graphen entspricht dann einer Knotenmenge, die keine Kanten enthält. Die Knoten einer Gruppe könnten wir mit einer Farbe markieren. Da möglichst wenig Schultage ausfallen sollen, suchen wir eine Färbung des Graphen mit möglichst wenigen Farben. Wie viele Farben benötigen wir in unserem Fall? Alle Kurse mit derselben Farbe können jetzt am gleichen Tag ihre Prüfung ablegen. Die Anzahl der Farben gibt uns somit die Anzahl der benötigten Prüfungstage.



Definition. Eine *(Knoten)-Färbung* eines Graphen $G = (V, E)$ ist eine Abbildung $c : V \rightarrow S$, wobei S eine beliebige Menge ist, sodass $c(u) \neq c(v)$ für alle Knoten $u, v \in V$ mit $(u, v) \in E$ gilt. Die Elemente von S heißen *Farben*. Ein Graph ist *k-färbbar*, falls es eine Färbung $c : V \rightarrow \{1, 2, \dots, k\}$ gibt. Das kleinste k , für das G noch k -färbbar ist, ist die *chromatische Zahl* $\chi(G)$ von G .

Chromatisch kommt übrigens von dem griechischen Wort *chroma* und heißt Farbe. In unserem Prüfungsbeispiel hätte der Direktor durchaus jedem Fach einen eigenen Prüfungstag oder eine eigene Farbe geben können. Der Graph ist also 7-färbbar. Aber es reichen auch drei Farben aus. Genügen auch zwei Farben?

Wie sieht es mit anderen Graphen aus? Wie viele Farben werden benötigt, um einen Baum zu färben? Welche Graphen mit n Knoten brauchen eine Farbe bzw. n Farben? Gibt es hier unterschiedliche Möglichkeiten? Wie sieht die chromatische Zahl bei Kreisen oder Rädern aus? Ein *Rad* ist ein Kreis mit einem zusätzlichen Knoten, der mit allen anderen Knoten verbunden ist. Wie färbt man die folgenden Graphen aus der Abbildung 7.1 am besten? Ohne Ausprobieren macht das Thema nur halb so viel Spaß.³²

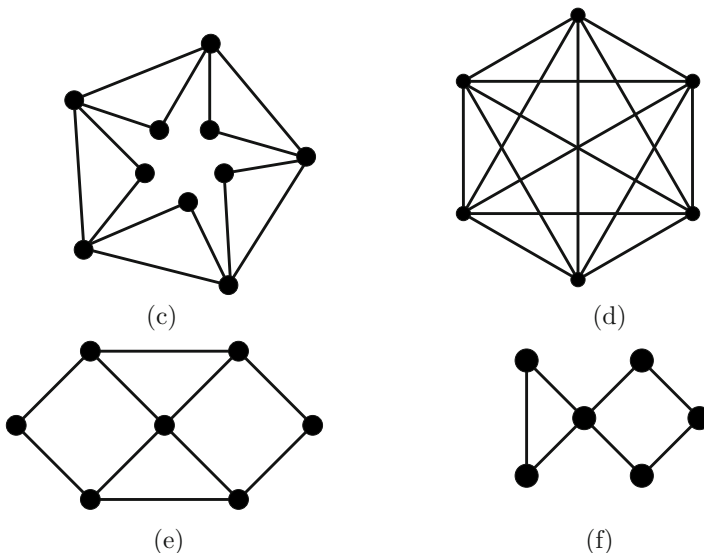


Abb. 7.1: Wie lautet die chromatische Zahl der einzelnen Graphen?

An den Beispielen sieht man schon, dass für jeden Graphen die chromatische Zahl $\chi(G)$ zwischen eins und n liegt, d.h.

$$1 \leq \chi(G) \leq n.$$

Über den maximalen Grad eines Graphen lässt sich diese Abschätzung noch verbessern:

Satz 7.1. *Die chromatische Zahl eines Graphen G liegt immer zwischen eins und $\Delta(G) + 1$, wobei $\Delta(G)$ den Maximalgrad von G angibt.*

Beweis: Den Satz zeigen wir mit einem Algorithmus, der zu jedem Graphen eine Färbung mit maximal $\Delta(G) + 1$ Farben konstruiert.

Algorithmus 7.1 Färbungsalgorithmus

Input: Graph G .

Output: Knotenfärbung mit maximal $\Delta(G) + 1$ Farben.

Schritt 1: Nummerieren Sie die Knoten des Graphen mit v_1, v_2, \dots, v_n .

Schritt 2: Färben Sie v_1 mit der Farbe 1.

Schritt 3: Betrachten Sie nun die Knoten der Reihe nach und wählen Sie für jeden Knoten die kleinste unbenutzte Farbe seiner schon gefärbten Nachbarn (Abb. 7.2).

Der Algorithmus färbt also die Knoten der Reihe nach, sodass zwei benachbarte Knoten keine gemeinsame Farbe erhalten. Da ein Knoten maximal $\Delta(G)$ Nachbarn hat, werden nicht mehr als $\Delta(G) + 1$ Farben verwendet. \square

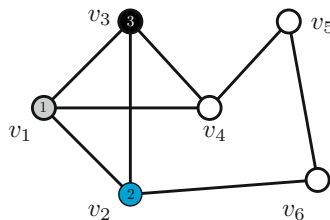


Abb. 7.2: Der Knoten v_4 soll als Nächstes gefärbt werden. Da bisher nur die Farben 1 und 3 bei seinen Nachbarn verwendet wurden, färbt der Algorithmus ihn mit der Farbe 2. Welche Farben erhalten die Knoten v_5 und v_6 ?

Den Algorithmus werden wir später noch einmal verwenden. Zuerst wollen wir uns überlegen, ob die Abschätzungen scharf sind, d.h. ob es Graphen mit $\chi = \Delta + 1$ gibt. Betrachten Sie dazu noch einmal die Beispiele von oben (Abb. 7.1).

Für vollständige Graphen und Kreise ungerader Länge gilt $\chi = \Delta + 1$. Aber gibt es noch mehr Graphen mit dieser Eigenschaft? Die Antwort hierzu lautet erstaunlicherweise nein. Im Jahr 1941 hat der Mathematiker Brooks gezeigt, dass fast alle Graphen schon mit einer Anzahl von $\Delta(G)$ Farben gefärbt werden können. Die einzigen Ausnahmen bilden Kreise ungerader Länge und vollständige Graphen.

Satz 7.2 (Brooks 1941). *Sei G ein zusammenhängender Graph mit mehr als zwei Knoten. Dann gilt $\chi(G) \leq \Delta(G)$, außer G ist vollständig oder ein Kreis ungerader Länge. In diesen beiden Fällen gilt $\chi(G) = \Delta(G) + 1$.*

Mit dem Satz von Brooks 7.2 können wir die Anzahl der zur Färbung benötigten Farben abschätzen. Die chromatische Zahl eines Graphen zu bestimmen, ist jedoch bisher nicht effizient möglich. Trotzdem können wir mit Hilfe des Färbungsalgorithmus 7.1 die chromatische Zahl für jeden Graphen bestimmen. Dazu nehmen wir einmal an, zu einem Graphen G ist uns eine $\chi(G)$ -Färbung gegeben. Als Nächstes sortieren wir die Knoten der Farbe nach, zuerst alle Knoten der Farbe 1, dann alle Knoten der Farbe 2 usw. Wenn wir mit dieser Reihenfolge den Färbungsalgorithmus 7.1 starten, erhalten wir wieder eine $\chi(G)$ -Färbung. Sie können es an dem Graphen mit der Färbung aus der Abbildung 7.3 ausprobieren.



Leider ist uns im Normalfall keine $\chi(G)$ -Färbung oder die entsprechende Reihenfolge gegeben. Wenn wir aber alle möglichen Knotenreihenfolgen ausprobieren, ist auch eine Reihenfolge dabei, die zu einer $\chi(G)$ -Färbung gehört. Der Färbungsalgorithmus berechnet zu der Reihenfolge dann eine $\chi(G)$ -Färbung, und wir müssen nur am Ende die Färbung mit den wenigsten Farben auswählen. Das Problem an dem Algorithmus ist, dass es $n!$ unterschiedliche Knotenreihenfolgen gibt. Der Algorithmus berechnet zwar eine optimale Knotenfärbung, aber es werden sehr viele Rechenoperationen benötigt.

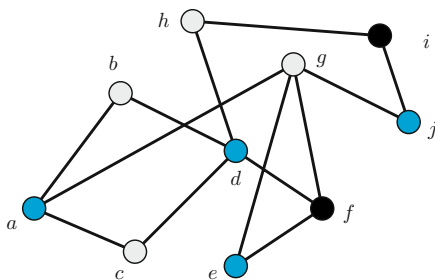


Abb. 7.3: Wenn die Knoten ihrer Farbe nach sortiert werden, konstruiert der Algorithmus 7.1 mit dieser Reihenfolge eine mindestens genauso gute Färbung.

Nun wollen wir uns statt mit dem Färben von Graphen mit dem Färben von Landkarten beschäftigen. Das Thema hat die Graphentheoretiker lange Zeit in Atem gehalten und sorgt auch heute noch für einige Diskussionen.

7.2 Das Vier-Farben-Problem

Ähnlich wie beim Königsberger Brückenproblem liegt der Ursprung der Färbung von Landkarten schon etwas zurück. Im Jahr 1852, genauer am 23. Oktober, schrieb August De Morgan, ein Professor für Mathematik aus London, an seinen Freund Sir William Hamilton: „A student of mine asked me today to give him a reason for a fact which I did not know was a fact — and do not yet. He says that if a figure be anyhow divided and the compartments differently coloured so that figures with any portion of common boundary line are differently coloured — four colours may be wanted, but not more — the following is the case in which four are wanted (Abb. 7.4). ...“

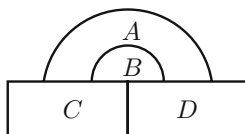


Abb. 7.4: So ungefähr sah die Zeichnung von Francis Guthrie aus. Für diese Landkarte reichen drei Farben nicht aus. Fallen Ihnen noch andere Fälle ein?

Der Student Francis Guthrie, von dem De Morgan spricht, hatte sich einige Landkarten, unter anderem die Karte von England, vorgenommen und es jedes Mal geschafft, für das Färben der einzelnen Länder nicht mehr als vier Farben zu benutzen. Zwei Länder sollten dabei unterschiedlich gefärbt werden, wenn sie eine gemeinsame Grenze haben. Solche Länder werden auch als *benachbart* bezeichnet. Ein Punkt zählt dabei nicht als Grenze. Wäre dies der Fall, würde die Landkarte in Abbildung 7.5 schon sieben Farben zum Färben benötigen. Des Weiteren ging Francis davon aus, dass jedes Land zusammenhängend ist. Darf ein Land aus mehreren Stücken bestehen, ist es einfach, eine Karte zu konstruieren, die mehr als zehn Farben benötigt. Wie müsste sie aussehen?

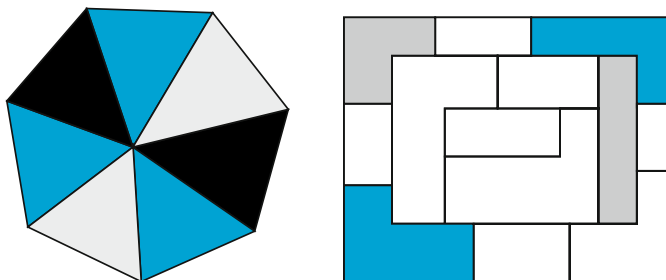


Abb. 7.5: Zwei Länder, die nur einen gemeinsamen Punkt besitzen, dürfen mit den gleichen Farben gefärbt werden. Wie kann man die rechte Landkarte mit möglichst wenig Farben zu Ende färben?

Auch wenn Hamilton kein weiteres Interesse an dem Problem zeigte, wurde es bald als die Vier-Farben-Vermutung bekannt. Viele Mathematiker versuchten sich

an einem Beweis und immer wieder schien das Problem gelöst. Die Schwierigkeit wurde zu Anfang sogar so weit unterschätzt, dass es 1886 am Clifton College in einem Wettbewerb für Schüler ausgeschrieben wurde. Dabei forderten die Veranstalter, dass „keine Lösung mehr als eine Seite, 30 Zeilen eines Manuskripts und eine Seite Diagramme umfasst“. Bis die Vermutung jedoch richtig bewiesen wurde, gingen noch einmal knapp neunzig Jahre ins Land. Dazu später mehr.

Einer der häufigsten Fehler vermeintlicher Lösungen hat seinen Ursprung in dem *Fünf-Prinzen-Problem* von Möbius. Dabei geht es um Folgendes: Es war einmal vor langer Zeit ein mächtiger König in Indien, der fünf Söhne hatte. Da er sich nicht für einen seiner Söhne als Nachfolger entscheiden konnte, wünschte sich der König, dass das Königreich nach seinem Tode in fünf Teile aufgeteilt werden solle, sodass jeder Teil an alle vier anderen Teile grenze. Falls dies den Söhnen nicht gelänge, so sollte keiner der Prinzen König werden. Wie kann das Königreich aufgeteilt werden?

Ein Satz von De Morgan zeigt, dass eine solche Aufteilung unmöglich ist.

Satz 7.3 (De Morgan). *Es gibt keine fünf Länder, sodass jeweils zwei Länder benachbart sind.*

Ob dieser Aussichten waren die Prinzen natürlich verzweifelt, da sie somit ihr Erbe nicht antreten konnten. Eines Tage erschien jedoch ein weiser Magier am Hofe des Königs und präsentierte ihnen eine Lösung. Man sagt, dass er reich beschenkt wurde und die fünf Söhne bis an ihr Lebensende friedlich das geteilte Reich regierten. Was hat der Magier den Prinzen geraten?³³



Die Vier-Farben-Vermutung wurde im Laufe der Jahre immer wieder als Folgerung des Satzes von De Morgan angesehen. Dabei gingen die Beweisführenden davon aus, dass die maximale Anzahl paarweise benachbarter Länder die benötigte Farbenanzahl bestimmt. Länder sind paarweise benachbart, wenn jedes der Länder mit jedem anderen eine gemeinsame Grenze hat. In der Landkarte auf Abbildung 7.5 sind jedoch immer nur zwei Länder miteinander benachbart. Trotzdem kann sie nicht mit weniger als drei Farben gefärbt werden.

Vielleicht haben Sie sich mittlerweile gefragt, warum wir uns mit dem Färben von Graphen beschäftigt haben, wenn es hier nur um Landkarten geht. Der Grund ist einfach: Wir können das Landkartenfärben in ein Graphenfärbungsproblem umwandeln. Dazu betrachten wir nicht mehr die Länder, sondern nur noch ihre Hauptstädte. Weiter gehen wir davon aus, dass die Hauptstädte zweier benachbarter Länder über eine Straße miteinander verbunden sind. Die Hauptstädte zusammen mit den Straßen stellen nun einen Graphen dar, in dem wir eine zulässige Färbung suchen. Die Umwandlung von Ländern mit Grenzen zu Hauptstädten mit Straßen wird auch als Konstruktion des dualen Graphen bezeichnet.

Definition. Sei G ein planarer Graph. Der *duale Graph* G^* zu einer planaren Darstellung von G hat in jeder Fläche von G genau einen Knoten. Des Weiteren

hat G^* für jede Kante $e \in E(G)$ genau eine Kante e' . Diese Kante e' verbindet die Knoten der an e anliegenden Flächen X und Y (Abb. 7.6).

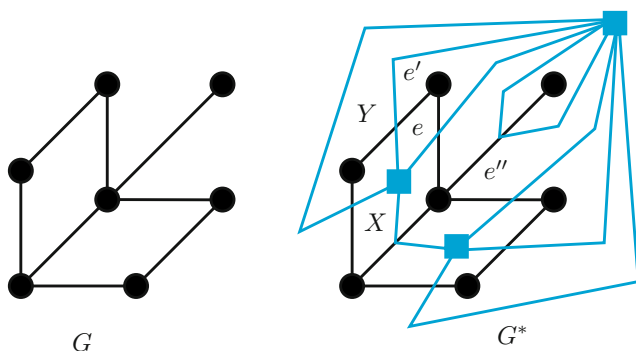


Abb. 7.6: Der duale Graph G^* von G hat genauso viele Kanten wie G .

In manchen Fällen, wie bei der Kante e'' aus der Abbildung 7.6, gilt $X = Y$ und die anliegenden Flächen sind dieselben. Dadurch entsteht eine Schlinge in G^* . Außerdem kann der duale Graph G^* parallele Kanten enthalten. Das liegt daran, dass unterschiedliche Kanten zu zwei gleichen Flächen benachbart sind.

Eine erste Eigenschaft, die sich von planaren Graphen auf ihre dualen Graphen überträgt, ist die Planarität: Als Erstes zeichnen wir alle Knoten des dualen Graphen in die Flächen von G . Als Nächstes halbieren wir die Kanten von G und verbinden die Knoten in der Fläche mit diesen Halbierungspunkten durch Kurven. Dabei achten wir darauf, dass sich die Kurven nicht schneiden. Betrachten wir nun zwei dieser Kurven, die sich an einem Halbierungspunkt treffen, als eine Kante von G^* , so erhalten wir eine planare Darstellung von G^* (Abb. 7.7).

Lemma 7.4. *Der duale Graph G^* zu einem planaren Graphen G ist immer planar.*

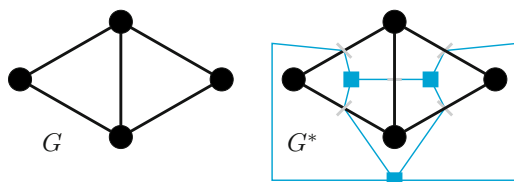


Abb. 7.7: Innerhalb einer Fläche schneiden sich die blauen Kurven nicht.

Die Planarität ist nicht der einzige Zusammenhang zwischen zwei dualen Graphen. Bei zusammenhängenden Graphen lässt sich jede Aussage zu Knoten und Kanten auf Flächen und Kanten des dualen Graphen übertragen. Was bedeutet das im Fall des *Handshaking-Lemmas*? Der Grad eines Knotens kann dabei als Länge



einer Fläche gesehen werden.³⁴

Der duale Graph G^* hängt immer stark von der jeweiligen Darstellung des ursprünglichen Graphen G ab. Dabei müssen die dualen Graphen zu unterschiedlichen Darstellungen eines Graphen nicht dieselben sein. Die dualen Graphen zu den beiden Darstellungen von G in der Abbildung 7.8 unterscheiden sich zum Beispiel in den Gradzahlen.

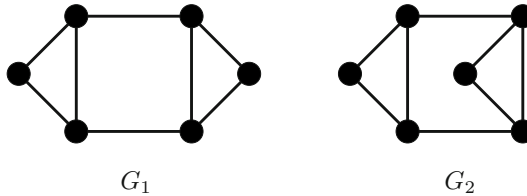


Abb. 7.8: Die dualen Graphen müssen nicht eindeutig sein.

Mit Hilfe der dualen Graphen können wir nun die Vier-Farben-Vermutung auf Graphen übertragen.

Vier-Farben-Vermutung. Jeder planare Graph lässt sich mit vier Farben färben.

Ein erster Meilenstein zur Lösung der Vermutung wurde von Alfred Bray Kempe gelegt, einem Rechtsanwalt und Mitglied der Mathematischen Gesellschaft. Er veröffentlichte 1879 im *American Journal of Mathematics* einen Beweis für die Vier-Farben-Vermutung, der Kempe die Wahl zum *Fellow of the Royal Society* einbrachte. Allerdings deckte Percy John Heawood 1890 einen Fehler in Kempes Beweis auf; Heawood konnte jedoch die von Kempe vorgestellte Technik anwenden, um den Fünf-Farben-Satz 7.5 zu beweisen, und auch der spätere Beweis für die Vier-Farben-Vermutung basiert auf diesen Ideen.

Satz 7.5 (Fünf-Farben-Satz, Heawood 1890). *Jeder planare Graph ist mit fünf Farben färbbar.*

Beweis: Der Beweis nutzt eine vollständige Induktion über die Anzahl der Knoten n . Der *Induktionsanfang* ist nicht kompliziert. Gilt $n \leq 5$, so können wir jedem Knoten eine eigene Farbe zuordnen. Damit sind wir fertig.

Die *Induktionsvoraussetzung* besagt nun: Jeder planare Graph mit weniger als n Knoten ist 5-färbbar. Im *Induktionsschritt* müssen wir zeigen, dass diese Anzahl an Farben auch für einen planaren Graphen mit n Knoten ausreicht.

Sei also $G = (V, E)$ ein planarer Graph und gehen wir davon aus, dass wir eine planare Darstellung gegeben haben. Nach Lemma 6.8 existiert mindestens ein Knoten $v \in V$ mit maximal fünf Nachbarn. Wir betrachten zwei Fälle.

Fall 1: Es gilt $d(v) \leq 4$. Dann löschen wir zunächst v aus G . Der neue Graph $G - v$ hat nur noch $n - 1$ Knoten und ist nach Induktionsvoraussetzung mit fünf Farben färbbar. Fügen wir v wieder in den Graphen $G - v$ ein, so können wir dem

Knoten die Farbe geben, die keiner seiner Nachbarn hat. Damit haben wir eine zulässige Fünf-Färbung von G gefunden (Abb. 7.9).

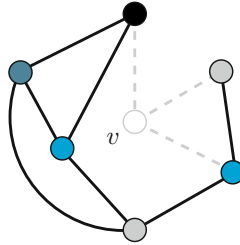


Abb. 7.9: Eine Farbe bleibt immer für v übrig.

Fall 2: Es gilt $d(v) = 5$. Wir betrachten wieder $G - v$ und finden dafür eine zulässige Fünf-Färbung. Werden von den Nachbarn von v nur vier unterschiedliche Farben verbraucht, so können wir v wieder hinzufügen und mit der fünften Farbe färben. Bleibt also nur noch der Fall, bei dem alle fünf Nachbarn von v mit unterschiedlichen Farben gefärbt sind. Wir bezeichnen die fünf Nachbarn im Uhrzeigersinn mit v_1, \dots, v_5 , wobei der Index auch jeweils die Farbe angibt (Abb. 7.10).

Im Folgenden werden wir die Färbung so verändern, dass die Nachbarn von v nur noch mit vier verschiedenen Farben gefärbt sind. Dazu definieren wir uns mit $H_{1,3}$ den Teilgraphen von $H = G - v$, der von allen Knoten der Farbe 1 und 3 induziert wird (Abb. 7.10).

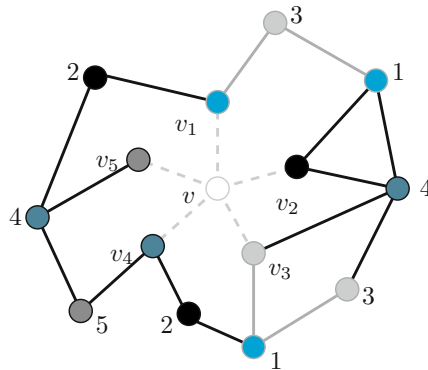


Abb. 7.10: Der Graph $H_{1,3}$ (dunkelgrau eingezeichnet) besteht aus allen Knoten der Farbe 1 und 3 und den dazugehörigen Kanten.

Fall a: In $H_{1,3}$ gibt es keinen Weg von v_1 nach v_3 . Dann können wir in der Zusammenhangskomponente mit v_1 die Knoten umfärben, indem wir allen Knoten der Farbe 1 die Farbe 3 und allen Knoten der Farbe 3 die Farbe 1 zuordnen. Damit ist der Knoten v_1 jetzt mit der Farbe 3 gefärbt und wir können v , nachdem wir ihn wieder dem Graphen hinzugefügt haben, mit 1 färben.

Fall b: Die Knoten v_1 und v_3 sind in $H_{1,3}$ über einen Weg miteinander verbunden (Abb. 7.11). In diesem Fall bringt das Umfärben von Knoten in der Komponente $H_{1,3}$ keine Veränderung an dem Knoten v . Betrachten wir deswegen den Teilgraphen $H_{2,4}$ von H , der analog zu $H_{1,3}$ definiert ist. Zwischen v_2 und v_4 kann kein Weg in $H_{2,4}$ existieren, da der Graph G planar dargestellt ist. Wie im Fall a können wir jetzt die Farben 2 und 4 in der Komponente von v_2 vertauschen. Somit lässt sich v wieder in $G - v$ einfügen und mit der Farbe 2 färben.

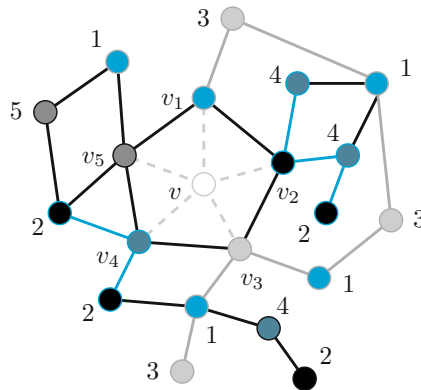


Abb. 7.11: Wenn v_1 und v_3 in einer Komponente liegen (hier dunkelgrau eingezeichnet), sind die Knoten v_2 und v_4 getrennt (blau eingezeichnet).

Damit haben wir eine gültige Fünf-Färbung von G gefunden, d.h. $\chi(G) \leq 5$, und der Fünf-Farben-Satz ist bewiesen. \square

Wie ging es nach Heawood weiter mit dem Vier-Farben-Problem? In den folgenden Jahrzehnten beschäftigten sich viele Mathematiker mit dem Beweis, sogar derart exzessiv, dass einige ihre Kinder und Frauen Landkarten färben ließen. 1922 konnte Philip Franklin zeigen, dass jeder planare Graph mit 25 Knoten 4-färbbar ist. Die Zahl wurde dann über die Jahre hinweg auf 27, 39 und 96 Knoten erweitert. Im Jahr 1976 gelang Kenneth Appel und Wolfgang Haken schließlich der Durchbruch: Sie veröffentlichten einen Beweis, in dem sie für 1.936 schwierige Fälle mit Hilfe eines Computerprogramms eine Färbung mit vier Farben angaben. Mit den Arbeiten von Heinrich Heesch, der in den 60er- und 70er-Jahren ein Verfahren entwickelt hatte, das Problem auf diese schwierigen Fälle zu reduzieren, war das Problem damit bewiesen. Die Universität von Illinois, an der die beiden Wissenschaftler Apple und Haken arbeiteten, ließ aus berechtigtem Stolz eine Zeit lang die Briefumschläge mit „four colors suffice“ stempeln. Es gilt also tatsächlich:

Satz 7.6 (Vier-Farben-Satz). *Für jeden planaren Graphen G gilt $\chi(G) \leq 4$.*

Nicht alle Mathematiker sind mit dem bisherigen Beweis zufrieden bzw. sehen den Satz als bewiesen an. Dabei kritisieren sie, dass es für keinen Menschen mög-

lich ist, den Beweis nachzuvollziehen. Letztendlich muss man sich hierbei auf die Korrektheit der Hardware und des Compilers verlassen. Von einer mathematischen Eleganz kann man bei diesem Beweis auch nicht sprechen. An dem Beweis wird deswegen weiterhin geforscht. Im Jahr 1996 konnten Neil Robertson, Daniel Sanders, Paul Seymour und Robin Thomas den Beweis auf 633 Fälle reduzieren. Immer noch zu viele, um sie „per Hand“ durchzugehen. Vielleicht besteht die Faszination des Vier-Farben-Problems darin, dass der Sachverhalt so einfach zu erklären ist, der Beweis jedoch unglaublich kompliziert ausfällt.

7.3 Aufgaben

Aufgabe 7.1. Jim hat sechs Kinder, die sich ständig streiten, und zwar Jan mit Jessica, Jessica mit Dorothee, Dorothee mit Jakob, Jakob mit Jochen, Jochen mit Johanna, Johanna mit Dorothee, Jessica mit Jakob und Jakob mit Jan. Wie viele Zimmer zum Spielen braucht er, damit endlich Frieden herrscht?

Aufgabe 7.2. Wie kann das folgende Bild (Abb. 7.12) mit vier Farben gefärbt werden?

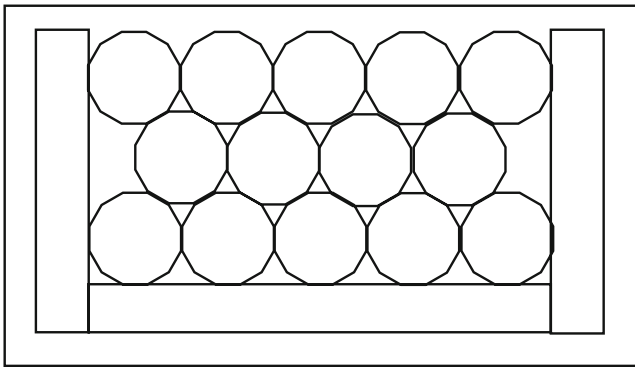


Abb. 7.12: Zwei aneinander grenzende Flächen dürfen nicht gleich angemalt werden.

Aufgabe 7.3. Im Jahr 1975 veröffentlichte ein bekannter Kolumnenschreiber, Martin Gardner, im Scientific American am 1. April einen Artikel über „*Six sensational discoveries that somehow or another have escaped public attention*“. Unter anderem schrieb er:

„The most sensational of last year’s discoveries in pure mathematics was surely the finding of a counterexample to the notorious four-color-map conjecture... Last November, William McGregor, a graph theorist of Wappingers Fall, N.Y., constructed a map of 110 regions that cannot be colored with fewer than five colors.“ In der Abbildung 7.13 finden Sie das oben genannte Gegenbeispiel. Kann man die Karte wirklich nicht mit vier Farben färben?

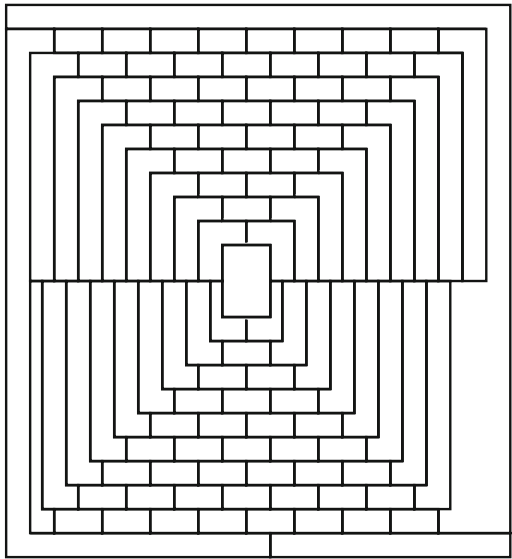


Abb. 7.13: Diese Karte erhitzte viele Gemüter zwischen April und Juni 1975.

Aufgabe 7.4. Finden Sie eine optimale Färbung für die folgenden Graphen und eine Reihenfolge der Knoten, sodass der Färbungsalgorithmus den Graphen möglichst schlecht färbt (Abb. 7.14).

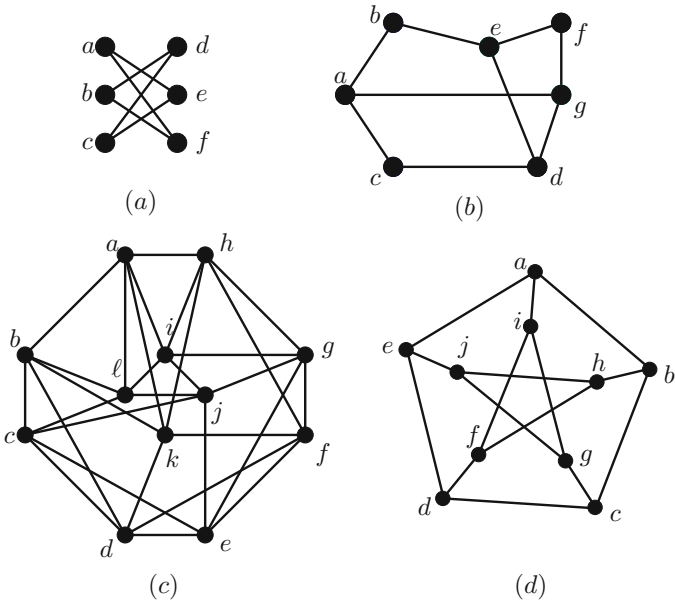


Abb. 7.14: Welche chromatische Zahl haben die Graphen?

Aufgabe 7.5. An verkehrsreichen Straßenkreuzungen und Einmündungen sollen Ampelanlagen verhindern, dass Fußgänger oder Fahrzeuge gleichzeitig unterwegs

sind, wenn es zu Kollisionen kommen kann. An der Kreuzung in Abbildung 7.15 sollten A und G nicht gleichzeitig grün haben, während sich der Verkehrsstrom von A und F nicht stören. Wie kann man die Situation als Graphen darstellen? Wie viele verschiedene Ampelphasen muss es mindestens geben, damit keine Kollisionen entstehen?

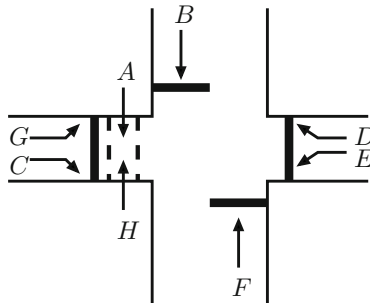


Abb. 7.15: Wie kann das Problem mit Hilfe der Graphentheorie gelöst werden.

Aufgabe 7.6. Beweisen Sie den Satz von De Morgan: Es gibt keine fünf Länder, sodass jeweils zwei Länder benachbart sind.

Aufgabe 7.7. Ein Graph ist genau dann 2-färbbar, wenn er keine ungeraden Kreise enthält.

Aufgabe 7.8. Formulieren Sie Sudoku als graphentheoretisches Problem.

Aufgabe 7.9. Die Abbildung 7.16 besteht aus übereinander gezeichneten Kreisen. Sie kann mit zwei Farben blau und weiß so gefärbt werden, dass zwei Flächen, die einen gemeinsamen Begrenzungsbogen besitzen, unterschiedliche Farben erhalten. Zeigen Sie, dass dies für jede Zeichnung gilt, die aus übereinander gezeichneten Kreise besteht.

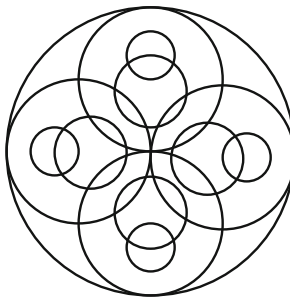


Abb. 7.16: Wie kann diese Zeichnung mit zwei Farben gefärbt werden?

Aufgabe 7.10. Zeigen Sie: Jeder outerplanare Graph ist mit drei Farben färbbar.

Aufgabe 7.11. Betrachten Sie den K_n .

1. Wie viele Farben werden benötigt, wenn eine Kante aus dem K_n gelöscht wird?
2. Wie viele Farben werden benötigt, wenn zwei Kanten, die denselben Endknoten haben, aus dem K_n gelöscht werden?
3. Wie viele Farben werden benötigt, wenn zwei Kanten, die nicht denselben Endknoten haben, aus dem K_n gelöscht werden?

Lösungen zu den Fragen im Text

32 Die Fragen lassen sich folgendermaßen zusammenfassen.

1. Es gilt

$$\chi(G) = 1 \Leftrightarrow E = \emptyset$$

2. Hat ein Graph mindestens eine Kante, so gilt

$$\chi(G) = 2 \Leftrightarrow G \text{ bipartit.}$$

3. Für ein Rad R_n gilt $\chi(G) = \begin{cases} 3 & \text{für ungerade } n \\ 4 & \text{für gerade } n \end{cases}$

4. Sei G ein Graph mit n Knoten, dann ist G genau dann vollständig, wenn $\chi(G) = n$ gilt.

33 Der Magier schlug den Prinzen vor, eine Brücke zwischen zwei Ländern zu bauen, um so die letzte Nachbarschaft herzustellen. Das entspricht natürlich keiner zulässigen Lösung im streng mathematischen Sinn, aber sie ist durchaus praktisch umsetzbar.

34 Das *Handshaking*-Lemma überträgt sich wie folgt auf den dualen Graphen: Für jeden planaren Graphen gilt

$$\sum_{F \in G} \ell(F) = 2m,$$

wobei F die Flächen von G bezeichnet und $\ell(F)$ deren Länge ist. Diese Aussage hatten wir früher schon verwendet.

8 Gerichtete Graphen und Turniergraphen

Übersicht

| | | |
|-----|---------------------------------------|-----|
| 8.1 | Gerichtete Graphen — Digraphen | 125 |
| 8.2 | Starker Zusammenhang | 128 |
| 8.3 | Gerichtete Euler-Graphen | 132 |
| 8.4 | Hamilton-Wege in Turniergraphen | 134 |
| 8.5 | Könige in Turniergraphen | 136 |
| 8.6 | Aufgaben | 141 |

8.1 Gerichtete Graphen — Digraphen

In den letzten Kapiteln haben wir schon einige Fragestellungen kennen gelernt, die sich mit Hilfe von Graphen beantworten lassen. Zum Beispiel hat Euler die Landmassen zwischen den Brücken als Knoten und die Brücken als Kanten modelliert, um das Königsberger Brückenproblem zu lösen. Dabei geht er davon aus, dass jede Brücke in zwei Richtungen begehbar ist. In der Praxis treten jedoch häufig Probleme auf, in denen eine „Beziehung“ zwischen zwei Objekten einseitig ist. Nur weil Anna Markus mag, muss Markus nicht unbedingt Anna mögen. Oder wenn wir ein Straßennetz als Graphen modellieren, müssen wir auch Einbahnstraßen berücksichtigen. Wir benötigen also Richtungen an den Kanten eines Graphen. Solche Graphen mit Richtungen nennt man auch gerichtete Graphen oder Digraphen (engl. *directed graph*). In Darstellungen von gerichteten Graphen deutet man die Richtung mit Hilfe von Pfeilen an (Abb. 8.1).

Definition. Ein Graph $G = (V, E)$ mit einer Orientierung heißt **gerichteter Graph** oder **Digraph**. Eine **Orientierung** weist jeder Kante eine Richtung zu, d.h. einen Anfangs- und einen Endknoten. Eine Kante, die von u nach v orientiert ist, schreiben wir auch als Kante (u, v) . Der Graph G ohne Orientierung wird als der **zugrunde liegende ungerichtete Graph** bezeichnet.

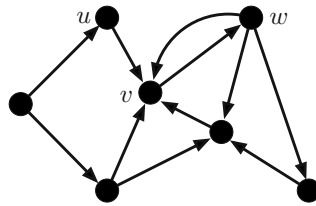


Abb. 8.1: Wenn wir u als Anna, v als Markus und w als Linda interpretieren, mag Anna Markus, Markus Linda und Linda Markus. Markus scheint Anna hingegen nicht zu mögen.

Wir hatten auch in ungerichteten Graphen eine Kante von u nach v mit (u, v) bezeichnet. Im ungerichteten Fall könnte man diese Kante auch als (v, u) schreiben. In einem Digraphen ist immer der erste Knoten der Anfangsknoten der Kante und der zweite der Endknoten.

Die meisten Begriffe, die wir bisher für Graphen kennen gelernt haben, lassen sich eins zu eins auf gerichtete Graphen übertragen. Wir sprechen weiterhin von Knoten und Kanten, Nachbarn und dem Grad eines Knotens. Manchmal unterteilen wir den Grad in den **Ingrad** $d^-(v)$, d.h. die Anzahl der Kanten, die v als Endknoten haben, und den **Ausgrad** $d^+(v)$, d.h. die Anzahl der Kanten, die v als Anfangsknoten haben. Welchen Ingrad und welchen Ausgrad hat der Knoten v aus der Abbildung 8.1?



Auch Teilgraphen und insbesondere (gerichtete) Wege sind wie bei ungerichteten Graphen definiert. Für Kreise gilt jedoch eine kleine Ausnahme: Spricht man von einem **Kreis** in einem gerichteten Graphen, so muss dieser nicht in eine Richtung orientiert sein (Abb. 8.2). Wir tun so, als ob wir gegen Einbahnstraßen fahren dürften. Will man andeuten, dass alle Kanten eines Kreises in die gleiche Richtung zeigen, so bezeichnet man diesen als **gerichteten Kreis** oder als **Zykel**. Eine ähnliche Ausnahme bilden Schnitte. Ein **Schnitt** in einem gerichteten Graphen besteht aus Kanten, die zwischen zwei Knotenmengen X und $V \setminus X$ mit $X \neq \emptyset$ und $X \subsetneq V$ verlaufen, wobei die Orientierung dieser Kanten keine Rolle spielt. In einem **gerichteten Schnitt** hingegen sind alle Kanten, die zwischen X und $V \setminus X$ verlaufen, in eine Richtung orientiert (Abb. 8.2).

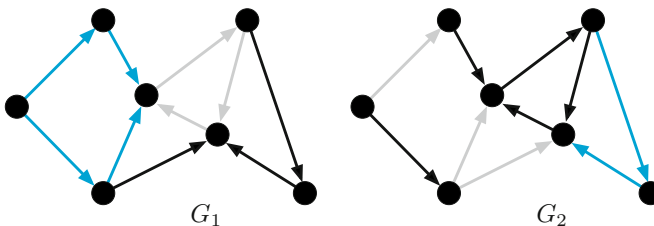


Abb. 8.2: In G_1 bilden die grauen Kanten einen Zykel bzw. einen gerichteten Kreis. Die blauen Kanten ergeben einen Kreis. Die grauen Kanten in G_2 sind dann ein gerichteter Schnitt, wohingegen die blauen nur einen Schnitt darstellen.

Aber nicht nur die Begriffe, sondern auch die Algorithmen lassen sich einfach auf gerichtete Graphen übertragen. Betrachten wir dazu die Breitensuche (Kapitel 2, bzw. hier Algo. 8.1). Ziel der Breitensuche ist es, einen Teilgraphen von G zu bestimmen, der einen Baum bildet. Dabei starten wir von einem beliebigen Knoten v_0 und speichern nach und nach Wege von v_0 aus zu allen Knoten, die von v_0 erreichbar sind.

Algorithmus 8.1 Breitensuche (engl. *breadth first search*, BFS)

Input: Sei $G = (V, E)$ ein zusammenhängender Graph.

Output: Ein aufspannender Baum $T = (V, E_0)$.

- Schritt 1:
- Sei E_0 zunächst leer.
 - Bereiten Sie zwei ebenfalls leere Knotenlisten L_1 und L_2 vor.
 - Wählen Sie einen beliebigen Startknoten $v_0 \in V$ und schreiben Sie v_0 in L_1 und L_2 .
- Schritt 2: Falls L_1 leer ist, dann STOPP. Ansonsten nehmen Sie den ersten Knoten v aus L_1 .
- Schritt 3: Falls alle Nachbarknoten von v in L_2 sind, so löschen Sie v aus L_1 . Ansonsten wählen Sie einen der Nachbarknoten w aus, der nicht in L_2 ist, schreiben Sie diesen ans Ende der Listen L_1 und L_2 und fügen Sie die Kante (v, w) zu E_0 hinzu.
- Schritt 4: Gehen Sie zu Schritt 2.
-

Betrachten wir nun einen gerichteten Graphen G und einen Knoten v_0 und passen den Schritt 3 folgendermaßen an:

- Schritt 3': Falls alle von v über eine Kante (v, w) erreichbaren Knoten in L_2 sind, so löschen Sie v aus L_1 . Ansonsten wählen Sie einen dieser Knoten w aus, der nicht in L_2 ist, schreiben Sie diesen ans Ende der Listen L_1 und L_2 und fügen Sie die Kante (v, w) zu E_0 hinzu.

Mit dieser kleinen Veränderung berechnet der Algorithmus einen Baum, der aus gerichteten (v_0, x) -Wegen besteht und alle von v_0 über einen gerichteten Weg erreichbare Knoten enthält.

In den folgenden Abschnitten werden wir noch mehr Algorithmen und Themengebiete kennen lernen, die von ungerichteten Graphen auf gerichtete Graphen übertragen werden können. Fangen wir dazu mit dem Zusammenhang bzw. den Zusammenhangskomponenten eines Graphen an. Dabei wird das Konzept noch etwas erweitert und nennt sich dann starker Zusammenhang.

8.2 Starker Zusammenhang

Beginnen wir mit der Definition von Zusammenhang bei gerichteten Graphen.

Definition. Ein gerichteter Graph heißt **zusammenhängend**, wenn sein zugrunde liegender ungerichteter Graph zusammenhängend ist.

Garantiert der Zusammenhang eines gerichteten Graphen jetzt auch, dass je zwei Knoten über einen gerichteten Weg miteinander verbunden sind, wie das bei ungerichteten Graphen der Fall ist? Das dies nicht der Fall ist, wurde der Begriff des starken Zusammenhangs eingeführt.

Definition. Ein gerichteter Graph $G = (V, E)$ heißt **stark zusammenhängend**, wenn zu je zwei Knoten $s, t \in V$ ein gerichteter Weg von s nach t und von t nach s existiert. Eine **starke Zusammenhangskomponente** H von G ist ein maximal stark zusammenhängender Teilgraph von G .

Wir wollen uns mit der Frage beschäftigen, wann ein gerichteter Graph stark zusammenhängend ist.

Satz 8.1. *Sei G ein Digraph. Dann sind die folgenden Aussagen äquivalent:*

1. G ist stark zusammenhängend.
2. G enthält keinen gerichteten Schnitt.
3. G ist zusammenhängend und jede Kante liegt auf einem gerichteten Kreis.

Der Satz gibt uns also zwei unterschiedliche Möglichkeiten, einen Digraphen auf starken Zusammenhang zu überprüfen. Zum einen können alle möglichen Schnitte darauf untersucht werden, ob sie gerichtet sind. Zum anderen könnte man für jede Kante nachvollziehen, inwiefern sie auf einem gerichteten Kreis liegt (Wie macht man das?). Als Nächstes wollen wir uns den Beweis von Satz 8.1 anschauen.



Beweis von Satz 8.1. $(1) \Rightarrow (2)$: Angenommen, es existiert ein gerichteter Schnitt, etwa $\delta(X)$ mit der Menge $X \subsetneq V$. Dann kann es nicht sowohl einen gerichteten Weg von X nach $V \setminus X$ als auch einen von $V \setminus X$ nach X geben. Damit ist der Graph aber nicht stark zusammenhängend.

$(2) \Rightarrow (3)$: Wir zeigen zuerst, dass der Graph G auf jeden Fall zusammenhängend ist. Nehmen wir an, G ist nicht zusammenhängend. Seien X die Knoten einer der Zusammenhangskomponenten. Dann gilt $\delta(X) = \emptyset$ und bildet somit einen gerichteten Schnitt. Damit haben wir einen Widerspruch.

Als Nächstes zeigen wir (wieder über einen Widerspruchsbeweis), dass jede Kante auf einem gerichteten Kreis liegt. Angenommen, die Kante $e = (u, v)$ liegt auf keinem gerichteten Kreis. Alle Knoten, die von v über einen gerichteten Weg erreichbar sind, bezeichnen wir mit X_v . Der Knoten u kann nicht in dieser Men-

ge liegen, da ansonsten ein gerichteter Weg von u nach v existieren und damit die Kante e auf einem gerichteten Kreis liegen würde. Dies widerspricht unserer Annahme. Betrachten wir nun den von X_v induzierten Schnitt $\delta(X_v)$. Nach der Definition von X_v sind alle Kanten von $V \setminus X_v$ nach X_v orientiert, sodass mit $\delta(X_v)$ ein gerichteter Schnitt vorliegt. Aufgrund der Annahme, dass in dem Graphen kein gerichteter Schnitt existiert, haben wir nun einen Widerspruch: Jede Kante liegt auf einem gerichteten Kreis.

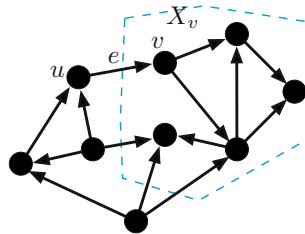


Abb. 8.3: Alle Knoten in X_v können von v aus erreicht werden. Sie sind blau umrandet. Dann bildet $\delta(X_v)$ aber einen gerichteten Schnitt.

(3) \Rightarrow (1): Wir werden nun zeigen, dass von jedem Knoten $r \in V$ jeder andere Knoten erreichbar ist. Sei w ein Knoten mit $w \neq r$. Weiter nehmen wir an, dass kein Weg von r nach w existiert. Die Argumentation ähnelt sehr der von oben. Mit X_r bezeichnen wir wieder die Menge aller Knoten, die von r aus erreichbar sind (Abb. 8.4). Der Knoten w liegt auch dieses Mal nicht in X_r . Nach der Definition von X_r kann keine Kante von X_r nach $V \setminus X_r$ führen. Da G allerdings zusammenhängend ist, gibt es mindestens eine Kante e , die die beiden Mengen miteinander verbindet. Dann gilt also $e = (u, v)$ mit $u \in V \setminus X_r$ und $v \in X_r$. Diese Kante liegt auf einem gerichteten Kreis, d.h. es gibt einen gerichteten Weg, der den Knoten v mit dem Knoten u verbindet. Dieser Weg führt aber auch über den Schnitt $\delta(X_r)$, zu dem eine Kante e' mit entgegengesetzter Orientierung gehört, d.h. ihr Anfangsknoten liegt in X_r und ihr Endknoten in $V \setminus X_r$. Das ist aber ein Widerspruch zu unserer Definition der Menge X_r . \square

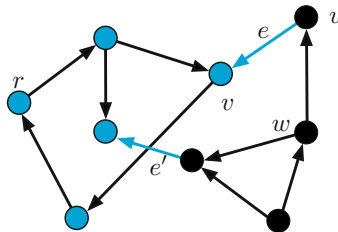


Abb. 8.4: Die blauen Knoten sind vom Knoten r aus erreichbar und bilden somit die Menge X_r . In der anderen Knotenmenge ist w enthalten. Da der Graph zusammenhängend ist, muss zwischen den schwarzen und den blauen Knoten mindestens eine Kante verlaufen (blau eingefärbt).

Wir können nun überprüfen, ob ein Graph stark zusammenhängend ist.

Eine leicht veränderte Fragestellung ergibt sich aus der folgenden Situation: In einer Stadt sind die Straßen alle sehr schmal (Abb. 8.5). Damit der Autoverkehr besser organisiert werden kann, sollen überall Einbahnstraßen eingerichtet werden. Natürlich muss dabei gewährleistet werden, dass man von jedem Punkt der Stadt zu jedem anderen gelangen kann. Wie müsste die Stadtverwaltung die Einbahnstraßenschilder aufstellen lassen? Kann man das Problem mit Hilfe der Graphentheorie lösen?

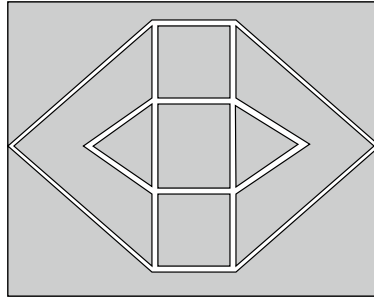


Abb. 8.5: In dieser Stadt sollen alle Straßen in Einbahnstraßen umgewandelt werden.

Für eulersche Graphen ist die Antwort einfach: Orientiere die Kanten einfach entlang einer Euler-Tour. Auch bei allen hamiltonschen Graphen können wir die Kanten entlang des Hamilton-Kreises orientieren und haben damit schon garantiert, dass jeder Knoten von jedem Knoten aus erreichbar ist. Grundsätzlich existiert eine solche Orientierung auch für andere Graphen, sofern sie keine Brücke enthalten (d.h. Kanten, die den Graphen durch ihr Löschen in mehrere Zusammenhangskomponenten zerfallen lassen (Definition auf Seite 13)).

Satz 8.2. *Ein ungerichteter Graph kann genau dann zu einem stark zusammenhängenden Digraphen orientiert werden, wenn er zusammenhängend ist und keine Brücke enthält.*

Beweis: Wir beginnen mit der Hin-Richtung. Sei \vec{G} ein stark zusammenhängender Digraph von G . Sei $e = (u, v)$ eine beliebige Kante. Dann existiert ein Weg in \vec{G} von v nach u , da der Graph stark zusammenhängend ist. Damit liegt die Kante e auf einem Kreis. Warum kann sie dann keine Brücke sein?



Für die Rück-Richtung benutzen wir eine ähnliche Idee, wie sie im Algorithmus von Hierholzer zur Konstruktion einer Euler-Tour enthalten ist: Wir wählen eine beliebige Kante $e = (u, v)$ aus. Da e keine Brücke ist, muss es einen weiteren Weg von v nach u geben. Zusammen mit der Kante e erhalten wir einen Kreis (Abb. 8.6 (b)). Die Kanten in diesem Kreis orientieren wir entlang einer Richtung. Enthält der Kreis alle Knoten, so können wir die anderen Kanten beliebig orientieren.

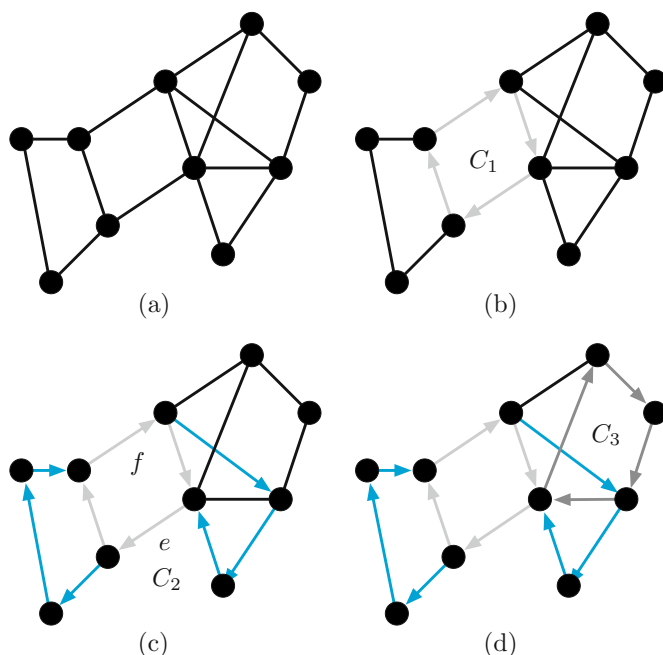


Abb. 8.6: Im ersten Schritt haben wir den Kreis C_1 gefunden und orientieren ihn in einer beliebigen Richtung (b). Der zweite Kreis C_2 besteht aus den blauen Kanten und den Kanten e, f . Da e, f schon gerichtet sind, ist die Orientierung von C_2 festgelegt (c). Der Kreis C_3 hingegen kann beliebig gerichtet werden (d).

Falls ein Knoten noch nicht erreicht wurde, gibt es sicher auch einen, der direkt mit dem bisherigen Kreis durch eine Kante verbunden ist. Ansonsten wäre der Graph nicht zusammenhängend. Von dieser Kante aus konstruieren wir einen Kreis aus nicht gerichteten und evtl. schon gerichteten Kanten. Dabei müssen wir darauf achten, dass die schon gerichteten Kanten in dieselbe Richtung zeigen. Wieso kann ein solcher Kreis immer konstruiert werden?³⁵ Besteht der Kreis nur aus noch nicht orientierten Kanten, so orientieren wir ihn in einer beliebigen Richtung. Enthält er hingegen schon gerichtete Kanten, so orientieren wir die übrigen Kanten in derselben Weise. Diese Prozedur können wir so lange fortsetzen, bis alle Knoten an mindestens einer orientierten Kante liegen. Nun ist jeder Knoten von jedem anderen erreichbar und die restlichen Kanten können eine beliebige Richtung erhalten. Wir haben damit eine gesuchte Orientierung bestimmt. \square



Mit diesen Aussagen können wir jetzt für die Stadtverwaltung ein sinnvolles Einbahnstraßensystem entwickeln.

8.3 Gerichtete Euler-Graphen

Beim Königsberger Brückenproblem war Euler davon ausgegangen, dass die Brücken jeweils in beide Richtungen begehbar sind — eine für die damalige Zeit wie auch für heutige Spaziergänge unstreitige Annahme. Allerdings kann es vorkommen, dass wir ein Straßennetz mit Einbahnstraßen gegeben haben und trotzdem eine Tour suchen, die jede Straße genau einmal beinhaltet. Für die Müllabfuhr und die Paketzusteller ist diese Fragestellung durchaus von Interesse. Wer will schon länger unterwegs sein, als er unbedingt muss? Der Begriff der Euler-Tour lässt sich auf gerichtete Graphen übertragen.

Definition. Ein gerichteter Graph G heißt genau dann **eulersch**, wenn ein gerichteter Kreis in G existiert, der jede Kante des Graphen genau einmal enthält. Ein solcher Kreis wird wieder als **Euler-Tour** bezeichnet.

Ähnlich wie für ungerichtete eulersche Graphen erhält man eine Charakterisierung von eulerschen Digraphen.

Satz 8.3. *Ein gerichteter Graph $G = (V, E)$ ist genau dann eulersch, wenn der zugrunde liegende ungerichtete Graph zusammenhängend ist und wenn $d^+(v) = d^-(v)$ für jeden Knoten $v \in V$ gilt.*



Der Beweis verläuft analog zum Beweis des Satzes von Euler 4.1. Auch der Algorithmus von Hierholzer lässt sich auf den gerichteten Fall anwenden. Welche Tour berechnet der Algorithmus von Hierholzer für den Graphen aus Abb. 8.7?

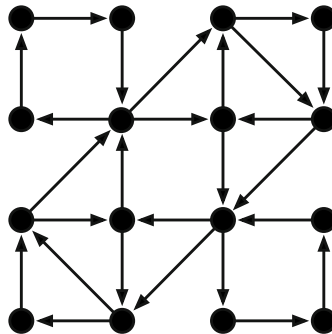


Abb. 8.7: Der Algorithmus von Hierholzer lässt sich auch auf gerichtete Graphen übertragen.

Der folgende, schon etwas kompliziertere Algorithmus konstruiert eine gerichtete Euler-Tour. Dabei müssen wir nicht nach und nach die Tour aus Kreisen zusammensetzen, sondern können die Tour in einem Schritt konstruieren.

Algorithmus 8.2 Gerichtete Euler-Touren

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

Output: Eine gerichtete Euler-Tour.

- Schritt 1: • Sei G' derjenige gerichtete Graph, der durch das Umdrehen der Richtungen aus G entsteht.
 • Wählen Sie einen Knoten $v \in V$ und konstruieren Sie einen spannenden Baum T' von G' mit v als Wurzel.
- Schritt 2: • Sei T der aus T' entstandene Baum mit umgedrehten Kanten: Damit enthält T für jeden Knoten $u \in V \setminus \{v\}$ einen Pfad nach v .
 • Bestimmen Sie eine beliebige Reihenfolge der Nicht-Baumkanten.
- Schritt 3: Konstruieren Sie eine Euler-Tour von v aus, indem Sie an jedem Knoten u die niedrigste unbesuchte Kante — entsprechend der Reihenfolge aus Schritt 2 — und erst dann eine Baumkante wählen.

Als Erstes sollten wir den Algorithmus an einem Beispiel ausprobieren.

Beispiel. In der Abbildung 8.8 sind Input, Schritt 1, Schritt 2 und Schritt 3 dargestellt. Im Schritt 2 werden alle schwarzen Nicht-Baumkanten nummeriert. In Schritt 3 wird dann eine Euler-Tour konstruiert. ■

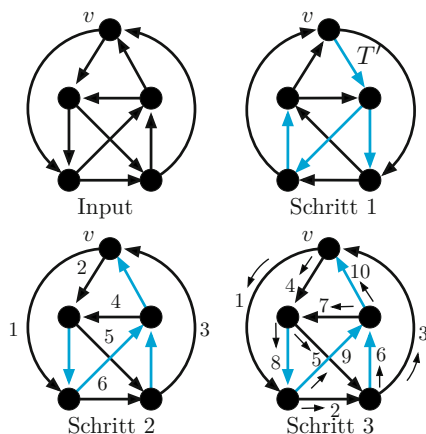


Abb. 8.8: Wir können jetzt eine Euler-Tour konstruieren, indem wir einfach der Nummerierung an den Kanten von Schritt 2 folgen.

Kann der Algorithmus auch bei ungerichteten Graphen angewandt werden? Der Begriff von Euler-Touren lässt sich also einfach auf Digraphen übertragen. Wie sieht es denn mit Euler-Wegen aus? Wie könnte eine gerichtete Version der Definition, der Charakterisierung und eines Algorithmus dafür aussehen?



Wir verlassen nun wieder die eulerschen Graphen und beschäftigen uns mit Turniergraphen.

8.4 Hamilton-Wege in Turniergraphen

In einem klassischen Turniermodus spielt jeder gegen jeden. Danach wird entschieden, wer der Beste ist. Geht man davon aus, dass es in jedem Spiel einen Sieger gibt (d.h. es gibt kein Unentschieden), so kann man das Ergebnis leicht mit Hilfe eines Digraphen darstellen. Jeder Knoten entspricht dabei einem Spieler. Hat der Spieler a gegen den Spieler b gewonnen, so richten wir die Kante von a nach b . Diese Art von Graphen wird auch als Turniergraph bezeichnet.

Definition. Ein vollständiger Graph mit einer Orientierung heißt **Turniergraph**.

Stellen wir uns also einen solchen Graphen vor. Können wir immer eine Reihenfolge der Spieler festlegen, nach der der Erste gegen den Zweiten, der Zweite gegen den Dritten usw. gewinnt? Eine solche Reihenfolge gibt uns zwar noch keine Rangfolge unter den Spielern, weil der erste Spieler durchaus noch gegen alle weiteren verlieren kann, aber es ist eine Möglichkeit, die Spieler zu sortieren. Die verblüffende Antwort auf die Frage nach einer möglichen Reihenfolge lautet, dass eine solche in jedem Turnier existiert. Übertragen auf die Graphentheorie bedeutet dies, dass jeder Turniergraph einen gerichteten Hamilton-Weg enthält.

Definition. Sei G ein gerichteter Graph. Ein gerichteter Weg, der jeden Knoten genau einmal enthält, heißt **Hamilton-Weg**.

In Abschnitt 5.1 hatten wir Hamilton-Kreise in ungerichteten Graphen kennen gelernt. Bisher ist es noch nicht gelungen ist, einen effizienten Algorithmus zu finden, der in einem Graphen einen Hamilton-Kreis berechnet. Ist es jetzt einfacher einen Hamilton-Weg zu bestimmen? Im Allgemeinen lautet die Antwort nein, egal ob ein gerichteter oder ein ungerichteter Graph gegeben ist. Aber in dem speziellen Fall von Turniergraphen ist es leicht möglich.



Satz 8.4. *In jedem Turniergraphen gibt es einen gerichteten Hamilton-Weg.*

Beweis: Den Beweis führen wir mit vollständiger Induktion über die Anzahl der Knoten des Graphen. In einem Turniergraphen mit zwei Knoten ist die einzige Kante des Graphen schon ein Hamilton-Weg. Nehmen wir nun an, dass die Aussage für alle Turniergraphen mit n Knoten gilt. Im Induktionsschritt zeigen wir, dass für einen Turniergraphen G mit $n + 1$ Knoten ein Hamilton-Weg existiert. Dazu löschen wir als Erstes einen beliebigen Knoten w aus dem Graphen. Die restlichen

n Knoten bilden wieder einen Turniergraphen und in diesem existiert nach der Induktionsvoraussetzung ein Hamilton-Weg. Wir bezeichnen ihn mit $p = v_1 v_2 \dots v_n$. Betrachten wir wieder den ursprünglichen Digraphen. Unser Ziel ist es nun den Weg p um den Knoten w zu erweitern. Dazu unterscheiden wir drei Fälle. Die ersten beiden sind einfach: Falls die Kante von w nach v_1 in G enthalten ist, ist wp ein Hamilton-Weg. Ist hingegen die Kante $(v_n, w) \in G$, so ist pw ein Hamilton-Weg. Bleibt noch der letzte und schwierigere Fall.

Hier sind die Kanten (w, v_1) und (v_n, w) nicht in G enthalten (Abb. 8.9). Wir wollen jetzt zeigen, dass ein Knotenpaar v_i, v_{i+1} existiert, sodass wir von v_i nach w über die Kante (v_i, w) und von w nach v_{i+1} über die Kante (w, v_{i+1}) gelangen. Dann haben wir den Knoten w in den vorherigen Hamilton-Weg eingebunden. Aber wieso gibt es ein solches Knotenpaar? Sei i der größte Index, für den die Kante (v_i, w) in G ist. Dann ist aber die Kante (w, v_{i+1}) in G enthalten und wir haben mit v_i, v_{i+1} das gesuchte Knotenpaar gefunden. \square

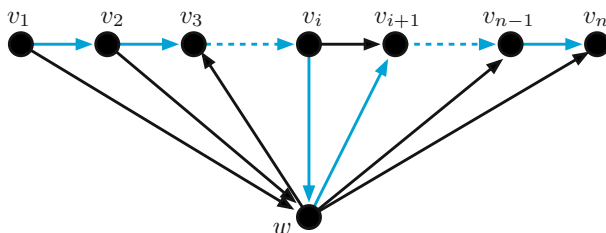


Abb. 8.9: Der Knoten w kann weder vorne noch hinten an den Weg angehängt werden. Aber er kann zwischen die Knoten v_i und v_{i+1} geschoben werden.

Im Beweis konstruiert man — wie gesehen — in jeder Iteration einen Hamilton-Weg. Wie kann man diese Idee in einen Algorithmus umwandeln?



Wir haben jetzt gezeigt, dass immer eine Reihenfolge existiert. Muss diese aber eindeutig sein? Wie die folgenden drei Turniergraphen mit vier Knoten (Abb. 8.10) zeigen, ist dies nicht der Fall: Im ersten existiert nur ein Hamilton-Weg, aber schon im zweiten haben wir zwei unterschiedliche, im dritten kann man schließlich von jedem Knoten aus einen Hamilton-Weg finden.

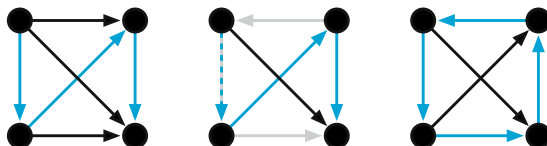


Abb. 8.10: In einem Turniergraphen können beliebig viele Hamilton-Wege existieren.

In welchen Turniergraphen gibt es nur eine einzige Rangfolge? Die Klasse der transitiven Turniergraphen hat zum Beispiel diese Eigenschaft. Ein Turniergraph heißt **transitiv**, wenn aus (a, b) und $(b, c) \in E$ auch $(a, c) \in E$ folgt. Das bedeutet



also, wenn der Spieler a den Spieler b schlägt und der Spieler b den Spieler c , dann muss auch Spieler a gegen Spieler c gewinnen. Der erste Graph aus der Abbildung 8.10 ist transitiv. Sind es die anderen beiden auch?

Satz 8.5. *In transitiven Turniergraphen gibt es nur einen einzigen Hamilton-Weg.*



Beweis: Der Beweis beruht auf der Idee, dass es in einem transitiven Turniergraphen keinen gerichteten Kreis geben kann. Wie lässt sich das beweisen?³⁶

Weiter wissen wir nach Satz 8.4, dass in dem Turniergraphen ein Hamilton-Weg existiert. Wir zeichnen nun den Graphen neu, indem wir zuerst die Knoten entlang des Hamilton-Wegs aufreihen und danach die restlichen Kanten hinzufügen (Abb. 8.11). Da kein gerichteter Kreis entstehen darf, sind alle Kanten von links nach rechts gerichtet. Ein weiterer Hamilton-Weg muss eine dieser hinzugefügten Kanten benutzen. Damit überspringt er irgendeinen Knoten und kann diesen auch später nicht mehr erreichen. Demnach kann kein weiterer Hamilton-Weg existieren und wir haben die Aussage bewiesen. \square

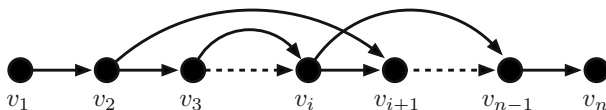


Abb. 8.11: Kanten können nur noch in Richtung des Wegs in den Graphen eingefügt werden.

Turniergraphen, in denen man von jedem Knoten ausgehend einen Hamilton-Weg finden kann, haben genau die gegensätzliche Eigenschaft von transitiven Turniergraphen. Jeder kann hier an erster Stelle stehen. Ein Beispiel hierfür bilden stark zusammenhängende Turniergraphen:

Satz 8.6. *Jeder stark zusammenhängende Turniergraph ist hamiltonsch.*

Wir haben uns bisher nur mit Rangfolgen in Wettkämpfen beschäftigt, dabei aber festgestellt, dass solche in vielen Fällen nicht eindeutig sind. Also ist es wenig sinnvoll, den Gewinner eines Turniers über eine solche Rangfolge festzulegen. Der Mathematiker Landau hat sich in den 50er-Jahren mit einer anderen Methode beschäftigt. Allerdings ging es dabei um Hühner und nicht um Spieler.

8.5 Könige in Turniergraphen

In einem Hühnerstall herrscht unter je zwei Hühnern eine Rangordnung. Das dominante Huhn verteidigt dabei seine Futterplatzansprüche gegen rangniedrigere Tiere mit Schnabelhieben (es hackt das andere Huhn weg) und festigt so seine Stel-

lung. Betrachtet man ein Verhaltensprotokoll, in dem exakt vermerkt ist, welches Huhn in einer Hühnergruppe welches andere Huhn hackt, so stellt sich die Frage nach dem dominantesten und dem schwächsten Huhn. Meistens lässt sich weder ein Huhn ausmachen, das alle anderen Hühner dominiert, noch eines, das von allen anderen durch Schnabelhiebe vertrieben wird. Der Mathematiker Landau hat sich in den 50er-Jahren intensiv mit solchen Hackordnungen auseinandergesetzt.

Was hat das jetzt alles mit der Graphentheorie zu tun? Ganz einfach: Einen Hühnerstall mit Hühnern und ihrem Verhalten können wir wieder als Turniergraphen modellieren. Jedes Huhn ist ein Knoten und die Kante zwischen zwei Hühnern wird vom dominierenden zum weggehackten Huhn gerichtet. Aber welches Huhn ist nun das dominanteste in einem Stall? Natürlich könnte man darunter den seltenen Fall verstehen, dass ein Huhn alle anderen weghackt. Wie können wir den Begriff sinnvoll erweitern? Landau stellt folgende Möglichkeit vor: Ein Huhn ist ein *König*, wenn es alle anderen Hühner entweder direkt oder über ein anderes von ihm dominiertes Huhn weghackt.

Definition. Ein *König* in einem Digraphen ist ein Knoten, der alle anderen Knoten über einen gerichteten Weg mit einer maximalen Länge von zwei erreicht.

Erstaunlich an der Definition ist, dass in einem Turniergraphen immer ein König existiert und der dazugehörige Beweis nicht weiter kompliziert ist.

Satz 8.7 (Hühner-Satz von Landau). *In jedem Turniergraphen existiert ein König.*

Beweis: Sei v ein Knoten in dem Turniergraphen. Wir bezeichnen mit N_v^+ die Knoten, die direkt von v aus erreichbar sind, und mit N_v^- die Menge der Knoten, die v direkt über eine Kante erreichen. Sei x der Knoten mit maximalem $|N_x^+|$ -Wert. Dann behaupten wir, dass x ein König ist (Abb. 8.12).

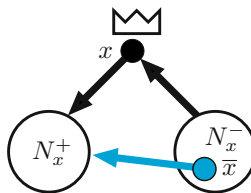


Abb. 8.12: Die dicken Kanten deuten an, dass alle Knoten aus der Menge mit dem einzelnen Knoten verbunden sind. Ein Knoten $\bar{x} \in N_x^-$ kann nicht existieren.

Angenommen, x ist kein König. Dann existiert ein Knoten $\bar{x} \in N_x^-$, der nicht über zwei Kanten mit x verbunden ist. Also kann der Turniergraph für keinen Knoten $w \in N_x^+$ die Kante (w, \bar{x}) enthalten. Folglich zeigen die Kanten aber von \bar{x} zu allen

Knoten $w \in N_x^+$. Somit gilt $|N_x^+| \geq |N_x^+| + |\{x\}|$. Das ergibt einen Widerspruch zu der Maximalität von N_x^+ . \square

Aus dem Beweis lässt sich sofort folgern:

Lemma 8.8. *Sei $G = (V, E)$ ein Turniergraph und $s = \max_{v \in V} \{|N_v^+|\}$. Dann ist jeder Knoten $v \in V$ mit $|N_v^+| = s$ ein König.*



Aber kann ein Knoten nur dann ein König sein, wenn er eine maximale Anzahl an Knoten direkt erreicht? Wie sieht ein Beispiel aus, in dem dies nicht gilt? Wie viele Knoten muss ein König mindestens erreichen?³⁷

Wir wollen uns weiter mit Königen beschäftigen. Um die nächsten Aussagen leichter beweisen zu können, benötigen wir das folgende Hilfslemma.

Lemma 8.9. *Jeder Knoten eines Turniergraphen, der eine eingehende Kante hat, wird auch direkt von einem König erreicht.*

Das Lemma lässt sich in der Sprache der Hühner fast besser verstehen: Jedes Huhn, das von einem anderen gehackt wird, wird auch von einem König gehackt. Das bedeutet noch nicht, dass jedes Huhn, das ein anderes Huhn verjagt, ein König ist.

Beweis von Lemma 8.9. Sei der Ingrad des Knotens v aus dem Graphen G nicht Null. Da die Menge N_v^- wieder einen Turniergraphen bildet, muss mindestens ein König x in N_v^- existieren (Abb. 8.13). Jetzt zeigen wir noch, dass x ein König von G ist. Nach der Wahl von x wird jeder Knoten in N_v^- von x in höchstens zwei Schritten erreicht. Zusätzlich erreicht x den Knoten v in einem Schritt und somit alle Knoten aus N_v^+ in höchstens zwei. Damit ist x auch ein König von G . \square

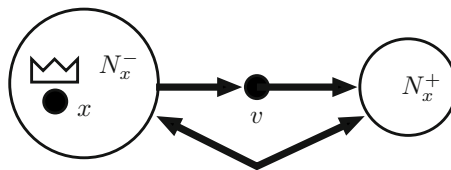


Abb. 8.13: Der Knoten x erreicht jeden Knoten in G über zwei Kanten.

Bisher hatten wir gezeigt, dass in einem Turniergraphen immer ein König existiert. Ein besonderer Fall tritt dann ein, wenn ein Huhn alle anderen Hühner dominiert. Wir nennen einen solchen Knoten einen *Diktator*.

Definition. Ein Knoten v ist ein **Diktator**, wenn er keine eingehenden Kanten hat.

Satz 8.10. *Ein Turniergraph hat genau dann nur einen König, wenn es einen Diktator gibt. Dieser Diktator ist dann auch der einzige König.*

Wie kann diese Aussage bewiesen werden? Beginnen Sie mit der Richtung „Falls es einen Diktator gibt, dann ...“ Die andere Richtung geht über einen Widerspruchsbeweis mit Hilfe von Lemma 8.9.³⁸



Es gibt also *nur* dann genau einen König, wenn eigentlich schon klar ist, dass ein Huhn alle anderen dominiert. Wann hat ein Turniergraph zwei Könige?

Satz 8.11. *Es existiert kein Turniergraph mit genau zwei Königen.*

Beweis: Angenommen, es existiert ein solcher Turniergraph mit den Königen x und \bar{x} . Dann muss \bar{x} von x in höchstens zwei Schritten erreichbar sein. Damit hat \bar{x} eine eingehende Kante. Nach Lemma 8.9 muss in der Menge $N_{\bar{x}}^-$ ein König enthalten sein. Da wir nur zwei Könige haben und \bar{x} nicht in $N_{\bar{x}}^-$ enthalten ist, ist der Knoten x der König in $N_{\bar{x}}^-$. Damit ist die Kante zwischen den Knoten x und \bar{x} von x nach \bar{x} gerichtet.

In umgekehrter Argumentationslinie muss x von \bar{x} erreicht werden. Wie oben erhalten wir, dass die Kante zwischen x und \bar{x} von \bar{x} nach x gerichtet ist. Das ist aber unmöglich. \square

Aus den beiden Aussagen zu einem oder zwei Königen erhält man sofort:

Lemma 8.12. *Jeder Turniergraph mit $n \geq 3$ hat mindestens drei Könige, falls kein Knoten einen Ingrad von Null hat.*

Aber gibt es überhaupt Turniergraphen mit genau drei Königen?

Beispiel. Die Abbildung 8.14 zeigt drei Turniergraphen mit $n = 3, 4, 5$, die jeweils drei Könige haben. Wie sieht jetzt ein Turniergraph mit 6, 10 oder n Knoten aus, der nur drei Könige besitzt? \blacksquare

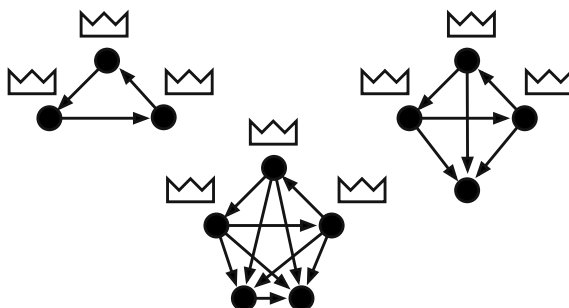


Abb. 8.14: Die drei Knoten mit Kronen sind die einzigen Könige in den Turniergraphen.

Wir wissen jetzt, dass es Turniergraphen mit drei Königen gibt, bleibt noch offen: Existiert vielleicht eine obere Schranke bei der Anzahl der Könige? Es könnte sein, dass es nicht mehr als sieben Könige in einem Turniergraphen geben kann. Leider gibt es eine solche generell festgelegte Obergrenze nicht.

Satz 8.13. *Für jedes $n \in \mathbb{N} \setminus \{2, 4\}$ existiert ein Turniergraph mit n Knoten und n Königen.*

Beweis: Für $n = 1$ stimmt die Aussage und für $n = 2$ hatten wir schon gezeigt, dass kein solcher Turniergraph existieren kann. Für drei Knoten brauchen wir nur den gerichteten Kreis bestehend aus drei Knoten zu betrachten. Erstaunlich ist nun, dass kein Turniergraph mit vier Knoten vier Könige haben kann. Als Beweis könnte man alle möglichen Turniergraphen auf vier Knoten überprüfen. Aber man kann auch die folgende Überlegung anstellen:

Der gesuchte Turniergraph kann keinen Knoten mit Ingrad 0 oder Ausgrad 0 enthalten, sonst hätten wir einen Diktator oder nur drei mögliche Könige. Also muss für jeden Knoten v gelten $1 \leq d^+(v) < 3$. Da im K_4 sechs Kanten existieren, haben zwei Knoten einen Ausgrad von 2 und zwei einen Ausgrad von 1. Damit ist der gesuchte Turniergraph (bis auf die Namensgebung der Knoten) schon eindeutig bestimmt und hat keinen König. Wie sieht er aus?

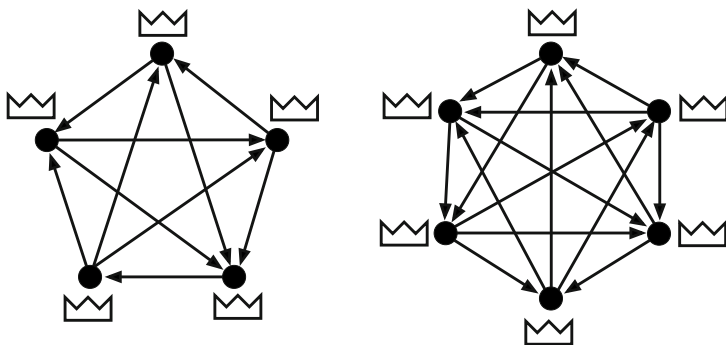


Abb. 8.15: In beiden Turniergraphen sind alle Knoten Könige.

Die Aussage für die restlichen Knoten werden wir über vollständige Induktion zeigen. Für $n = 5$ und $n = 6$ erfüllen die Turniergraphen aus der Abbildung 8.15 die Eigenschaft. Wieso wir den Induktionsanfang für zwei Startwerte wählen, werden Sie gleich sehen.

Wir gehen davon aus, dass die Aussage für $k \leq n$ Knoten korrekt ist und konstruieren jetzt einen Turniergraphen mit $n + 1$ Knoten, der $n + 1$ viele Könige enthält. Dazu betrachten wir als Erstes einen Turniergraphen mit $n - 1$ Knoten, der $n - 1$ Könige hat. Wenn wir nun zwei Knoten x und y so in den Graphen einfügen, dass alle Kanten von x zu den $n - 1$ alten Knoten, alle Kanten von den alten Knoten zu dem Knoten y gerichtet werden und die Kante zwischen x und y von y nach x

orientiert ist, so ist jeder der $n + 1$ Knoten ein König (Abb. 8.16). Jetzt müsste auch klar sein, warum wir für $n \in \{5, 6\}$ die Aussage beweisen mussten. \square

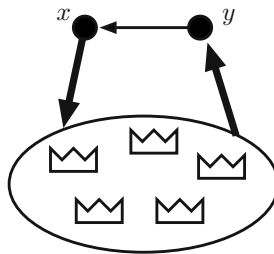


Abb. 8.16: Verbindet man x mit allen Königen, alle Könige mit y und dann y und x , so erhalten wir einen neuen Turniergraphen, in dem jeder Knoten ein König ist.

Abschließend bleibt noch festzustellen, dass sogar für fast jedes $n \in \mathbb{N}$ und $1 \leq k \leq n$ ein Turniergraph mit n Knoten und k Königen existiert. Die einzigen Ausnahmen bilden $k = 2$ und $n = k = 4$. Damit wollen wir die Hühner nicht weiter beim Streit um die Hackordnung stören und beschäftigen uns als Nächstes mit kürzesten Wegen.

8.6 Aufgaben

Aufgabe 8.1. Lassen sich alle Knoten von a aus erreichen (Abb. 8.17)? Wie sieht es mit anderen Startknoten aus? Führen Sie eine Breiten- und Tiefensuche durch.

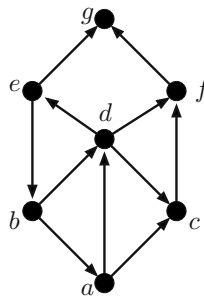


Abb. 8.17: Wie sieht der Breitensuch- bzw. der Tiefensuchbaum in dem Graphen aus?

Aufgabe 8.2. Sind die beiden Graphen (Abb. 8.18) stark zusammenhängend? Wenn nicht, suchen Sie einen gerichteten Schnitt.

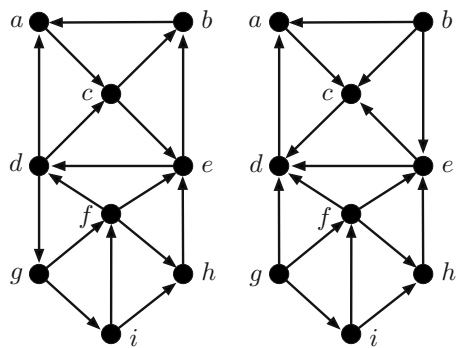


Abb. 8.18: Welche Kriterien erfüllt ein stark zusammenhängender Graph?

Aufgabe 8.3. Finden Sie eine gerichtete Euler-Tour (Abb. 8.19).

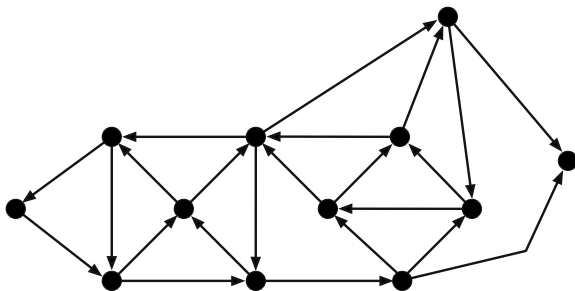


Abb. 8.19: Wieso ist der Graph eulersch?

Aufgabe 8.4. Wie können die Kanten orientiert werden, damit der ungerichtete Graph stark zusammenhängend wird (Abb. 8.20)?

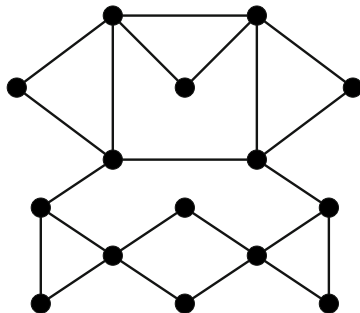


Abb. 8.20: Der zugrunde liegende ungerichtete Graph.

Aufgabe 8.5. Finden Sie einen Hamilton-Weg in dem Turniergraph (Abb. 8.21)?

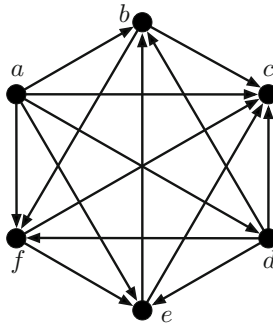


Abb. 8.21: Wieso existiert ein Hamilton-Weg in dem Graphen?

Lösungen zu den Fragen im Text

- 35 Der Graph ist stark zusammenhängend. Damit liegt die Kante e auf einem Kreis. Angenommen, wir haben eine Kante schon gerichtet, d.h. von u nach v . Da wir nach und nach nur Kreise richten, existiert auch schon ein gerichteter Weg von v nach u . Je nachdem, welche „Richtung“ wir in unserem Kreis benötigen, nutzen wir die Kante oder den Weg.
- 36 Mit Hilfe eines Widerspruchs. Angenommen es existiert ein gerichteter Kreis a, b, c, \dots, e . Wegen der Transitivität muss die Kante (a, e) in G enthalten sein. Auf dem Kreis liegt aber die Kante (e, a) . Damit haben wir einen Widerspruch.
- 37 Ein Knoten v muss nur einen Knoten w erreichen, der wiederum mit allen andern Knoten verbunden ist. Dann ist v ein König, erreicht aber nur einen Knoten direkt. Der Knoten w ist dann übrigens auch ein König.
- 38 Falls ein Diktator existiert, kann kein anderer Knoten den Diktator erreichen. Also ist er der einzige König.
Angenommen, es existiert nur ein König und dieser hat eine eingehende Kante. Dann muss nach Lemma 8.9 ein weiterer König existieren. Widerspruch.

9 Kürzeste Wege

Übersicht

| | | |
|-----|--|-----|
| 9.1 | Der Kürzeste-Wege-Baum | 145 |
| 9.2 | Ein Optimalitätskriterium und der Dijkstra-Algorithmus | 149 |
| 9.3 | Negative Kosten | 155 |
| 9.4 | Aufgaben | 158 |

9.1 Der Kürzeste-Wege-Baum

Das Hotel "Zum goldenen Einhorn" in Mittelberg, der Hauptstadt von Schön-Auenland, will durch exklusive Angebote auf seiner Homepage noch mehr Gäste anlocken. Dazu versucht Hans-Jürgen, der Chef des Hauses, eine Auswahl an Tagesausflügen in die umliegenden Orte zu entwickeln. Vor ihm liegt eine Straßenkarte, in die er die Fahrzeiten pro Strecke eingetragen hat (Abb. 9.1). Welche Routen sollte er seinen Touristen vorschlagen, um möglichst schnell in die umliegenden Orte zu gelangen?

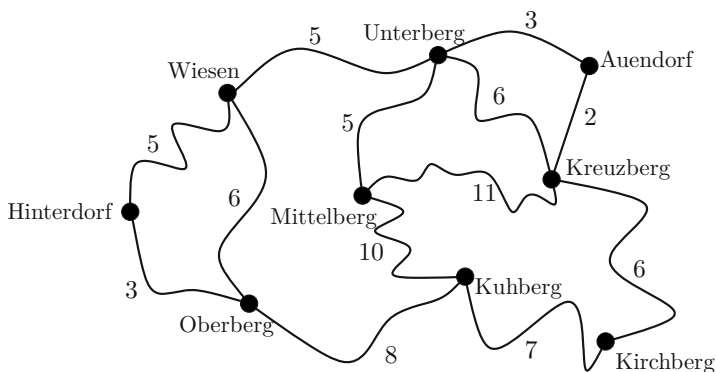


Abb. 9.1: Das Hotel „Zum goldenen Einhorn“ in Mittelberg erfreut sich wegen seiner Lage großer Beliebtheit.

Hans-Jürgen will mit Ausflügen in einen direkt benachbarten Ort beginnen und berechnet als Erstes die Entfernung von Mittelberg nach Kreuzberg. Er kommt auf



11 Kilometer. Finden Sie eine kürzere Strecke?³⁹ Wie sieht es mit der Entfernung zwischen Hinterdorf und Mittelberg aus?

Das Problem, das Hans-Jürgen für jede einzelne Stadt aus dem Kreis Schön-Auenland lösen muss, heißt das kürzeste Wege Problem (engl. *shortest path problem*, SP). Wir definieren hier das Problem für gerichtete Graphen. Dadurch sind auch gleich kürzeste Wege-Probleme mit Einbahnstraßen abgedeckt.

Definition. Sei $G = (V, E)$ ein Digraph (gerichteter Graph) mit einer Kantenbewertung $c(e) \geq 0$ für jede Kante $e \in E$ und seien $s, t \in V$ zwei Knoten (s für engl. *source* und t für engl. *target*). Ein (s, t) -Weg in G heißt **kürzester Weg**, falls sein Gewicht verglichen mit dem Gewicht jedes anderen (s, t) -Wegs minimal ist. Das **Gewicht** $c(p)$ eines Wegs p ist definiert als

$$c(p) = \sum_{e \in p} c(e).$$

Wir beschränken uns hier erstmal auf Graphen mit positiven Kantengewichten. Bei negativen Kantengewichten wird es deutlich schwieriger, einen kürzesten Weg zu bestimmen.

Vielleicht haben Sie sich gefragt, warum in der Definition ein gerichteter Graph vorkommt. In der Straßenkarte (Abb. 9.1) sind auch keine Richtungen vorgegeben. Diese Formulierung hat den einfachen Grund, dass wir die Definition so allgemein wie möglich halten wollen. Hier im Schön-Auenland-Fall kann jede Straße tatsächlich in beide Richtungen benutzt werden. Aber in fast allen Städten gibt es auch Einbahnstraßen. Eine gerichtete Kante modelliert dann diese Einschränkung, wohingegen wir mit einem ungerichteten Graphen dieses Problem nicht lösen könnten. Wie lässt sich die Straßenkarte (Abb. 9.1) in einen gerichteten Graphen umwandeln?⁴⁰

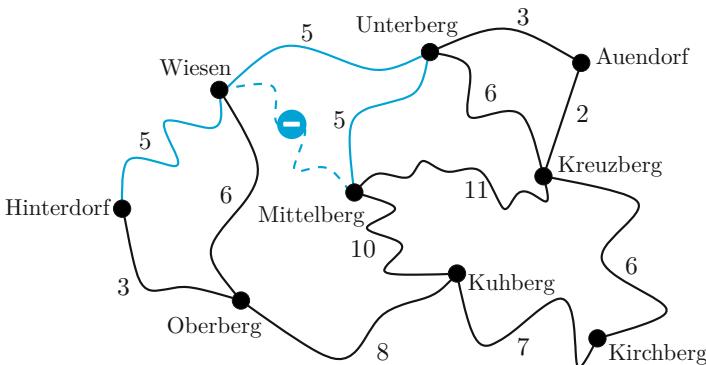


Abb. 9.2: Die blaue Strecke markiert den kürzesten Weg von Mittelberg nach Hinterdorf. Eine kürzere Strecke als die blau eingezeichnete von Mittelberg nach Wiesen, z.B. die blau gepunktete, kann nicht existieren.

Zurück zu Hans-Jürgen und seinen Ausflügen. Mittlerweile hat er die kürzeste Route von Mittelberg nach Hinterdorf berechnet. Diese führt über Wiesen. Muss Hans-Jürgen jetzt noch die kürzeste Strecke von Mittelberg nach Wiesen berechnen? Nein, denn er kann einfach den Abschnitt der Strecke Mittelberg Hinterdorf bis Wiesen nehmen. Wenn dies kein kürzester Weg wäre, dann könnte er auch eine kürzere Strecke von Mittelberg nach Hinterdorf finden (Abb. 9.2).

Diese erste Beobachtung lässt sich leicht in ein erstes Lemma fassen.

Lemma 9.1. *Sei p ein kürzester (s, t) -Weg. Dann ist jeder zusammenhängende Teilweg von p ein kürzester Weg.*

Beweis: Angenommen, es gibt einen Teilweg von p , der kein kürzester Weg ist. Dann existiert ein anderer kürzerer Weg, der die beiden Endknoten des Teilwegs verbindet. Tauschen wir den kürzeren Weg für das bisherige Teilstück ein, erhalten wir einen kürzeren (s, t) -Weg (Abb. 9.3). Das ist aber ein Widerspruch. \square

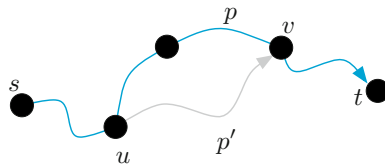


Abb. 9.3: Der Teilweg von p zwischen u und v kann auch durch den Weg p' ersetzt werden.

Im Folgenden werden wir die Notation $p_{(v_i, v_j)}$ für den Teilweg des Wegs $p = v_0 v_1 \dots v_i \dots v_j \dots v_k$ benutzen, der von v_i nach v_j führt.

Hans-Jürgen ist es mittlerweile zu anstrengend geworden, alle kürzesten Wege nach Mittelberg zu berechnen. Deswegen hat er seinem Sohn Klaus diese Aufgabe übertragen. Am nächsten Morgen übergibt Klaus ihm die fertige Lösung. Hans-Jürgen stellt erstaunt fest, dass sie zusammengenommen einen Baum bilden. Eigentlich hatte er nicht damit gerechnet, freut sich aber umso mehr darüber, weil er nun eine sehr übersichtliche Karte ins Netz stellen kann. Hat Hans-Jürgen in dem speziellen Fall Glück gehabt, oder gibt es immer einen Baum, der nur aus kürzesten Wegen besteht? Fallen Ihnen Beispiele ein, in denen kürzeste Wege von einem Knoten s zu den anderen Knoten keinen Baum bilden? Falls ein Baum jedoch nur aus kürzesten Wegen besteht, so wird er auch *Kürzester-Wege-Baum* genannt.



Definition. Sei $G = (V, E)$ ein Digraph und $c(e) \geq 0$ eine Kantenbewertung für jede Kante $e \in E$ und $s \in V$. Ein spannender gerichteter Baum T mit der Wurzel s ist ein **Kürzester-Wege-Baum**, wenn jeder (s, v) -Weg in T ein kürzester Weg in G ist.

Glücklicherweise existiert fast immer ein solcher Baum.

Satz 9.2 (Existenz Kürzester-Wege-Baum). *Sei $G = (V, E)$ ein Digraph mit einer positiven Kantenbewertung. Dann sind die folgenden Bedingungen äquivalent:*

1. Für jeden Knoten $v \in V$ gibt es einen (s, v) -Weg in G .
2. Es existiert ein Kürzester-Wege-Baum mit der Wurzel s in G .



Beweis: $(2) \Rightarrow (1)$: Kann es sein, dass ein Knoten v existiert, zu dem kein (s, v) -Weg in G existiert? Wieso widerspricht dies der Definition eines Kürzesten-Wege-Baums?

$(1) \Rightarrow (2)$: Für jeden Knoten v existiere ein Weg von s nach v . Wir konstruieren nun einen Kürzesten-Wege-Baum T aus diesen Wegen:

Schritt 1: Wählen Sie für jeden Knoten v einen *kürzesten* (s, v) -Weg und fügen Sie diesen in den Teilgraphen $G' = (V, E')$ von G (Abb. 9.4) ein.

Schritt 2: Berechnen Sie in G' einen spannenden Baum T' .

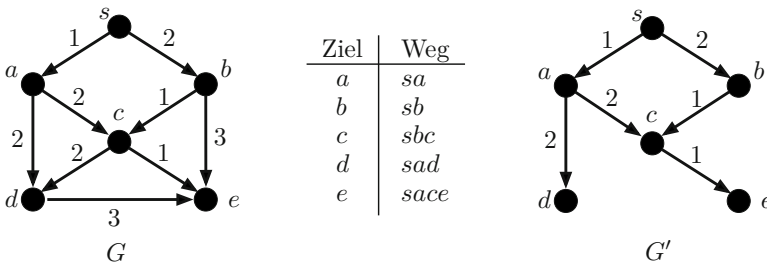


Abb. 9.4: Links ist der Graph G mit seinen Kantengewichten dargestellt. Daneben ist eine Tabelle mit kürzesten Wegen von s zu jedem anderen Knoten als Knotenfolge angegeben. Daraus ergibt sich dann der Graph G' auf der linken Seite. Wie kann ein spannender Baum im Digraphen berechnet werden?

Wir müssen jetzt zeigen, dass T' ein Kürzester-Wege-Baum ist. Angenommen, es existiert ein Weg p in T' von s nach y , der kein kürzester (s, y) -Weg in G ist. Sei dann v der letzte Knoten auf p , für den der Teilweg von s nach v ein kürzester Weg ist. Wir bezeichnen diesen Teil mit $p_{(s,v)}$ (Abb. 9.5). Sei w der nachfolgende Knoten von v auf dem Weg p . Der Weg von s nach w ist also kein kürzester Weg mehr. Da die Kante (v, w) in T' liegt, muss sie auch schon in G' enthalten gewesen sein (T' ist ja ein Teilgraph von G'). Nach der Konstruktion von G' gibt es einen kürzesten Weg \tilde{p} von s nach x , auf dem die Kante (v, w) liegt. Dann ist aber $\tilde{p}_{(s,v)}$, d.h. der Teilweg von \tilde{p} von s nach v , nach Korollar 9.1 auch ein kürzester Weg genauso wie der Teilweg $\tilde{p}_{(s,w)}$. Damit gilt $c(p_{(s,v)}) = c(\tilde{p}_{(s,v)})$ und daraus ergibt sich

$$\begin{aligned} c(p_{(s,v)}) + c((v, w)) &= c(\tilde{p}_{(s,v)}) + c((v, w)) \\ &= c(\tilde{p}_{(s,w)}), \end{aligned}$$

d.h. der Weg $p_{(s,v)}$ zusammen mit der Kante (v,w) bildet wieder einen kürzesten Weg. Das ist aber ein Widerspruch zur Wahl von v . \square

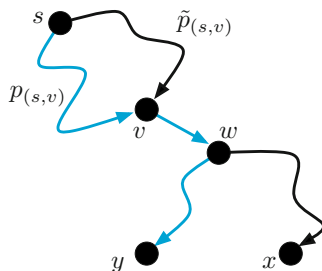


Abb. 9.5: Der Weg p ist blau eingezeichnet. Der Teilweg $\tilde{p}_{(s,v)}$ ist ein kürzester Weg und ist damit genauso lang wie der Weg p' .

Jetzt wissen wir zwar, dass es in den interessanten Fällen einen Kürzesten-Wege-Baum gibt, aber ist es auch einfach möglich einen solchen Baum zu finden? Wie können wir ihn berechnen? Im nächsten Abschnitt beschäftigen wir uns mit diesen beiden Fragen.

9.2 Ein Optimalitätskriterium und der Dijkstra-Algorithmus

Wir haben bereits ganz unterschiedliche Baumprobleme betrachtet. In Kapitel 3.3 sind uns insbesondere zwei Kriterien begegnet, mit denen man einen spannenden Baum auf seine Minimalität überprüfen kann. Ein ähnliches Kriterium suchen wir nun für einen Kürzesten-Wege-Baum. Wie kann Hans-Jürgen die Lösung seines Sohnes leicht überprüfen? Ist der spannende Baum aus der Abbildung (Abb. 9.6) vielleicht ein Kürzester-Wege-Baum?⁴¹

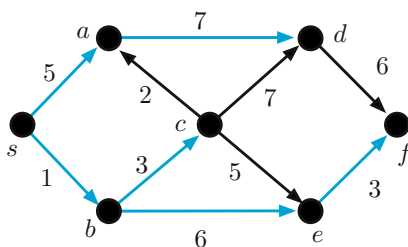


Abb. 9.6: Ist jeder Knoten über einen kürzesten Weg mit s verbunden?

Bevor Ihnen mit Satz 9.3 eine Möglichkeit angeboten wird, den Baum aus der Abbildung 9.6 auf einen kürzesten-Wege-Baum zu überprüfen, benötigen wir noch eine Notation. Im Folgenden bezeichnen wir mit $\text{dist}_T(v)$ bzw. $\text{dist}(v)$ (engl. *di-*



stance) das Gewicht des Wegs von s nach v in dem Baum T . Im obigen Beispiel gilt z.B. $\text{dist}(f) = 10$ oder $\text{dist}(s) = 0$. Wie sieht es mit $\text{dist}(b)$ oder $\text{dist}(d)$ aus?⁴²

Satz 9.3 (Optimalität Kürzester-Wege-Baum). *Sei $G = (V, E)$ ein Digraph mit positiven Kantengewichten $c(e)$ zu jeder Kante $e \in E$ und sei $s \in V$. Ferner sei T ein gerichteter spannender Baum von G . Dann sind die folgenden Aussagen äquivalent:*

1. *Der Baum T ist ein Kürzester-Wege-Baum zum Knoten s .*
2. *Für alle Nicht-Baumkanten $e = (v, w)$ gilt die Dreiecksungleichung*

$$\text{dist}(v) + c(e) \geq \text{dist}(w).$$

Der Name **Dreiecksungleichung** rührt daher, dass man die drei Knoten s , v und w als Dreieck auffasst und dabei beachtet, dass in Dreiecken grundsätzlich die Summe zweier Seiten größer oder gleich der dritten Seite ist (Abb. 9.7).

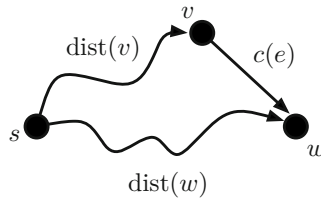


Abb. 9.7: Die Bezeichnung Dreiecksungleichung bezieht sich auf die drei Knoten s , v und w , die mit den dazugehörigen Wegen ein Dreieck bilden.

Beweis zu Satz 9.3. (1) \Rightarrow (2): Sei $e = (v, w) \notin T$ eine Nicht-Baumkante. Der Weg p' von s entlang des Baums zu v und dann von v zu w über die Kante e (Abb. 9.8) hat ein Gewicht von

$$c(p') = \text{dist}(v) + c(e).$$

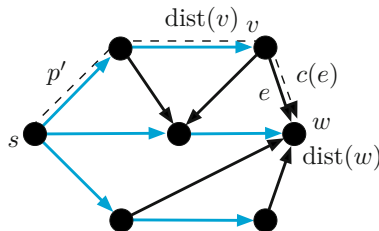


Abb. 9.8: Die Länge des gestrichelten Wegs p' beträgt $c(p') = \text{dist}(v) + c(e)$.

Da T ein Kürzester-Wege-Baum ist, kann das Gewicht von p' nicht kleiner sein als das Gewicht des Wegs von s zu w entlang des Baums T , d.h.

$$\text{dist}(w) \leq c(p') = \text{dist}(v) + c(e).$$

Damit gilt die Dreiecksungleichung und wir sind mit dem Beweis in dieser Richtung fertig.

(2) \Rightarrow (1): Für alle Nicht-Baumkanten gelte nun die Dreiecksungleichung. Wir zeigen, dass T ein Kürzester-Wege-Baum ist, indem wir für jeden in s startenden Weg p zeigen, dass dessen Gewicht größer oder gleich $\text{dist}(v)$ ist, wobei v der Endknoten von p ist. Dazu verwenden wir die Methode der vollständigen Induktion über die Länge der Wege. Mit der Länge ist hier die Anzahl der Kanten gemeint. Damit wir nicht immer von der Länge reden müssen, führen wir noch die folgende Notation ein: Zu einem Weg $p = v_0 v_1 \dots v_r$ sei $\ell(p) = r$ die Anzahl der Kanten auf dem Weg (oder die Länge des Wegs).

Behauptung: Für jeden Knoten $v \in V$ und jeden Weg p von s nach v gilt $c(p) \geq \text{dist}(v)$.

Induktionsanfang: $\ell(p) = 0$, d.h. der Weg p besteht nur aus s . Dann gilt $c(p) = 0 = \text{dist}(s)$.

Induktionsannahme: Für jeden Weg p von s nach v mit $\ell(p) = k$ gilt $c(p) \geq \text{dist}(v)$.

Induktionsschritt: Sei nun p' ein Weg von s nach w mit $\ell(p') = k + 1$. Zu zeigen ist, dass $c(p') \geq \text{dist}(w)$ gilt.

Sei v der Knoten vor w auf dem Weg p' und e die Kante zwischen v und w . Dann ist $\ell(p'_{(s,v)}) = k$ und nach der Induktionsannahme gilt $c(p'_{(s,v)}) \geq \text{dist}(v)$. Falls e eine Nicht-Baumkante ist, gilt nach Aussage des Satzes die Dreiecksungleichung. Ist e hingegen eine Baumkante, so gilt sogar $\text{dist}(v) = \text{dist}(w) + c(e)$ nach der Definition von dist .

Daraus ergibt sich dann

$$\begin{aligned} c(p') &= c(p'_{(s,v)}) + c(e) \\ &\geq \text{dist}(v) + c(e) && \text{(Induktionsannahme)} \\ &\geq \text{dist}(w) && \text{(Dreiecksungleichung)} \end{aligned}$$

Damit haben wir die Induktion abgeschlossen und der Beweis ist fertig. □

Wir haben jetzt ein Kriterium, anhand dessen wir entscheiden können, ob ein spannender Baum ein Kürzester-Wege-Baum ist. Überprüfen Sie noch mal die Dreiecksungleichungen an dem Baum aus der Abbildung 9.6.



Die Berechnung eines Kürzesten-Wege-Baums gelingt mit dem Algorithmus von Dijkstra. Dieser ist benannt nach seinem Erfinder Edsger W. Dijkstra, einem Informatiker aus den Niederlanden. Ursprünglich war der Algorithmus nur zum Testen von Hardware gedacht. Heute ist er einer der Standardalgorithmen zur Bestimmung eines kürzesten Wegs und immer noch sind Mathematiker und Informatiker dabei, den Algorithmus mit allen möglichen Tricks zu beschleunigen.

Der Algorithmus startet in dem Knoten s und sucht dann nach und nach Wege zu den anderen Knoten. Hat er einen Weg zum Knoten v gefunden, welcher allerdings nicht der kürzeste Weg sein muss, so „merkt“ er sich zum einen das bisherige Gewicht dieses Wegs mit dem Wert $\text{dist}(v)$ und zum anderen „speichert“ er, welcher der letzte Knoten $\text{pred}(v)$ (engl. *predecessor*) vor dem Knoten v auf diesem Weg ist. Mit dem jeweiligen Vorgänger können wir später den Weg zu einem beliebigen

Knoten v von v aus nach s zurückverfolgen. Weiter versucht der Algorithmus nun, nicht nur den Wert $\text{dist}(v)$ für jeden Knoten zu verbessern, sondern auch andere Knoten zu erreichen. Wie er dabei vorgeht, schauen wir uns am besten an einem Beispiel an.

Beispiel. Wir betrachten den unten gegebenen Digraphen $G = (V, E)$ mit Kantengewichten $c(e)$ und dem Startknoten s (Abb. 9.9).

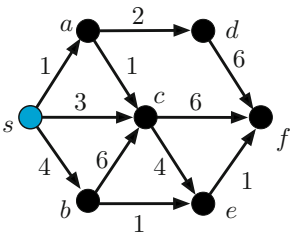


Abb. 9.9: Im vorgegebenen Graphen wollen wir einen Kürzesten-Wege-Baum mit dem Dijkstra-Algorithmus berechnen.

Bisher haben wir noch keinen Weg zu irgendeinem Knoten $v \in V \setminus \{s\}$ ausfindig gemacht. Deswegen setzen wir $\text{dist}(s) = 0$ und $\text{dist}(v) = \infty$ für jeden Knoten $v \in V \setminus \{s\}$. Wenn es noch keine Wege gibt, gibt es auch keine Vorgänger auf diesen Wegen, also gilt auch $\text{pred}(v) = \text{NULL}$ (NULL steht dafür, dass wir noch keinen wirklichen Vorgänger für den Knoten haben). Als Tabelle lässt sich das so zusammenfassen:

| | s | a | b | c | d | e | f |
|------------------|-----|----------|----------|----------|----------|----------|----------|
| $\text{dist}(v)$ | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| $\text{pred}(v)$ | s | NULL | NULL | NULL | NULL | NULL | NULL |

Als Nächstes fangen wir an, alle Knoten von s aus zu erforschen. Die Knoten a, b, c sind direkt von s aus erreichbar. Damit können wir deren dist - und pred -Werte wie folgt verändern:

| | s | a | b | c | d | e | f |
|------------------|-----|-----|-----|-----|----------|----------|----------|
| $\text{dist}(v)$ | 0 | 1 | 4 | 3 | ∞ | ∞ | ∞ |
| $\text{pred}(v)$ | s | s | s | s | NULL | NULL | NULL |

Eine weitere Suche ab s erübrigt sich nun. Deswegen markieren wir s blau (wir sagen auch „ s ist gescannt“) und nehmen den Knoten mit dem kleinsten Abstand zu s . Das ist der Knoten a . Fragt sich nur, ob dieses Vorgehen sinnvoll ist. Machen wir uns den Ansatz noch mal klar: Wir haben den Knoten a direkt von s aus

erreicht. Falls ein kürzerer Weg von s zu a existieren würde, dann müsste der über einen der beiden anderen Knoten b oder c verlaufen. Aber deren Abstand zu s ist ja größer und da wir nur positive Kantengewichte haben, liefert der Umweg keine Verbesserung. Die Wahl macht also Sinn.

Vom Knoten a aus erreichen wir den Knoten d und c . Jetzt müssen wir überprüfen, ob der Weg über a zu c kürzer als der bisherige ist. Weil dem so ist, setzen wir den Vorgänger von c auf a und verändern auch die Distanz von 3 auf 2:

| | s | a | b | c | d | e | f |
|------------------|-----|-----|-----|-----|-----|----------|----------|
| $\text{dist}(v)$ | 0 | 1 | 4 | 2 | 3 | ∞ | ∞ |
| $\text{pred}(v)$ | s | s | s | a | a | NULL | NULL |

Als Nächstes kommt nun der Knoten c an die Reihe. Von ihm aus erreichen wir die Knoten f und e . Zu beiden Knoten hatten wir bisher noch keinen Weg gefunden, sodass für diesen neuen Weg kein Vergleichswert vorliegt.

| | s | a | b | c | d | e | f |
|------------------|-----|-----|-----|-----|-----|-----|-----|
| $\text{dist}(v)$ | 0 | 1 | 4 | 2 | 3 | 6 | 8 |
| $\text{pred}(v)$ | s | s | s | a | a | c | c |

Jetzt geht es um den Knoten d , der nur mit dem Knoten f benachbart ist. Der Abstand zum Knoten s würde sich aber nicht verbessern, wenn man über den Knoten d zu f gelangen würde. Damit ist jetzt der Knoten b dran. Wie sieht die Tabelle danach aus? Verändert sich etwas? Können wir das Verfahren stoppen?

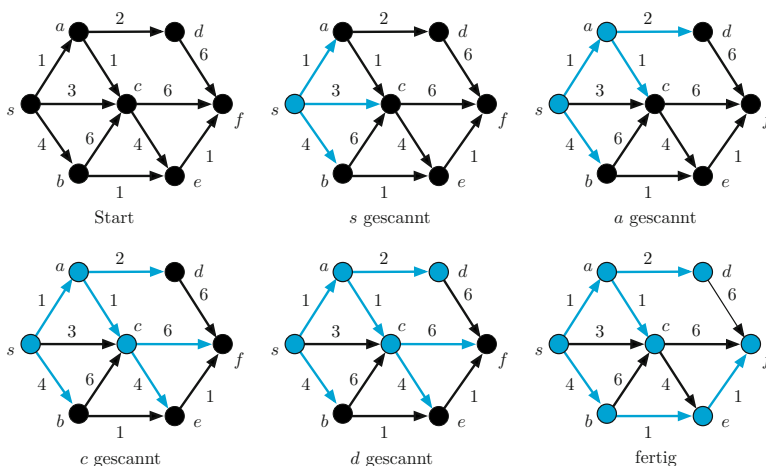


Abb. 9.10: Der Verlauf des Dijkstra-Algorithmus. Die blauen Knoten wurden schon gesamt.

Die Schritte, die wir hier über Tabellen dargestellt haben, kann man auch gut an Graphen veranschaulichen, indem man sich immer die bisherigen Wege markiert. In Abbildung 9.10 sehen Sie den Verlauf des Algorithmus noch einmal. ■

Etwas technischer aufgeschrieben sieht der Algorithmus von Dijkstra dann folgendermaßen aus:

Algorithmus 9.1 Dijkstra-Algorithmus

Input: Digraph $G = (V, E)$ mit positiven Kantengewichten und Knoten s .

Output: Kürzester-Wege-Baum T zu allen von s aus erreichbaren Knoten.

Schritt 1:

- Setzen Sie für jeden Knoten v die vorläufige Distanz $\text{dist}(v) = \infty$.
- Setzen Sie $\text{dist}(s) = 0$.
- Setzen Sie die Vorgänger $\text{pred}(v) = \text{NULL}$ und $\text{pred}(s) = s$.
- Setzen Sie die Menge der ungescanten Knoten $V' = V$.

Schritt 2: Solange ein Knoten $v \in V'$ mit $\text{dist}(v) < \infty$ existiert, wiederholen Sie die folgende Prozedur:

Schritt a: Wählen Sie $v' \in V'$ mit $\text{dist}(v')$ minimal, also $\text{dist}(v') \leq \text{dist}(v) \forall v \in V'$.

Schritt b: Für alle benachbarten Knoten $w \in V'$ von v' testen Sie:
Falls $\text{dist}(v') + c((v', w)) < \text{dist}(w)$, dann

- setzen Sie $\text{dist}(w) = \text{dist}(v') + c((v', w))$
- setzen Sie $\text{pred}(w) = v'$.

Schritt c: Löschen Sie v' aus V' .

Satz 9.4. *Der Dijkstra-Algorithmus berechnet in einem Digraphen mit positiver Kantenbewertung einen Kürzesten-Wege-Baum zu allen Knoten, die von der Wurzel erreichbar sind.*

Wenn wir jetzt aber nur einen kürzesten Weg von Frankfurt nach Berlin bestimmen wollen, müssen wir dann tatsächlich – wie von dem Algorithmus vorgegeben – jeden Knoten der Deutschlandkarte (Abb. 9.11) von Frankfurt ausgehend scannen, bevor wir wissen, dass wir einen kürzesten Weg berechnet haben? Reicht es schon aus, dass wir den Algorithmus stoppen, sobald wir Berlin erreicht haben? Wenn wir uns an das vorige Beispiel erinnern, so haben wir dort den dist -Wert von einigen Knoten auch noch bzw. wieder verändert, nachdem wir sie schon erreicht hatten. Deswegen dürfen wir an der Stelle noch nicht aufhören. Was ist, wenn ein Knoten gescannt wurde? Es wird laut Algorithmus der Knoten dann aus V' gelöscht, wodurch sich sein dist -Wert und auch der bis dahin bestimmte Weg nicht mehr verändert. Damit muss der Weg optimal sein.

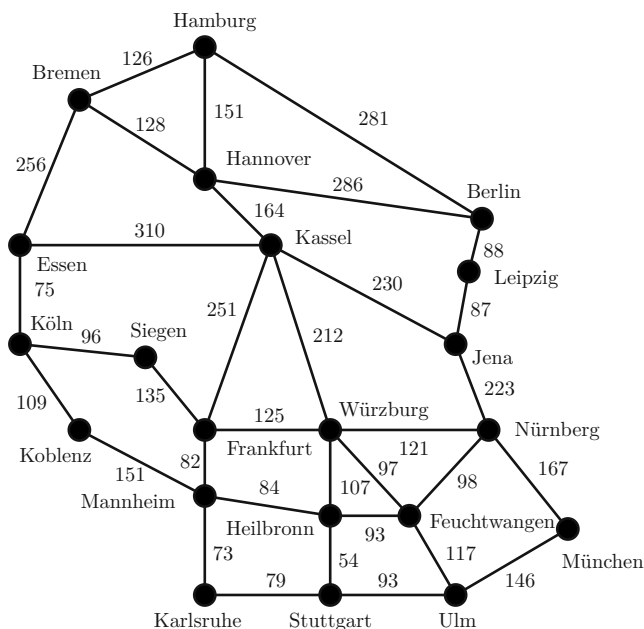


Abb. 9.11: In der Deutschlandkarte sind wieder alle Strecken als ungerichtete Kanten modelliert, da wir sie sowohl in die eine, als auch in die andere Richtung nutzen können. Wie kommen wir jetzt am schnellsten von Frankfurt nach Berlin?



Mit diesen Methoden können wir auch Hotelchef Hans-Jürgen helfen, wenn er nicht nur Ausflüge in das Schön-Auenland vorbereitet und dadurch sein Kartenmaterial etwas aufwändiger wird. Und Sie können sich jetzt überlegen, ob Sie wirklich den kürzesten Weg zur Schule, Arbeit oder Freunden nehmen oder ob es nicht doch noch schneller gehen könnte.

9.3 Negative Kosten

Bisher hatten wir das Kürzeste-Wege-Problem nur mit positiven Kostenfunktionen betrachtet, d.h. $c(e) \geq 0$ für alle $e \in E$. Dies ist auch insbesondere bei der Berechnung von einem kürzesten Weg zwischen zwei Orten sinnvoll. Bei der Berechnung von Wechselkursen hingegen ist es notwendig, auch mit negativen Kosten umgehen zu können.

Eine Bank möchte 100 Dollar für möglichst wenig Euro erwerben. Zum einen kann sie direkt den Preis für einen Dollar in Euro erfragen und dann den Betrag für 100 Dollar bezahlen. Oder sie kann zuerst Dollar für Yen erstehen, anschließend Yen für Lira und dann erst Lira für Euro kaufen. Dabei werden die 100 Dollar mit dem jeweiligen Wechselkurs multipliziert bis die Bank am Ende Euros erhält. Natürlich soll ein möglichst günstiger Kaufweg gefunden werden. Die unterschiedlichen Währungen können nun als Knoten in einem Graphen $G = (V, E)$ und die Preise

für den Kauf einer Währung i von der Währung j als Kosten w_{ij} der Kante (i, j) dargestellt werden. Gesucht ist dann ein Weg p zwischen dem Dollarknoten s und dem Euroknoten t mit geringen Kosten. Die Kosten für p sind gegeben durch

$$w(p) = \prod_{e \in p} w(e).$$

Diese Kostenfunktion bzw. auch die Kantenbewertung lässt sich auf ein Kürzeste-Wege-Problem transformieren, indem man die Kantenbewertung w_{ij} der Kante (i, j) durch $c_{ij} = \log w_{ij}$ ersetzt. Dann gilt $\exp(c_{ij}) = w_{ij}$ und somit

$$w(p) = \prod_{e \in p} w(e) = \exp\left(\sum_{e \in p} c(e)\right) = \exp(c(p)).$$

Daraus folgt insbesondere

$$w(p) \leq w(p') \Leftrightarrow c(p) \leq c(p')$$

für alle (s, t) -Wege p und p' . Allerdings ist jetzt die Positivität der Kosten nicht mehr gewährleistet. Gibt es denn überhaupt einen kürzesten Weg, wenn negative Kosten vorkommen? Und wenn ja, können wir ihn dann auch berechnen? Wie sieht es in den folgenden beiden Graphen G_1 und G_2 aus (Abb. 9.12)?

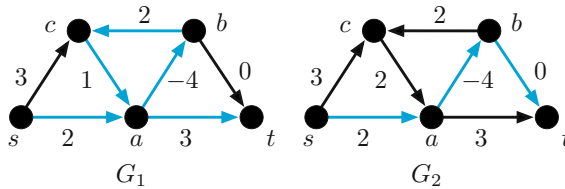


Abb. 9.12: Die Graphen G_1 und G_2 sind dieselben. Allerdings sind zwei Kostenfunktionen mit negativen Gewichten gegeben.

In G_1 hat der Kreis abc Kosten von -1 . Ein kürzester Weg von s nach t sollte also zum Knoten a verlaufen, dann den Kreis abc unendlich oft umrunden und anschließend erst die Kante (a, t) benutzen. Einen wirklichen „kürzesten“ Weg gibt es also nicht. In G_2 hingegen verläuft der kürzeste Weg über die Kante (a, b) und dann von b nach t . Die Kosten hierfür sind -2 . Ob ein kürzester Weg in einem Graphen existiert, hängt also davon ab, ob die gegebene Kostenfunktion einen **negativen Zykel**, d.h. einen gerichteten Kreis mit negativem Gewicht, in dem Graphen erzeugen. Kantenkosten c bezeichnen wir als **konservativ**, falls dies nicht der Fall ist. Für konservative Kantenbewertungen gelten die gleichen Aussagen wie für positive.

Satz 9.5. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$ und c eine konservative Kantenkostenfunktion. Dann gelten folgende Aussagen:

1. Jeder zusammenhängende Teilweg eines kürzesten (s, t) -Wegs ist ebenfalls ein kürzester Weg.

2. *Es existiert ein Kürzester-Wege-Baum mit der Wurzel s in G , falls alle Knoten v von s aus erreichbar sind.*
3. *Ein spannender, gerichteter Baum vom Knoten s aus ist genau dann ein Kürzester-Wege-Baum, wenn für jede Nicht-Baumkante $e = (v, w)$ die Dreiecksungleichung*

$$\text{dist}(v) + c((v, w)) \geq \text{dist}(w)$$

erfüllt ist.

Leider berechnet der Dijkstra-Algorithmus keinen Kürzesten-Wege-Baum bei konservativen Kantengewichten. Wie sieht ein Beispiel aus, bei dem dies nicht der Fall ist?



Wir benötigen also noch einen Algorithmus, der einen solchen Baum berechnet. Außerdem stellt sich die Fragen wie eine konservative Kantenbewertung erkannt werden kann. Beide Fragen wurden von den Mathematikern Ford 1956, Bellman 1958 und Moore 1959 mit dem folgenden Algorithmus beantwortet.

Algorithmus 9.2 Moore-Bellman-Ford-Algorithmus

Input: Digraph $G = (V, E)$ mit Kantenkostenfunktion c und Knoten s .

Output: Kürzester-Wege-Baum T zu allen von s aus erreichbaren Knoten oder ein negativer Zykel.

- Schritt 1:
- Setzen Sie für jeden Knoten v die vorläufige Distanz $\text{dist}(v) = \infty$.
 - Setzen Sie $\text{dist}(s) = 0$.
 - Setzen Sie die Vorgänger $\text{pred}(v) = \text{NULL}$ und $\text{pred}(s) = s$.

- Schritt 2: Wiederholen Sie $n - 1$ -mal die folgende Prozedur:
 Betrachten Sie alle Kanten $e = (v, w) \in E$. Falls $\text{dist}(v) + c(e) < \text{dist}(w)$, dann
- setzen Sie $\text{dist}(w) = \text{dist}(v) + c((v, w))$
 - setzen Sie $\text{pred}(w) = v$.

- Schritt 3: Bilden Sie den Graphen $T = (V, E')$ mit $E' = \{(u, v) \in E \mid u = \text{pred}(v)\}$. Falls T ein Baum ist, ist dieser ein Kürzester-Wege-Baum von s zu allen von s aus erreichbaren Knoten. Andernfalls ist der Kreis in T ein negativer Zykel.
-

Welchen Kürzesten-Wege-Baum berechnet der Moor-Bellman-Ford-Algorithmus in dem Graphen aus Abbildung 9.13 von dem Knoten s aus? Was berechnet der Algorithmus von einem anderen Startknoten aus?



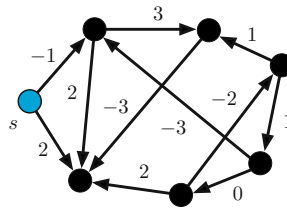


Abb. 9.13: Von s aus soll ein Kürzester-Wege-Baum berechnet werden.

Die Berechnung von kürzesten Wegen bei negativen Kostenfunktionen wird uns in Kapitel 11 wieder begegnen. Davor beschäftigen wir uns noch mit dem Problem, Brandschutzbestimmungen beim Bau eines Einkaufszentrums einzuhalten bzw. Güter in einem Netzwerk zu transportieren.

9.4 Aufgaben

Aufgabe 9.1. Der kleine Frosch Daniel sitzt in der Mitte des Teichs auf einem großen Seerosenblatt. Nicht allzu weit von ihm entfernt zieht schon seit einigen Stunden ein Schwarm von fetten Fliegen seine Runden. Aus Erfahrung weiß er, dass ein großes Blatt ihm viel Sprungkraft gibt und kleinere ihn eher weniger unterstützen. Doch wie soll er den Schwarm erreichen, wenn er nur wahlweise so viele Blätter nach vorne oder nach hinten springen darf, wie auf dem Blatt steht (Abb. 9.14)?



Abb. 9.14: Von seiner Startposition aus wird Daniel erst einmal vier Blätter nach vorne katapultiert. Aber wie geht es weiter?

Angenommen, Daniel braucht für jeden Sprung eine Minute. Wie lange können die Fliegen mindestens noch in Ruhe vor sich hin schwärmen, bevor sie sich in Sicherheit bringen müssen?

Aufgabe 9.2. Teil 1: Klaus möchte von Berlin nach Hamburg fahren und sucht dafür die günstigste Route. Deswegen hat er in seinen Straßenatlas auch schon mal die Benzinkosten für jeden Streckenabschnitt eingetragen (Abb. 9.15). Welchen Weg sollte er wählen?

Klaus ist bei seiner Berechnung von einem Benzinpreis von 0,07 Euro pro Kilometer ausgegangen. Muss er anders fahren, wenn sich die Benzinpreise auf 0,05 Euro pro Kilometer verändert haben?

Seit einigen Jahren ist die Bundesregierung dabei ein Maut-System zu entwerfen. Mittlerweile ist sie erfolgreich bei der Installation des Systems und erhebt für jeden

Streckenabschnitt Extrakosten von 2,00 Euro. Verändert sich die kostengünstigste Route?

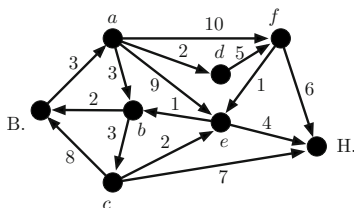


Abb. 9.15: Die Kosten sind in Euro für die Streckenabschnitte angegeben. Dabei geht Klaus von einem Benzinpreis von 0,07 Euro pro Kilometer geht.

Teil 2: Sei $G = (V, E)$ ein Graph, $s, t \in V$, $c : E \rightarrow \mathbb{N}$ eine Kantenkostenfunktion und p ein kürzester (s, t) -Weg bzgl. c . Weiter seien die Kantenkostenfunktionen c_1 und c_2 durch $c_1(e) = \alpha \cdot c(e)$ für alle $e \in E$ und $\alpha > 0$ und $c_2(e) = c(e) + \beta$ für alle $e \in E$ und $\beta > 0$ gegeben. Ist p auch ein kürzester (s, t) -Weg bzgl. c_1 und c_2 ?

Aufgabe 9.3. Berechnen Sie mit Hilfe des Dijkstra-Algorithmus einen Kürzesten-Wege-Baum von a zu allen anderen Knoten in dem ungerichteten Graphen G , der durch die Adjazenzmatrix 9.1 gegeben ist.

| | a | b | c | d | e | f | g | h |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| a | | 11 | 2 | 5 | | | | |
| b | 11 | | 9 | | 3 | 7 | | |
| c | 2 | 9 | | 2 | | 3 | 1 | |
| d | 5 | | 2 | | | | 2 | |
| e | | 3 | | | | 2 | | 1 |
| f | | 7 | 3 | | 2 | | 1 | 1 |
| g | | | 1 | 2 | | 1 | | 3 |
| h | | | | | 1 | 1 | 3 | |

Tab. 9.1: Die Kantengewichte zwischen den Knoten x und y sind durch den Eintrag an der Stelle x, y gegeben, falls die Kante (x, y) in dem Graphen enthalten ist.

Aufgabe 9.4. Teil 1: Anne hat heute viel vor. Gerade hat die Schule aufgehört. Jetzt möchte sie etwas essen (Knoten E in Abb. 9.16), um dann anschließend in eines der Schwimmbäder zu fahren (Knoten S in Abb. 9.16). Nach einigen Bahnen will sie noch zum Friseur (Knoten F in Abb. 9.16) und anschließend Eis (Knoten Eis in Abb. 9.16) für sich und ihre Mutter einkaufen. Welchen Weg sollte sie einschlagen, damit sie möglichst wenig Zeit auf dem Fahrrad verbringt. Die Fahrzeiten in Minuten sind für die einzelnen Streckenabschnitte angegeben.

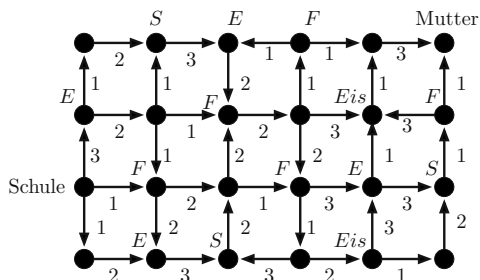


Abb. 9.16: An allen Knoten markiert mit E gibt es etwas zu Essen, S markieren die Schwimmbäder, F die Friseure und Eis die Eisdielen.

Teil 2: Gegeben sei ein Graph $G = (V, E)$, zwei Knoten $s, t \in V$, eine Kantenkostenfunktion $c : E \rightarrow \mathbb{N}$ und k Knotenmengen $V_i \subseteq V$. Gesucht ist ein kürzester (s, t) -Weg, der nacheinander einen Knoten aus V_1 , dann aus V_2 usw. enthält. Zwischen diesen Knoten können auch noch andere Knoten auf dem Weg liegen. Wie kann das Problem als einfaches Kürzestes-Wege-Problem modelliert werden?

Aufgabe 9.5. Der Kettengraph ist folgendermaßen aufgebaut: In einer Reihe von Knoten v_1, \dots, v_n sind immer zwei aufeinander folgende Knoten durch zwei parallele Kanten miteinander verbunden (Abb. 9.17). Wie viele einfache Wege zwischen v_1 und v_n gibt es in einem Kettengraphen, der n Knoten hat?

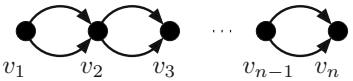


Abb. 9.17: In einem Kettengraphen gibt es sehr viele einfache Wege.

Aufgabe 9.6. Betrachten Sie einen vollständigen (noch ungerichteten) Graphen K_n und nummerieren Sie die Knoten in einer beliebigen Reihenfolge. Anschließend orientieren Sie alle Kanten so, dass sie von der niedrigeren Nummer zur höheren zeigen. Wie viele einfache Wege zwischen v_1 und v_n gibt in diesem Graphen (Abb. 9.18)?

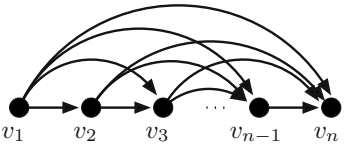


Abb. 9.18: In dem Graphen gibt es keine eingehenden Kanten in den Knoten v_1 . Dafür hat der Knoten v_n keine ausgehenden.

Aufgabe 9.7. Sei $G = (V, E)$ ein gerichteter Graph, $s \in V$ und $c : E \rightarrow \mathbb{N}$ eine Kostenfunktion. Sei e_1 die günstigste Kante in dem Graphen, d.h. $c(e_1) < c(e)$ für alle $e \in E \setminus \{e_1\}$. Zeigen Sie oder widerlegen Sie: Es gibt einen Kürzesten-Wege-Baum von s aus, der die Kante e_1 enthält.

Aufgabe 9.8. Sei $G = (V, E)$ ein Graph, $s \in V$ und $c : E \rightarrow \mathbb{N} \setminus \{0\}$ eine Kantenkostenfunktion. Zeigen Sie:

1. Sei T_{SP} ein Kürzester-Wege-Baum in G von s aus und T_{MST} ein minimal spannender Baum in G , dann haben die beiden Bäume mindestens eine gemeinsame Kante.
2. Für $c : E \rightarrow \mathbb{N} \cup \{0\}$ gilt diese Aussage nicht.

Aufgabe 9.9. Sei $G = (V, E)$ ein gerichteter Graph mit Kantengewichten $0 < w(e) < 1$ für jede Kante $e \in E$ und $s, t \in V$. Das Gewicht $w(e)$ gibt an, wie hoch die Wahrscheinlichkeit ist, dass diese Kante nicht ausfällt. Gesucht ist nun ein (s, t) -Weg p mit maximaler Zuverlässigkeit $w(p)$,

$$w(p) = \prod_{e \in p} w(e).$$

Wie kann dieser Weg berechnet werden?

Aufgabe 9.10. In einigen Anwendungen ist es wichtig, einen eindeutigen kürzesten Weg zu haben. Mit Perturbation, d.h. kleinen Veränderungen der Kantengewichte, kann dies erreicht werden. Formal ist Perturbation der Kostenfunktion folgendermaßen definiert: Sei $G = (V, E)$ ein Graph mit ganzzahligen Kosten $c : E \rightarrow \mathbb{N}$. Sei e_1, \dots, e_m eine beliebige Nummerierung der Kanten. Die perturbierbare Kostenfunktion $c' : E \rightarrow \mathbb{R}$ ist dann definiert durch $c'(e_i) := c(e_i) + (\frac{1}{2})^i$. Zeigen Sie:

1. In G existiert immer ein eindeutiger kürzester Weg bzgl. c' .
2. Ein kürzester Weg bzgl. c' ist ebenfalls ein kürzester Weg bzgl. c .
3. Ein kürzester Weg bzgl. c muss kein kürzester Weg bzgl. c' sein.

Lösungen zu den Fragen im Text

- 39 Ja, über Auendorf ist die Strecke kürzer (10 Einheiten).
- 40 Jede ungerichtete Kante (u, v) kann durch zwei antiparallele Kanten (u, v) und (v, u) ersetzt werden.
- 41 Nein, der Knoten d wird nicht über einen kürzesten Weg erreicht.
- 42 $\text{dist}(b) = 1$, $\text{dist}(d) = 12$

10 Maximale Flüsse

Übersicht

| | | |
|------|--|-----|
| 10.1 | Flüsse und der Dekompositionssatz von Ford-Fulkerson | 163 |
| 10.2 | Das maximale Fluss-Problem | 169 |
| 10.3 | Der Max-Fluss-Min-Schnitt-Satz | 176 |
| 10.4 | Aufgaben | 181 |

10.1 Flüsse und der Dekompositionssatz von Ford-Fulkerson

In einer kleinen Stadt soll ein neues zweistöckiges Einkaufszentrum errichtet werden.

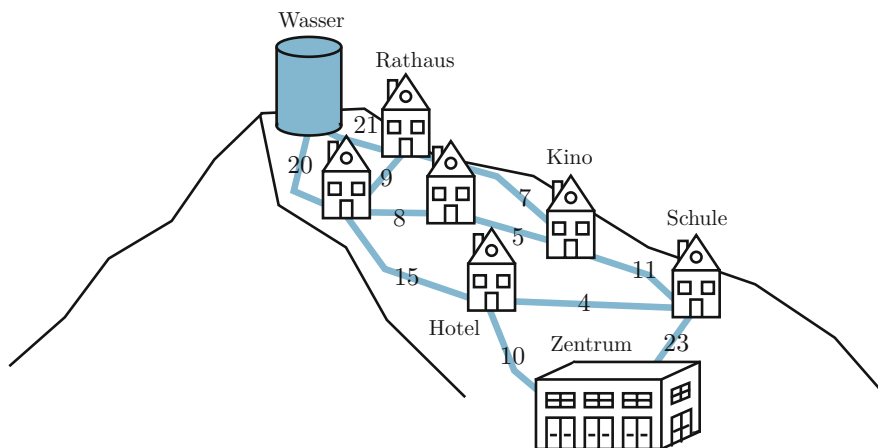


Abb. 10.1: Kann das Einkaufszentrum in der gewünschten Größe gebaut werden? Die Durchflussraten sind mit $100 \text{ l} \cdot \text{min}^{-1}$ angegeben.

Natürlich müssen die Bauherren viele Sicherheitsauflagen erfüllen. Eine davon sorgt dafür, dass im Brandfall genügend Wasser zum Löschen vorhanden ist. Die Feuerwehr hat eine Tabelle erstellt, die angibt, wie viel Wasser pro Minu-

te zur Verfügung stehen muss, um brennende Gebäude effektiv löschen zu können (Tab. 10.1).

| Löschwasserbedarf | Objekt(e) |
|---|--|
| $300 \text{ l} \cdot \text{min}^{-1}$ | Lauben u.ä. |
| $500 \text{ l} \cdot \text{min}^{-1}$ | kleine freistehende Häuser ≤ 2 Geschosse |
| $900 \text{ l} \cdot \text{min}^{-1}$ | Wohngebäude ≤ 3 Geschosse |
| $1100 \text{ l} \cdot \text{min}^{-1}$ | Wohngebäude ≤ 3 Geschosse mit Geschäften oder Gewerbebetriebe. |
| $1700 \text{ l} \cdot \text{min}^{-1}$ | Geschäfts- oder Gewerbegebäude mit ≤ 3 Geschossen |
| $3000 \text{ l} \cdot \text{min}^{-1}$ | Geschäfts- oder Gewerbegebäude mit > 3 Geschossen, Industrie- oder Lagergebäude |
| über $3500 \text{ l} \cdot \text{min}^{-1}$ | Industrie- oder Lagergebäude mit übergroßen Brandabschnitten, Holzlagerplätzen u.ä. bauliche Anlagen |

Tab. 10.1: Löschwasserbedarf für den Objektschutz.

Der Architekt des Unternehmens hat berechnet, dass für das Einkaufszentrum im Notfall genügend Löschwasser vom entfernten Wasserwerk gepumpt werden kann. Hat er Recht? Bei seinen Berechnungen geht er davon aus, dass im Brandfall kein anderer Wasserabnehmer (z.B. das Kino) Wasser aus dem Leitungssystem bezieht. Für seine Berechnung liegen dem Architekten das Wassersystem und die maximale Durchflussrate pro Minute und Rohrabschnitt vor (Abb. 10.1).

Wie kann dieses Problem mathematisch formuliert werden? Das Wassersystem an sich sieht einem Graphen schon recht ähnlich. Die Rohre können wir als Kanten zwischen den Verzweigungspunkten, unseren Knoten, interpretieren. Da das Wasser im Normalfall immer bergab und in einem Rohr nicht in zwei unterschiedliche Richtungen fließen kann, haben wir es hier mit einem gerichteten Graphen zu tun. Des Weiteren soll nun vom Wasserwerk zum Einkaufszentrum Wasser geschickt werden. Anstatt Wasser können wir uns Einheiten vorstellen, die sich entlang unterschiedlicher Wege von einem ausgewählten Knoten s (engl. *source*), hier dem Wasserwerk, zu einem anderen Knoten t (engl. *target*), dem Einkaufszentrum, bewegen. Um dies zu modellieren, bezeichnen wir mit \mathcal{P} die Menge aller einfachen gerichteten Wege, die von s nach t gehen. Wie viele Einheiten nun entlang eines Wegs $p \in \mathcal{P}$ geschickt werden sollen, geben wir mit $x(p)$ an. Eine solche Zuordnung bezeichnet man auch als **Wegfluss**.

Beispiel. Die Wege p_1, p_2 und p_3 bilden die Menge \mathcal{P} mit dem Startknoten s und dem Zielknoten t in dem Graphen G (Abb. 10.2). Jetzt können wir unterschiedliche Wegflüsse konstruieren. Der Wegfluss $x(p_1) = 2$, $x(p_2) = 1$ und $x(p_3) = 3$ (kurz $x = (2, 1, 3)$) schickt entlang des Wegs p_1 genau zwei, entlang des Wegs p_2 nur

noch eine und entlang des Wegs p_3 drei Einheiten. Wie viel fließt dann über die Kanten e_1 oder e_2 ? Wie sieht ein anderer Wegfluss aus? ■

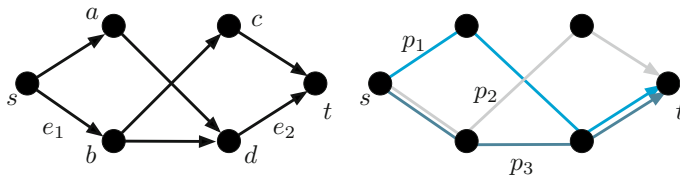


Abb. 10.2: Der Graph G hat drei unterschiedliche Wege, entlang derer Fluss geschickt werden kann.

Zurück zu unserem Einkaufszentrum. Bisher haben wir noch nicht berücksichtigt, dass die Rohre unterschiedliche Dicken haben. Durch manche können wir viel Wasser pro Minute schicken, andere wiederum sind zu schmal, um große Wassermengen weiterleiten zu können. Diese Anforderung fügen wir in unser Modell ein, indem wir den Kanten obere Kapazitäten zuweisen. Die Frage ist nun, ob wir einen Wegfluss finden, der genügend Einheiten transportiert und der an keiner Stelle die oberen Kapazitäten überschreitet. Einen Wegfluss, der die Kapazitäten einhält, nennen wir auch einen **zulässigen** Wegfluss.

Beispiel. Wir haben jetzt den Graphen aus dem letzten Beispiel auf Seite 164 um obere Kapazitäten erweitert. Der Wegfluss $x = (2, 1, 3)$ (wie oben) lässt fünf Einheiten über die Kante e_2 fließen (zwei vom Weg p_1 und drei vom Weg p_3), obwohl nur vier zugelassen sind. Damit hält x die oberen Kapazitäten nicht ein. Wie sieht ein Wegfluss aus, der diese berücksichtigt? Gibt es einen Wegfluss, der insgesamt sechs Einheiten verschickt? ■

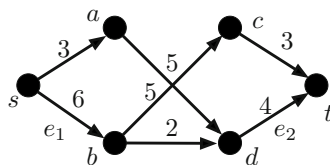


Abb. 10.3: Die Zahlen an den Kanten geben die oberen Kapazitäten an.

Die Anzahl der Einheiten, die ein Wegfluss insgesamt von s nach t verschickt, wird auch als **Flusswert** oder **Wert** bezeichnet.

An sich könnten wir uns jetzt zurücklehnen, da wir eine zulässige Modellierung des Problems gefunden haben. Allerdings würde jeder Praktiker sofort die Hände über dem Kopf zusammenschlagen. Das Problem bei Wegflüssen besteht darin, dass sehr viele (exponentiell viele) gerichtete einfache Wege von s nach t existieren können. In dem gerichteten Graphen in der Abbildung 10.4 gibt es zum Beispiel $2^{\frac{1}{3} \cdot (n-1)}$ unterschiedliche. Zu einem Wegfluss muss dann für jeden Weg, der Fluss



führt, eine Beschreibung des Wegs vorliegen. Das wird relativ komplex, wie man in der Abbildung 10.4 sieht.

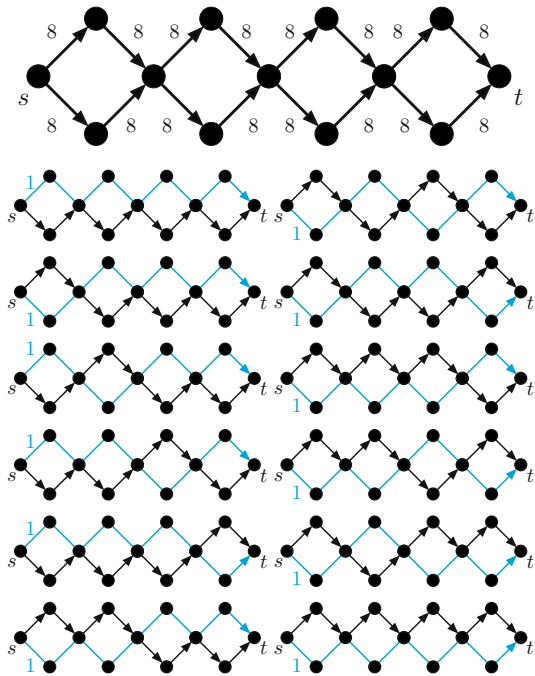


Abb. 10.4: In diesem Graphen existieren schon exponentiell viele Wege von s nach t .

Eine Möglichkeit, das Problem zu umgehen, ist, sich nicht mehr die Wege, sondern nur noch die Belastung auf den einzelnen Kanten zu merken (Abb. 10.5).

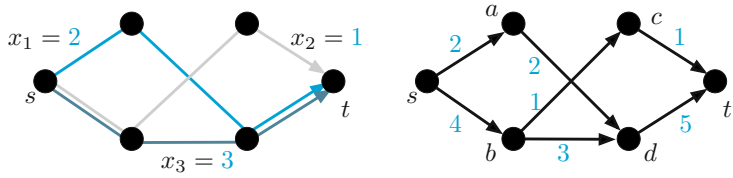


Abb. 10.5: Statt des Wegflusses $x = (2, 1, 3)$ aus dem Beispiel auf Seite 164, merken wir uns die Belastungen auf den einzelnen Kanten.

Ergibt jede mögliche Belastung einen zulässigen Wegfluss, bzw. welche Eigenschaften muss eine Belastung erfüllen? Natürlich darf die Belastung pro Kante nicht die obere Kapazität der Kante überschreiten oder negativ sein. Reicht das schon aus? Noch nicht ganz. Betrachten wir die folgende Belastung der einzelnen Kanten (Abb. 10.6). An dem Knoten b sieht man, dass die eingehenden Kanten insgesamt einen Wert von 4 haben. Die ausgehenden Kanten transportieren jedoch nur 3 Einheiten. Wenn wir uns wieder die Situation mit dem Rohrleitungssystem vorstellen, entsteht an dieser Stelle ein kleiner See. Ähnliches passiert am Knoten a .

Dort sollen pro Minute 5 Einheiten abfließen, jedoch erreichen nur 2 Einheiten pro Minute den Knoten. Damit solche Situationen ausgeschlossen werden, darf an einem Knoten nur genauso viel hinein- wie herausfließen. Eine solche Belastung bezeichnen wir auch als (s, t) -Fluss.

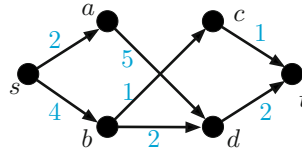


Abb. 10.6: Durch die vorgegebene Belastung auf den Kanten würde am Knoten b ein See entstehen und am Knoten a hätten wir eine wundersame Flussvermehrung.

Definition. Sei $G = (V, E)$ ein gerichteter Graph mit oberen Kapazitäten $u(e)$ für jede Kante $e \in E$ und zwei markierte Knoten $s, t \in V$ (kurz: **Netzwerk** (G, u, s, t)). Ein (s, t) -**Fluss** ist eine Kantenbewertung $f : E \rightarrow \mathbb{R}_{\geq 0}$, die die Kantenkapazitäten einhält und für die an jedem Knoten $v \in V \setminus \{s, t\}$ Flusserhaltung gilt. Ein (s, t) -Fluss f hält die **Kantenkapazitäten** ein, wenn

$$f(e) \leq u(e) \quad \forall e \in E.$$

An einem Knoten $v \in V$ liegt **Flusserhaltung** vor, wenn

$$\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = 0.$$

Mit $\delta^-(v)$ werden die eingehenden Kanten in v bezeichnet und mit $\delta^+(v)$ die ausgehenden Kanten. Der **Wert** (engl. *value*) eines (s, t) -Flusses ist definiert durch

$$\text{value}(f) = \sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e).$$

Die Definition hat nichts mehr mit Wegen zu tun, sondern weist jeder Kante einen bestimmten Wert zu. Der Wert von f soll nun angeben, wie viel der Fluss von s nach t transportiert. Aber stimmt das überhaupt? Können wir zu einem beliebigen Fluss eine Menge von einfachen (s, t) -Wegen finden, sodass sich der Fluss in diese Wege zerlegen lässt? Für den Graphen G in Abb. 10.7 finden wir keine Zerlegung in (s, t) -Wege, dafür aber eine in (s, t) -Wege und gerichtete Kreise.

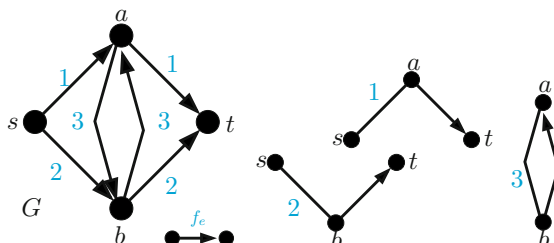


Abb. 10.7: Der Fluss kann in zwei Wege und einen Kreis zerlegt werden.

Ford und Fulkerson, die Väter der Flusstheorie, haben 1962 gezeigt, dass eine solche Zerlegung von jedem (s, t) -Fluss möglich ist.

Satz 10.1 (Dekompositionssatz für (s, t) -Flüsse, Ford-Fulkerson 1962). *Sei f ein (s, t) -Fluss in dem Netzwerk (G, u, s, t) . Dann gilt:*

- *Der Fluss f lässt sich in gerichtete (s, t) -Wege und gerichtete Kreise zerlegen.*
- *Die Anzahl der Wege und Kreise kann auf höchstens m beschränkt werden.*

Beweis: Für den Beweis verwenden wir vollständige Induktion nach der Anzahl der Kanten m' mit positivem Flusswert $f(e) > 0$. Für $m' = 0$, d.h. es existiert keine Kante mit positivem Flusswert, stimmt die Behauptung. Sei also $m' > 0$.

Die Beweisidee ist die Folgende: Als Erstes konstruieren wir einen (s, t) -Weg oder einen Kreis, der nur Kanten mit positiven Flussaufkommen enthält (Teil 1). Anschließend löschen wir entlang dieses Wegs so viel Fluss, dass es mindestens auf einer Kante kein Flussaufkommen mehr gibt (Teil 2). Für den so entstandenen Fluss gibt es nach der Induktionsvoraussetzung eine Flusszerlegung. Nimmt man den konstruierten Weg mit dem gelöschten Flussaufkommen zu der Zerlegung dazu, erhält man wieder eine Flusszerlegung von f .

Teil 1: Die Konstruktion eines (s, t) -Wegs oder Kreises mit positiven Flussaufkommen. Dazu wählen wir zuerst eine Kante (v_0, w_0) mit positivem Flussaufkommen und bestimmen nach und nach einen Weg mit positivem Fluss von w_0 nach t oder einen Kreis. Sei also $e = (v_0, w_0)$ eine Kante mit $f(e) > 0$. Wegen der Fluss-erhaltung existiert eine Kante (w_0, w_1) mit positivem Flussaufkommen oder es gilt $w_0 = t$. Im ersten Fall existiert wieder eine Kante (w_1, w_2) mit positivem Flussaufkommen oder es gilt $w_1 = t$. Wir können nun so lange weitermachen, bis wir bei t ankommen oder einen Knoten erreichen, den wir schon vorher betrachtet haben. Wir erhalten somit eine Folge $\{v_0, w_0, w_1, \dots, w_k\}$ benachbarter Knoten mit positivem Fluss und $w_k = t$ oder $w_k \in \{v_0, w_0, w_1, \dots, w_{k-1}\}$ (Abb. 10.8).

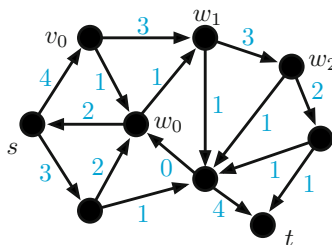


Abb. 10.8: Welcher Knoten kann als w_3 bzw. w_4 gewählt werden?

Falls wir bei t angekommen sind, suchen wir nun entweder einen (s, v_0) -Weg mit positivem Flussaufkommen oder einen Kreis. Dazu gehen wir auf ähnliche Art vor: Gilt $v_0 \neq s$, dann existiert eine Kante (v_1, v_0) mit positivem Flusswert. Nun ist $v_1 = s$ oder es existiert eine Kante (v_2, v_1) mit positivem Fluss u.s.w. Auch

hier hören wir auf, wenn wir bei s oder bei einem schon besuchten v_i oder w_i angekommen sind.

Auf diese Weise erhalten wir in jedem Fall einen Weg von s nach t oder einen Kreis, der nur Kanten mit positivem Flusswert enthält. Wir bezeichnen beide mit p .

Teil 2: Definition eines neuen Flusses. Mit Hilfe von p definieren wir nun den Fluss f' . Dazu sei $\mu = \min_{e \in E(p)} f(e)$ der maximale Fluss, der über p geschickt werden kann. Daraus ergibt sich ein neuer (s, t) -Fluss f' mit

$$f'(e) = \begin{cases} f(e) & \text{falls } e \notin p \\ f(e) - \mu & \text{falls } e \in p. \end{cases}$$

Die Anzahl der Kanten mit positiven Flusswert von f' beträgt maximal noch $m' - 1$. Für f' gibt es nun eine Zerlegung in Kreise und (s, t) -Wege. Dieser Zerlegung zusammen mit p ergibt dann eine Flusszerlegung von f . Damit ist der Satz bewiesen. \square

Der Beweis sagt uns nicht nur, dass ein (s, t) -Fluss eine Antwort auf die Feuer-schutzfrage des Einkaufszentrums ist. Er stellt uns auch eine Anleitung zur Verfügung, wie wir zu einem gegebenen Fluss eine Zerlegung in gerichtete Kreise und gerichtete (s, t) -Wege finden. Sobald wir diese haben, können wir einen Wegfluss mit demselben Flusswert konstruieren. Wie sieht eine Umwandlung des Flusses f aus Abb. 10.8 in einen Wegfluss aus?



Wir haben jetzt also eine gute Formulierung unseres Problems. Gelöst haben wir es deswegen aber noch nicht, d.h. wir wissen noch nicht, ob wir die feuertechnischen Vorgaben einhalten.

10.2 Das maximale Fluss-Problem

Auch ohne mathematische Lösung wurde das Einkaufszentrum unter Beachtung der Sicherheitsvorschriften gebaut. Da das Einkaufszentrum in der Stadt gut angenommen wird, ist schon ein Anbau geplant. Der Architekt ist damit beauftragt festzustellen, um wie viele Stockwerke das Gebäude unter dem Aspekt des Brand-schutzes erhöht werden kann. Er muss also wissen, wie viel Wasser ihm maximal zur Verfügung steht. Die Fragestellung hat sich leicht geändert. Statt eines (s, t) -Flusses mit einem bestimmten Wert suchen wir nun einen (s, t) -Fluss mit maximalem Wert. Wie kann ein solcher Fluss berechnet werden?

Ein erster intuitiver Ansatz ist, sich einen Weg von s nach t zu suchen und solange dort Fluss entlang zu schicken, bis eine Kante keinen Fluss mehr aufnehmen kann. Zum Beispiel können wir in Abbildung 10.9 damit starten, entlang der oberen Kanten Fluss zu schicken (Abb. 10.9 (a)). Die oberen Kapazitäten sind hier schwarz eingezeichnet und der Fluss blau. Nachdem wir den Fluss geschickt haben, suchen wir einfach den nächsten Weg und wiederholen dasselbe. In unserem

Beispiel könnten wir jetzt den Weg $p_1 = svwt$ wählen (Abb. 10.9 (b)). Ist auch hier Fluss vom Wert 2 verschickt, finden wir keinen Weg mehr von s nach t , der noch Kapazitäten zur Verfügung hat. Der Weg p_1 hat die beiden Wege $p_2 = svt$ und $p_3 = swt$ blockiert (Abb. 10.9 (c)). Allerdings ist der (s, t) -Fluss nicht maximal. Hätten wir statt des Wegs p_1 den Weg p_2 und dann den Weg p_3 gewählt, wären wir optimal. Wie können wir die Idee retten, entlang Wegen Fluss zu verschicken?

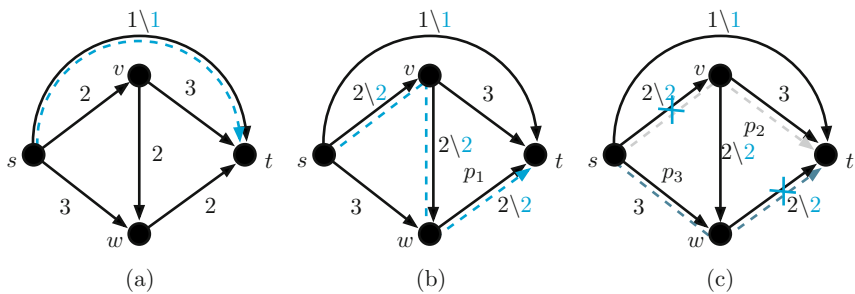


Abb. 10.9: Wir schicken als Erstes Fluss entlang des (s, t) -Wegs $p = st$ mit einem Wert von 1 (Teil (a)). Als Nächstes schicken wir 2 Einheiten entlang des Wegs p_1 , der dann das Verschicken von mehr Fluss verhindert (Teil (b) und (c)).

Unsere Auswahl der Wege war beliebig. Daraus folgt die Problematik, dass ein Weg Kanten blockiert, die noch besser zu nutzen wären. Wenn wir uns jetzt die Option offen halten, geschickten Fluss auch wieder zurückzuschicken bzw. umzuleiten, dann könnten Blockaden auch wieder abgebaut werden. Schauen wir uns das noch mal an den Graphen und dem Fluss von oben an.

Beispiel. Wir haben also den Fluss f gegeben, der aus dem Schicken von Fluss entlang der Wege $p = st$ und $p_1 = svwt$ entsteht (Abb. 10.10 (a)). In dem Graphen sind jetzt die Kanten (s, t) , (s, v) , (v, w) und (w, t) durch den Fluss blockiert, was in der Abbildung 10.10 (b) grau markiert ist.

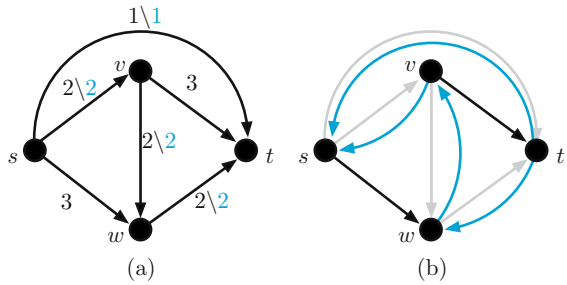


Abb. 10.10: In (a) ist der Fluss f aus Abb. 10.9 (b) noch mal gezeigt. Der Fluss blockiert jetzt die in (b) grau gefärbten Kanten. Dafür können wir entlang der blauen Kanten Fluss zurückschicken.

Wenn wir uns jetzt umentscheiden wollen, so können wir das als das Zurückpumpen des Flusses interpretieren. Um dieses Zurückpumpen in unser Modell zu

übertragen, fügen wir für jede flussführende Kante (x, y) die antiparallele Kante (y, x) in den Graphen ein. Sie sind hier blau eingezeichnet. Was passiert, wenn wir jetzt nach einem (s, t) -Weg suchen, der nur schwarze und blaue Kanten benutzt (die grauen sind ja schon „voll“)? Was bedeutet es, entlang dieses Wegs Fluss zu schicken? Die Fragen werden wir gleich noch genauer betrachten. Zuerst wollen wir dieses „Zurückpumpen“ etwas formalisieren. ■



Die Definition des Residualgraphen spiegelt die Idee der Pumpkanten. Sie ist zwar relativ lang, hört sich aber komplizierter an, als sie tatsächlich ist.

Definition. Sei $G = (V, E)$ ein Graph. Mit $e = (u, v) \in E$ bezeichnen wir die **Vorwärtskanten** und mit $\overleftarrow{e} = (v, u)$ die **Rückwärtskante** von e . Der Graph G^{\leftrightarrow} enthält alle Kanten $e \in E$ und alle Rückwärtskanten. Sei nun (G, u, s, t) ein Netzwerk und f ein (s, t) -Fluss. Die **Residualkapazität** u^f ist definiert für alle Kanten in G^{\leftrightarrow} durch

$$\begin{aligned} u^f(e) &= u(e) - f(e) \quad \text{für alle } e \in E \\ u^f(\overleftarrow{e}) &= f(e) \quad \text{für alle } e \in E. \end{aligned}$$

Der **Residualgraph** G^f zu dem Netzwerk (G, u, s, t) und dem Fluss f ist ein Teilgraph von G^{\leftrightarrow} , der alle Kanten enthält, deren Residualkapazität nicht Null ist.

Gehen wir die Definition noch mal langsam an unserem Beispiel durch.

Beispiel. Der Graph G^{\leftrightarrow} enthält alle Kanten von G und ihre Rückwärtskanten (hier blau eingetragen) (Abb. 10.11 (b)). Bisher hatten wir den (s, t) -Fluss f berechnet. Der Residualgraph G^f enthält nun alle Kanten von G^{\leftrightarrow} , deren Residualkapazitäten nicht Null sind (Abb. 10.11 (c)). ■

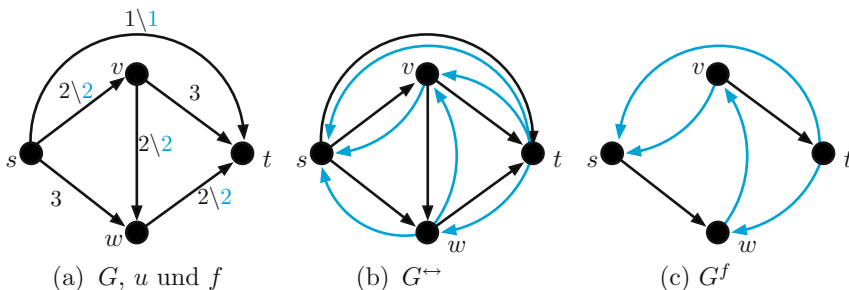


Abb. 10.11: Der Residualgraph enthält alle Kanten, über die noch Fluss geschickt werden kann, und die „Pumpkanten“.

Jetzt haben wir zwar diese komplizierte Formulierung, aber bringt uns das weiter? Suchen wir uns wieder einen (s, t) -Weg p in G^f . Dann können wir auch hier

wieder Einheiten entlang der Wege schicken (natürlich ohne die Residualkapazitäten der einzelnen Kanten zu überschreiten). Dabei verändern wir den Fluss f folgendermaßen: Auf jeder Vorwärtskante von p wird der Flusswert erhöht und auf jeder Rückwärtskante der Fluss erniedrigt (also quasi wieder zurückgeschickt) (Abb. 10.12).

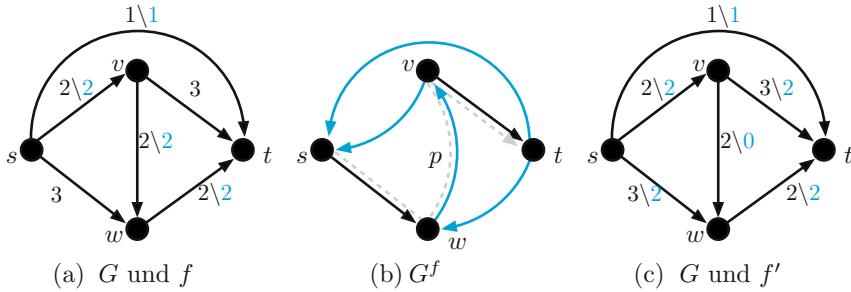


Abb. 10.12: Verändert man den Fluss entlang des Wegs p um den Wert 2, so erhält man den neuen Fluss f' .

Das Resultat dieser Umleitungaktion ist in diesem Beispiel wieder ein Fluss, der zwei Einheiten mehr verschickt als der ursprüngliche Fluss. Der folgende Satz besagt nun, dass die neue Kantenbewertung immer ein Fluss ist, dessen Flusswert höher ist als der des alten.

Satz 10.2. *Sei p ein einfacher (s,t) -Weg in G^f des Netzwerks (G, u, s, t) und des Flusses f und es gelte $\gamma \leq \min_{e \in p} u^f(e)$. Dann ist*

$$f'(e) := \begin{cases} f(e) + \gamma & \text{für alle Kanten } e \in p \\ f(e) - \gamma & \text{für alle Kanten } \overleftarrow{e} \in p \\ f(e) & \text{sonst} \end{cases}$$

ein (s,t) -Fluss in (G, u, s, t) . Für den Wert von f' gilt

$$\text{value}(f') = \text{value}(f) + \gamma.$$

Beweis: Als Erstes wollen wir überprüfen, ob f' die oberen Kapazitäten einhält und positiv bleibt. Sei e eine Vorwärtskante und in p enthalten. Dann gilt

$$0 \leq f(e) \leq f'(e) = f(e) + \gamma \leq f(e) + u^f(e) = f(e) + u(e) - f(e) = u(e).$$

Sei nun die Rückwärtskante \overleftarrow{e} in p . Dann gilt nach der Definition von $f'(e)$

$$u(e) \geq f(e) \geq f'(e) = f(e) - \gamma \geq f(e) - u^f(e) = f(e) - f(e) = 0.$$

Testen wir also, ob f' an jedem Knoten außer s und t die Flusserhaltung erfüllt ist. Da für Knoten, die nicht auf p liegen, keine Veränderung eintritt, können wir uns auf die Pfadknoten konzentrieren. Dazu folgende Überlegung: Für Pfadknoten, ausgenommen s und t , existiert eine Kante $e' = (w, v) \in p$, die in den Knoten reinführt, und eine Kante $e'' = (v, u) \in p$, die aus dem Knoten herausführt (Abb. 10.13). Ansonsten wäre p kein Weg. Jetzt können nur die folgenden vier Fälle auftreten: e', e'' sind Vorwärtskanten (Vk), e', e'' sind Rückwärtskanten (Rk), e' ist eine Rückwärtskante und e'' ist eine Vorwärtskante oder e' ist eine Vorwärtskante und e'' ist eine Rückwärtskante. Dass f' die Flusserhaltung erfüllt, sieht man, wenn man den Einfluss und den Ausfluss für die vier Fälle betrachtet (Abb. 10.13).

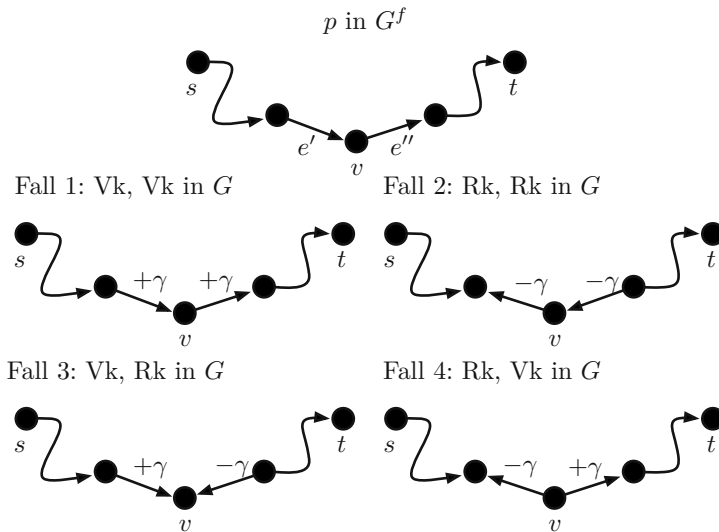


Abb. 10.13: Egal ob e' oder e'' Vorwärts- oder Rückwärtskanten (Vk oder Rk) sind, die neue Kantenbewertung erfüllt die Flusserhaltung.

Bleibt zu zeigen, dass sich auch der Flusswert verändert. Sei nun e_1 die erste Kante von p und eine Vorwärtskante. Dann wird der Ausfluss an dieser Kante um den Wert γ erhöht. Ist nun e_1 eine Rückwärtskante, dann verringert sich der Einfluss in s . In beiden Fällen gilt

$$\text{value}(f') = \text{value}(f) + \gamma.$$

Damit haben wir auch diesen Teil bewiesen. □

Nach Aussage des Satzes lohnt sich eine Flussveränderung entlang eines (s, t) -Wegs in G^f auf jeden Fall. Aus diesem Grund bezeichnet man einen solchen Weg auch als **f -augmentierenden Weg**. Um nicht in grauer Theorie zu verharren, nehmen wir uns doch ein weiteres Beispiel vor.



Beispiel. Betrachte den Graphen G mit den oberen Kapazitäten u und den Fluss f in Abbildung (10.14). Die obere Kapazität der Kante steht an erster Stelle, der Flusswert an der zweiten. Wie sieht der Residualgraph G^f zu dem Graphen G und dem Fluss f aus? Finden Sie einen f -augmentierenden Weg p ? Welcher Fluss entsteht, wenn f entlang p verändert wird? Wie häufig kann dieser Prozess mit dem jeweils neuen Fluss wiederholt werden? Welcher Flusswert ergibt sich?⁴³ ■

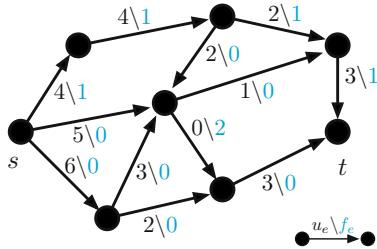


Abb. 10.14: Kantenbeschriftungen $u(e) \setminus f(e)$ geben als erste Stelle die obere Kapazität $u(e)$ der Kante und als zweite den Flusswert $f(e)$.

Unser bisheriges Vorgehen entspricht dem Algorithmus von Ford-Fulkerson für maximale Flüsse.

Algorithmus 10.1 Ford-Fulkerson für Maximale Flüsse

Input: Netzwerk (G, u, s, t) .

Output: Maximaler Fluss f .

Schritt 1: Setzen Sie $f(e) = 0$ für alle Kanten $e \in E$.

Schritt 2: Bestimmen Sie G^f und $u^f(e)$.

Schritt 3: Konstruieren Sie einen einfachen (s, t) -Weg p in G^f . Falls keiner existiert: STOPP.

Schritt 4: Verändern Sie den Fluss f entlang des Wegs p um $\gamma := \min_{e \in p} u^f(e)$.

Schritt 5: Gehen Sie zu Schritt 2.

Der Algorithmus legt nicht fest, welcher augmentierende Weg in jedem Durchlauf verwendet werden soll. An der Optimalität des Algorithmus ändert dies nichts, wie wir im nächsten Abschnitt sehen werden. Allerdings kann die Anzahl der *Iterationen*, d.h. wie häufig augmentiert wird, von dem größten Wert der oberen Kapazitäten abhängen. Wie viele Iterationen werden im ungünstigsten Fall in dem Netzwerk aus der Abbildung 10.15 benötigt, bis der Algorithmus terminiert?⁴⁴ Wie muss das Beispiel verändert werden, dass der Algorithmus $2N$ Iterationen durchläuft, wobei N eine beliebige natürliche Zahl ist?



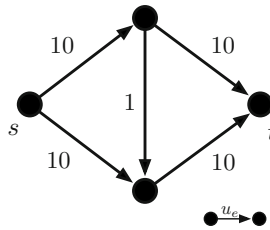


Abb. 10.15: Der Graph wird auch als Diamantgraph bezeichnet.

Eine einfach Idee von Edmonds und Karp im Jahr 1972 hat das Problem der vielen Iterationen gelöst: Anstatt einen beliebigen f -augmentierenden Weg im Residualgraphen zu suchen, wählen sie den bezüglich der Anzahl der Kanten kürzesten augmentierenden Weg.

Algorithmus 10.2 Edmonds-Karp für Maximale Flüsse

Input: Netzwerk (G, u, s, t) .

Output: Maximaler Fluss f .

Schritt 1: Setzen Sie $f(e) = 0$ für alle Kanten $e \in E$.

Schritt 2: Bestimmen Sie G^f und $u^f(e)$.

Schritt 3: Konstruieren Sie einen **kürzesten** (s, t) -Weg p bzgl. der **Anzahl der Kanten in G^f** . Falls keiner existiert: STOPP.

Schritt 4: Verändern Sie den Fluss f entlang des Wegs p um $\gamma := \min_{e \in p} u^f(e)$.

Schritt 5: Gehen Sie zu Schritt 2.

Welche Auswirkung hat das Vorgehen von Edmonds-Karp auf die Maximale-Fluss-Berechnung im Diamantgraphen aus Abbildung 10.15? Bei dieser Wahl des augmentierenden Wegs reichen $0,5 \cdot mn$ Iterationen zur Berechnung des maximalen Flusses aus.

Aus dem Algorithmus lässt sich leicht folgern, dass bei ganzzahligen oberen Kapazitäten immer ein maximaler Fluss existiert, der nur ganzzahlige Werte auf den Kanten annimmt. Diese Aussage werden wir später für Matchings noch brauchen.

Lemma 10.3. Sei (G, u, s, t) ein Netzwerk mit $u(e) \in \mathbb{N}$ für alle $e \in E$. Dann existiert ein maximaler Fluss f in G mit $f(e) \in \mathbb{N}$ für alle $e \in E$.

Als Nächstes zeigen wir, dass beide Algorithmen einen maximalen Fluss berechnen. Außerdem lernen wir ein Kriterium kennen, mit dem wir den Wert des maximalen Flusses bestimmen können, ohne den Fluss zu berechnen.



10.3 Der Max-Fluss-Min-Schnitt-Satz

Kurzer Szenenwechsel: Die junge Aktivistengruppe Watercuts hat sich zum Ziel gesetzt, auf den drohenden Wassermangel aufmerksam zu machen. Da in dem neuen Einkaufszentrum der Wasserverbrauch besonders hoch ist, wollen sie ein Zeichen setzen und die gesamte Wasserzufuhr dorthin unterbrechen. Der Plan ist nun folgender: Beim nächsten Vollmond soll es zu den Wasserleitungen gehen, um dort so viele Rohre zu zersägen, bis kein Wasser mehr zum neuen Einkaufszentrum gelangen kann. Martin und Mandy unterbreiten der Gruppe drei verschiedene Pläne, die besagen, welche Rohre zerstört werden sollen (Abb. 10.16). Zu welchem Plan würden Sie der Gruppe raten? Dabei können Sie davon ausgehen, dass für das Zersägen eines Rohres der Breite fünf auch fünfmal so viel Arbeit auf unsere Aktivisten zukommt im Vergleich zu einem Rohr der Breite eins.

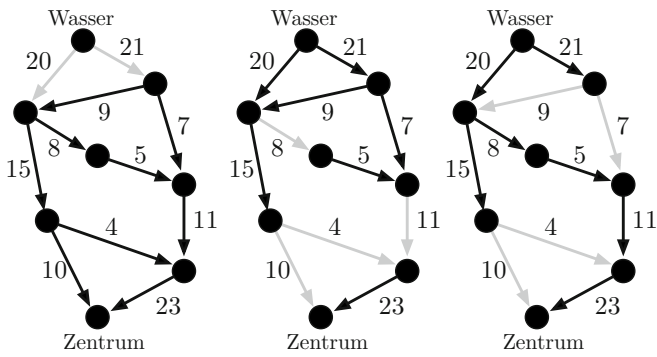


Abb. 10.16: Die grauen Rohre sollen zersägt werden. Welchen Plan würden Sie ausführen lassen? Oder haben Sie einen besseren Vorschlag?

In die Graphentheorie übersetzt sucht die Aktivistengruppe einen (s, t) -Schnitt mit geringer Kapazität:

Definition. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$ und $\emptyset \neq X \subsetneq V$ mit $s \in X$ und $t \notin X$ eine Knotenmenge. Dann ist die Kantenmenge

$$\delta(X) := \{(u, v) \in E \mid u \in X \text{ und } v \in V \setminus X \text{ bzw. } v \in X \text{ und } u \in V \setminus X\}$$

ein (s, t) -Schnitt. Die **Vorwärtskanten** des Schnittes sind die Kanten

$$\delta^+(X) := \{(u, v) \in E \mid u \in X \text{ und } v \in V \setminus X\}$$

und die **Rückwärtskanten**

$$\delta^-(X) := \{(u, v) \in E \mid v \in X \text{ und } u \in V \setminus X\}.$$

Ein (s, t) -Schnitt enthält also eine Menge von Kanten, die „durchtrennt“ werden müssen, damit s und t nicht mehr miteinander verbunden sind. In jedem Schnitt gibt es dann zwei Arten von Kanten: diejenigen, die aus der Knotenmenge X herausgehen (Bezeichnung $\delta^+(X)$), und diejenigen, die in die Knotenmenge hineingehen (Bezeichnung $\delta^-(X)$) (Abb. 10.17). Die Kapazität eines (s, t) -Schnittes wird nur über seine Vorwärtskanten definiert.

Definition. Die **Kapazität** des (s, t) -Schnittes $\delta(X)$ im Netzwerk (G, u, s, t) beträgt

$$\text{cap}(X) := \sum_{e \in \delta^+(X)} u(e).$$

Welchen Wert hat der Schnitt aus der Abbildung 10.17, wenn alle Kanten die Kapazitäten eins haben?

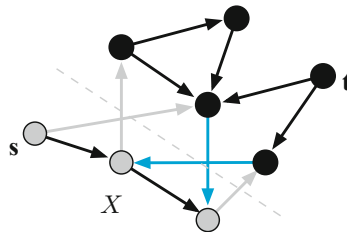


Abb. 10.17: Die gefärbten Kanten bilden den Schnitt $\delta(X)$, wobei X die grauen Knoten enthält. Die blauen Kanten werden auch mit $\delta^-(X)$ und die grauen mit $\delta^+(X)$ bezeichnet.

Zurück zu dem Problem der Watercuts: Welcher von Martins und Mandys Plänen stellt einen (s, t) -Schnitt dar und welche Kapazitäten haben diese dann? Welchen Plan bevorzugen Sie jetzt oder kann man das Ziel mit einem noch geringeren Aufwand erreichen?



Vielleicht drängt sich Ihnen mittlerweile die Frage auf, was das alles mit maximalen Flüssen zu tun hat. Einen ersten Zusammenhang zwischen den beiden Konzepten zeigt das folgende Lemma.

Lemma 10.4. Sei (G, u, s, t) ein Netzwerk. Für jeden (s, t) -Schnitt $\delta(X)$ und jeden (s, t) -Fluss f gilt:

1. Der Flusswert ergibt sich aus der Belastung der Vorwärtskanten abzüglich der Belastung der Rückwärtskanten des Schnittes, d.h.

$$\text{value}(f) = \sum_{e \in \delta^+(X)} f(e) - \sum_{e \in \delta^-(X)} f(e)$$

2. Die Kapazität des Schnittes beschränkt den Flusswert, d.h.

$$\text{value}(f) \leq \text{cap}(X).$$

Beweis: Für den Wert des Flusses f gilt

$$\text{value}(f) = \sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e).$$

An jedem Knoten $v \in V \setminus \{s, t\}$ gilt die Flusserhaltung

$$\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = 0.$$

Also verändert die Addition dieses Wertes für jeden Knoten $v \in X \setminus \{s\}$ den Flusswert nicht:

$$\begin{aligned} \text{value}(f) &= \sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e) \\ &= \sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e) + \underbrace{\sum_{v \in X \setminus \{s\}} \left(\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) \right)}_{=0} \\ &= \sum_{v \in X} \left(\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) \right). \end{aligned}$$

Jetzt wird jedoch der Flusswert auf jeder Kante (x, y) , die in X liegt, genau zweimal gezählt: einmal positiv in der Menge $\delta^+(x)$ und einmal negativ in der Menge $\delta^-(y)$ (Abb. 10.18). Damit tragen zu dem Flusswert nur noch die Kanten bei, die entweder aus der Menge X herausgehen ($e \in \delta^+(X)$) oder die in die Menge X hineingehen ($e \in \delta^-(X)$). Daraus folgt

$$\text{value}(f) = \sum_{e \in \delta^+(X)} f(e) - \sum_{e \in \delta^-(X)} f(e). \quad (10.1)$$

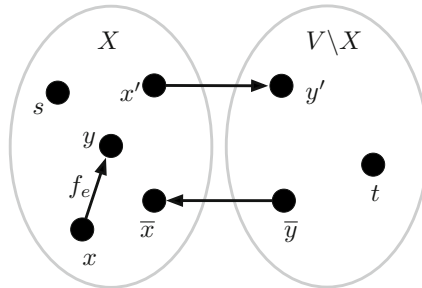


Abb. 10.18: Die Kante $e = (x, y)$ liegt in der Menge X . Damit wird der Flusswert $f(e)$ einmal positiv für den Knoten x und einmal negativ für den Knoten y gezählt.

Die zweite Aussage folgt aus Gleichung (10.1), da der Fluss die oberen Kapazitäten einhält:

$$\begin{aligned} \text{value}(f) &= \sum_{e \in \delta^+(X)} f(e) - \sum_{e \in \delta^-(X)} f(e) \\ &\leq \sum_{e \in \delta^+(X)} f(e) \\ &\leq \sum_{e \in \delta^+(X)} u(e) = \text{cap}(X). \end{aligned}$$

□

Mit diesen beiden Aussagen lässt sich nun die Optimalität des Algorithmus von Ford-Fulkerson beweisen.

Satz 10.5 (Optimalitätskriterium für maximale Flüsse). *Sei (G, u, s, t) ein Netzwerk. Ein (s, t) -Fluss f ist genau dann maximal, wenn es keinen f -augmentierenden (s, t) -Weg in G^f gibt.*

Beweis: Die Hin-Richtung ist klar. Überlegen Sie sich dazu, wieso es für einen maximalen (s, t) -Fluss keinen f -augmentierenden (s, t) -Weg mehr geben kann?⁴⁵ Die Rück-Richtung ist etwas komplizierter: Sei R die Menge aller Knoten, die von s aus in G^f erreichbar sind, d.h.



$$R := \{v \in V \mid \text{es gibt einen gerichteten } (s, v)\text{-Weg in } G^f\}.$$

Insbesondere gilt dann $s \in R$ und $t \notin R$ (ansonsten würde ja ein augmentierender (s, t) -Weg in G^f existieren).

Die Menge R bestimmt einen (s, t) -Schnitt in G . Betrachten wir eine Vorwärtskante (x, y) aus $\delta^+(R)$ (Abb. 10.19). Da der Knoten y nicht in R liegt, darf die Kante (x, y) nicht im Residualgraphen G^f enthalten sein. Dann muss aber $f(e) = u(e)$ gelten. Sei nun die Kante $(u, v) \in \delta^-(R)$, d.h. wir erreichen den Knoten v von s aus im Residualgraphen, den Knoten u jedoch nicht. Dann darf über diese Kante (u, v) kein Fluss fließen, da sonst die Kante (v, u) in G^f enthalten wäre.

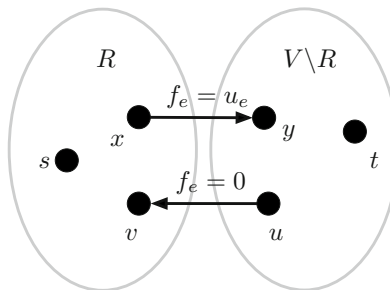


Abb. 10.19: Für alle Vorwärtskanten des Schnittes $\delta(R)$ gilt $f(e) = u(e)$ und für alle Rückwärtskanten $f(e) = 0$.

Insgesamt ergibt sich mit Lemma 10.4:

$$\begin{aligned}
 \text{value}(f) &= \sum_{e \in \delta^+(R)} f(e) - \sum_{e \in \delta^-(R)} f(e) \\
 &= \sum_{e \in \delta^+(R)} u(e) - \sum_{e \in \delta^-(R)} 0 \\
 &= \text{cap}(X).
 \end{aligned}$$

Da allgemein $\text{value}(f) \leq \text{cap}(X)$ gilt, ist f maximal. \square

Mit dem Algorithmus von Ford-Fulkerson steht der Berechnung eines maximalen Flusses nichts mehr im Wege.

Satz 10.6. *Der Algorithmus von Ford-Fulkerson berechnet in jedem Netzwerk (G, u, s, t) einen maximalen (s, t) -Fluss.*

Wir wollen das Thema der maximalen Flüsse noch nicht ganz beenden: Müssen wir denn immer erst einen maximalen Fluss berechnen, um zu wissen, wie viel durch ein Netzwerk passt oder kann man ein davon unabhängiges Kriterium finden? Könnten die (s, t) -Schnitte nicht genau ein solches darstellen?

Satz 10.7 (Max-Fluss-Min-Schnitt-Satz, Ford-Fulkerson 1956, Elias, Feinstein, Shannon 1956). *In einem Flussnetzwerk ist der maximale Flusswert eines (s, t) -Flusses gleich der minimalen Kapazität eines (s, t) -Schnittes.*

Wie viel Wasser kann also durch das folgende Netzwerk gepumpt werden (Abb. 10.20)? Wie sieht darin ein maximaler Fluss und wie ein minimaler Schnitt aus?



Vielleicht haben Sie den Beweis zu dem Max-Fluss-Min-Schnitt-Satz 10.7 vermisst. Aber wir haben ihn quasi schon geführt. Schauen Sie sich dazu noch mal den Beweis zu Satz 10.5 an.

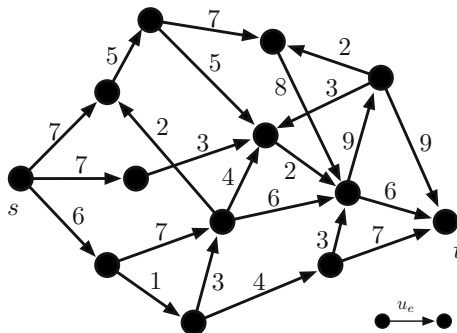


Abb. 10.20: Berechnen Sie einen maximalen Fluss und einen minimalen Schnitt in diesem Graphen.

10.4 Aufgaben

Aufgabe 10.1. In dem Graphen aus der Abb. 10.21 sind die Kantenkapazitäten in schwarz und ein (s, t) -Fluss in blau gegeben. Ist der Fluss maximal? Wenn nicht, wenden Sie den Algorithmus von Ford-Fulkerson auf den Graphen und den gegebenen Fluss an.

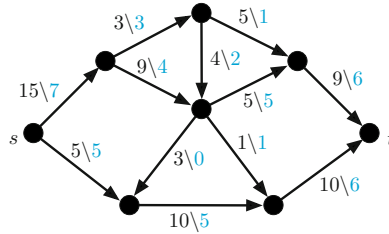


Abb. 10.21: Die schwarzen Kantenbewertungen geben die oberen Kapazitäten an, die blauen den Fluss.

Aufgabe 10.2. Teil 1: Das Fuhrunternehmen Container & Co hat sich auf den Transport von Containern spezialisiert. Das Konzept ist dabei das Folgende: An vielen wichtigen Orten wurden Sammelzentren installiert, zwischen denen Lastwagen unterwegs sind, um die Container zu transportieren (Abb. 10.22). Zwischen dem Sammelzentrum V und dem Sammelzentrum E fährt z.B. ein Lastwagen, der 4 Container pro Fahrt transportieren kann.

Wie viele Container können maximal von Sammelzentrum HC zum Sammelzentrum HT transportiert werden?

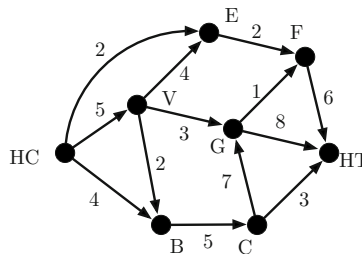


Abb. 10.22: Zwischen je zwei Sammelzentren steht nur eine begrenzte Anzahl an Transportmöglichkeiten zur Verfügung.

Teil 2: Bei der Berechnung der maximalen Containerzahl haben die Planer bisher außer Acht gelassen, dass in einem Sammelzentrum nicht beliebig viele Container von einem Lastwagen in den anderen umgeladen werden können. In der Sammelstelle V ist nur das Umladen von 4 Containern möglich (Abb. 10.23).

Wie viele Container können nun maximal zwischen HC und HT an einem Tag transportiert werden?

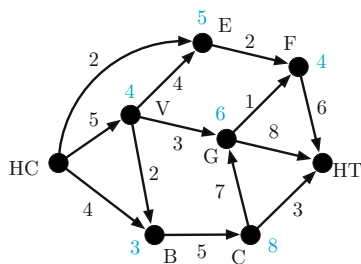


Abb. 10.23: Zwischen je zwei Sammelzentren steht nur eine begrenzte Anzahl an Transportmöglichkeiten zur Verfügung.

Teil 3: Verallgemeinert erhält man das folgende Problem: Zusätzlich zu den Kapazitäten der Kanten haben auch die Knoten eine bestimmte Kapazität, die durch den Zu- bzw. Abfluss nicht überschritten werden darf. (D.h. für das Flussnetzwerk $N = (G, u, s, t)$ mit $G = (V, E)$ existiert eine zusätzliche Kapazitätsfunktion $d : V \rightarrow \mathbb{R}_{\geq 0}$ mit

$$\sum_{\delta^+(v)} f(e) \leq d(v) \quad \text{für alle } v \in V \setminus \{s, t\}$$

als zusätzliche Beschränkung für einen Fluss f .) Gesucht ist wiederum ein maximaler Fluss, der neben den Kantenkapazitäten auch die Knotenkapazitäten einhält. Wie kann man das Problem als maximales Flussproblem mit oberen Kantenkapazitäten aber ohne Knotenkapazitäten modellieren?

Aufgabe 10.3. Gegeben seien zwei Gitter G_1 und G_2 mit jeweils acht blau eingekreisten Startknoten (Abb. 10.24). Gibt es acht Wege, die jeweils von einem blauen markierten Knoten zu einem Randknoten führen, sodass diese Wege keine gemeinsamen Kanten nutzen?

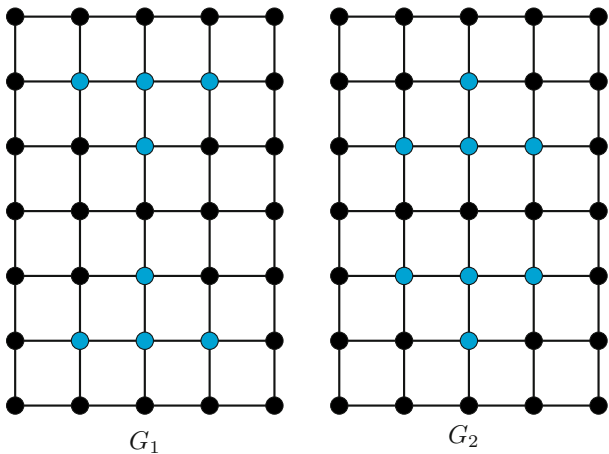


Abb. 10.24: Interpretiert man die dicken Punkte als Klassenzimmer, so entspricht diese Aufgabe der Notwendigkeit unabhängige Fluchtwege für jede Klasse festzulegen.

Aufgabe 10.4. Teil 1: Die Firma Datensecure betreibt ein Telekommunikationsnetzwerk in Deutschland (Abb. 10.25). Um die Sicherheit zu erhöhen, hat sich die Firma überlegt, Datensätze in kleinere Pakete zu unterteilen. Jedes Teilpaket eines Dokumentes soll nun entlang eines eigenen Wegs durch das Netzwerk geroutet werden. Dabei sollen zwei Teilpakete nicht über eine gemeinsame Leitung verlaufen, da ansonsten der Sicherheitsgewinn durch das Teilen wieder aufgehoben wäre (die Firma geht davon aus, dass die Leitungen und nicht die Server ein Sicherheitsrisiko darstellen).

1. In wie viele Pakete kann ein Dokument, das von Berlin nach Hamburg geschickt werden soll, sinnvoll unterteilt werden?
2. Entlang welcher Wege sollten die Pakete geschickt werden?

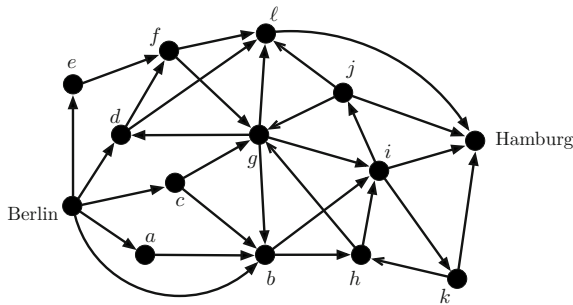


Abb. 10.25: Das Telekommunikationsnetzwerk der Firma Datensecure besteht aus Leitungen, die zwischen den unterschiedlichen Servern verlaufen.

Teil 2: Sei $G = (V, A)$ ein Graph, $s, t \in V$. Wie kann die maximale Anzahl kantendisjunkter (s, t) -Wege in G ermittelt werden?

Aufgabe 10.5. Konstruieren Sie aus dem folgenden (s, t) -Fluss einen Wegfluss (Abb. 10.26).

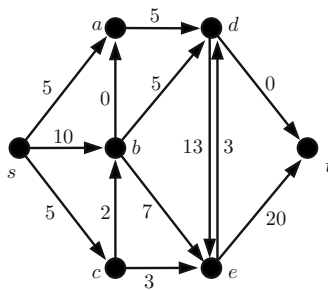


Abb. 10.26: Der (s, t) -Fluss ist durch den ersten Wert an der Kante gegeben.

Aufgabe 10.6. Teil 1: Die Firma KLIMASAVE hat in Deutschland einen großen Absatzmarkt für Klimaanlage entdeckt. Deswegen hat sie ganz Deutschland mit

einem Netz von Produktionsstätten wie auch von Verkaufsstätten überzogen. Verbunden sind die einzelnen Komplexe durch ein Transportsystem (Abb. 10.27). Für die einzelnen Verkaufsstätten ist KLIMASAVE bekannt, wie viele Klimaanlage dort vertrieben werden können. Genauso ist ihr die maximale Produktionsleistung der einzelnen Produktionsstätten bekannt.

Wie viele Klimaanlage kann die Firma unter Berücksichtigung der Produktions- und Verkaufseinschränkungen und der Einschränkungen durch das Transportnetz maximal verkaufen?

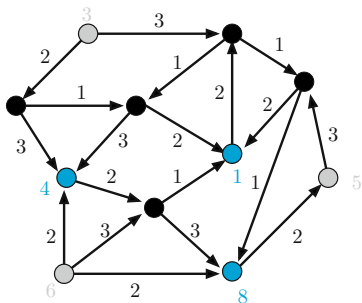


Abb. 10.27: Das Produktions- und Verkaufszentrum der Firma KLIMASAVE. Die grauen Knoten repräsentieren die Produktionszentren und die blauen die Verkaufszentren.

Teil 2: Wie kann bei mehreren Quellen und Senken mit begrenzten Sende- bzw. Empfangskapazitäten ein maximales Transportvolumen ermittelt werden?

Aufgabe 10.7. Teil 1: Nach einem großen Fussballspiel wollen alle Fans wieder zurück nach Hause und müssen dazu zur U-Bahn-Haltestelle. Zur Haltestelle gibt es mehrere Wege, die teils über Rolltreppen und teils einfach entlang enger Gänge gehen (Abb. 10.28). Der U-Bahn-Betreiber fragt sich nun: Wie viele Menschen kommen maximal pro Minute unten auf den Gleisen an? Von dieser Information will er abhängig machen, wie häufig die Bahnen fahren sollen.

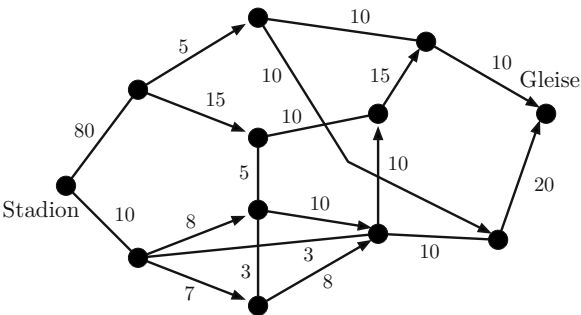


Abb. 10.28: Das Gangsystem vom Stadion zur Haltestelle ist nach den letzten Umbauarbeiten ziemlich verworren geworden.

Teil 2: Wie kann ein maximaler Fluss in einem Graphen, der auch ungerichtete Kanten enthält, berechnet werden?

Aufgabe 10.8. In einem Netzwerk mit ganzzahligen oberen Kapazitäten existiert immer ein ganzzahliger maximaler Fluss. Muss andersherum auch jeder maximale Fluss ganzzahlig sein?

Aufgabe 10.9. Seien S und T zwei minimale (s, t) -Schnitte in einem Netzwerk (G, u, s, t) . Zeigen Sie, dass $S \cup T$ und $S \cap T$ ebenfalls zwei minimale (s, t) -Schnitte in G bilden.

Lösungen zu den Fragen im Text

- 43 Der maximale Flusswert in diesem Netzwerk beträgt 6.
- 44 Es können 20 Iterationen von Nöten sein.
- 45 Da wir ansonsten entlang dieses Wegs den (s, t) -Fluss verändern können und einen neuen Fluss mit einem höheren Wert erhalten. Dann kann f aber nicht maximal gewesen sein.

11 Kostenminimale Flüsse

Übersicht

| | | |
|------|--------------------------------|-----|
| 11.1 | Problemstellung | 187 |
| 11.2 | Ein Optimalitätskriterium..... | 190 |
| 11.3 | Zwei Algorithmen | 193 |
| 11.4 | Aufgaben | 199 |

11.1 Problemstellung

Die Firma Back & Co hat ganz Deutschland mit ihren Backfilialen überzogen. In den einzelnen Filialen werden die Brote zwar noch gebacken, die Herstellung der rohen Teigwaren erfolgt jedoch in Backzentren. Jeden Mittag melden die einzelnen Filialen ihren Brotbedarf für den kommenden Tag an. Die gängigen Mengen sind dabei ein oder zwei Transporterlieferungen pro Filiale. In den Backzentren werden dann die benötigten Backwaren hergestellt. Natürlich haben auch die Backzentren eine begrenzte Produktionskapazität. Sind alle Backwaren fertig, so werden sie auf die einzelnen Transporter verteilt und zu den Filialen gefahren. Die Fahrtkosten eines Transporters sind abhängig von der Entfernung, die zwischen dem Backzentrum und der Filiale liegt. Damit die Kosten des Unternehmens gesenkt werden, sollen nun möglichst kostengünstige Routen berechnet werden, die den Brotbedarf der einzelnen Filialen decken und gleichzeitig die Produktionskapazitäten der Backzentren nicht überschreiten.

In Abbildung 11.1 ist das Transportnetz der Firma Back & Co dargestellt. Pro Verkäuferin erwartet die Filiale eine Transporterlieferung. Die Produktionskapazität der Backzentren sind durch ihre Anzahl an vorhandenen Transportern markiert. Für jeden Straßenzug sind nur seine maximale Belastung an Verkehr und die Fahrtkosten angegeben. Welche Routen sollten die Transporter wählen, sodass dem Unternehmen möglichst wenige Transportkosten entstehen?

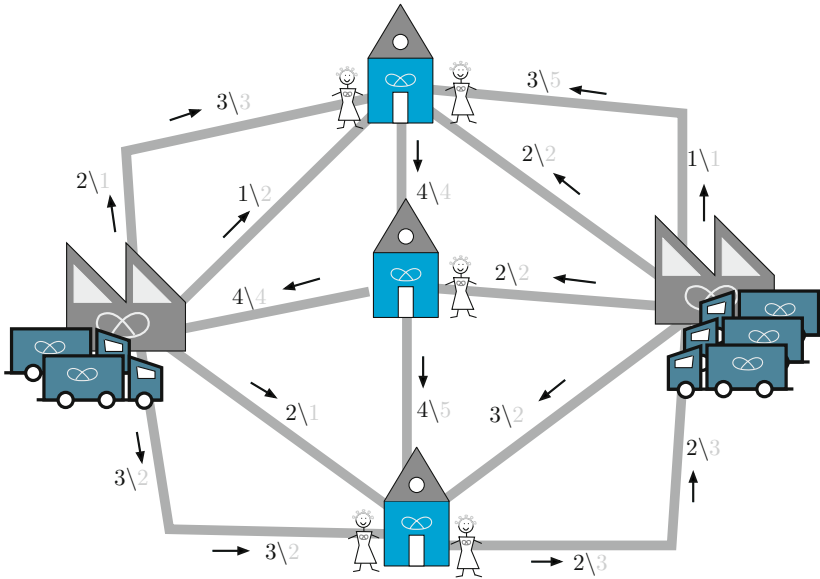


Abb. 11.1: An den Straßenzügen ist die maximale Belastung in schwarz und die Fahrzeit in grau angegeben.

Dieses Transportproblem kombiniert einige Probleme, die wir bereits kennen gelernt haben. Zum einen wollen wir so kostengünstig oder so schnell wie möglich von einem Ort zum anderen kommen. Dies entspricht einem Kürzesten-Wege-Problem. Zum anderen sollen Einheiten in einem Netzwerk verschickt werden, ein typisches Fluss-Problem. Dementsprechend ist die mathematische Formulierung des Problems auch eine Mischung aus beiden Problemstellungen.

Das zugrunde liegende Transportnetzwerk modellieren wir als Graphen $G = (V, E)$. Da die Belastung durch den Verkehr auf einigen Straßen reguliert ist, weisen wir den Kanten wie beim Maximalen-Fluss-Problem obere Kapazitäten u zu. Zusätzlich zu den oberen Kapazitäten haben wir noch eine Kostenfunktion c , die jeder Kante Transportkosten zuweist. Das Angebot und die Nachfrage modellieren wir durch eine Knotenbewertung $b(v) \in \mathbb{Z}$, $v \in V$. Gilt $b(v) > 0$, so stellt dieser Knoten das Angebot (z.B. die Backwaren) $b(v)$ zur Verfügung. Ein solcher Knoten heißt auch **Quelle**. Gilt hingegen $b(v) < 0$, so existiert an dem Knoten v eine Nachfrage von $b(v)$ Einheiten. Diese Knoten werden auch als **Senken** bezeichnet. Ein Knoten v mit $b(v) = 0$ hat weder ein Angebot noch eine Nachfrage und entspricht in unserem Beispiel einer Straßenkreuzung, bei der nichts eingeladen oder abgeliefert werden soll. Im Folgenden gehen wir davon aus, dass $\sum_{v \in V} b(v) = 0$ gilt. Eine solche Knotenbewertung bezeichnen wir auch als **Balance**. Die graphentheoretische Modellierung des Netzwerks in der Abbildung 11.1 entspricht dann dem Graphen in der Abbildung 11.2 links.

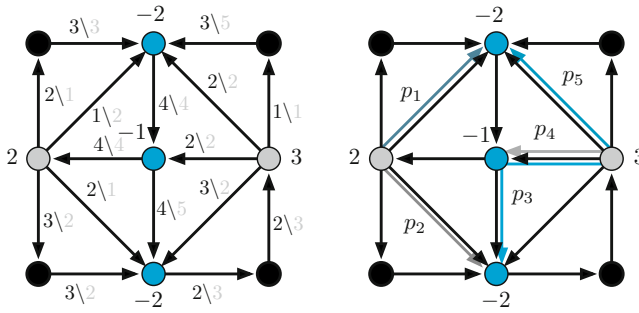


Abb. 11.2: Die grauen Knoten sind die Quellen und die blauen die Senken. Schickt man jeweils einen Lastwagen entlang der Wege p_1, \dots, p_5 , so werden alle Backfilialen versorgt.

Jetzt fehlt nur noch die Darstellung einer Lösung. Rechts in der Abbildung 11.2 ist schon eine mit Hilfe von Wegen eingezeichnet. Aber wir hatten schon bei den maximalen Flüssen festgestellt, dass ein solcher Ansatz über Wege nicht sinnvoll ist. Deswegen definieren wir genauso wie bei den (s, t) -Flüssen nun b -Flüsse über die Belastungen auf den einzelnen Kanten.

Definition. Sei $G = (V, E)$ ein gerichteter Graph mit oberen Kapazitäten $u(e)$ für jede Kante $e \in E$ und b eine Balance. Ein **b -Fluss** ist eine Kantenbewertung $f(e)$, $e \in E$, die die oberen Kapazitäten einhält und für die der Einfluss in einen Knoten v abzüglich des Ausflusses in dem Knoten v dem Wert $b(v)$ entspricht, d.h.

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

Zeichnen Sie einen b -Fluss in die Abbildung 11.2. Gibt es unterschiedliche b -Flüsse in dem gegebenen Graphen und den Balancen? Ist ein (s, t) -Fluss auch ein b -Fluss und wie sehen dann die Balancen aus?

Ein b -Fluss muss nicht immer existieren. Wie sieht ein Graph mit oberen Kapazitäten und Balancen aus, für die es keinen b -Fluss gibt? Wie kann eine solche Überprüfung auf ein maximales Fluss Problem übertragen werden?⁴⁶

Ist die Existenz eines solchen Flusses geklärt, so wollen wir natürlich einen mit den geringsten Kosten auswählen. Wir gehen hier davon aus, dass auch negative Kosten (d.h. Gewinne) in der Kostenfunktion auftreten können.

Definition. Sei $G = (V, E)$ ein gerichteter Graph mit oberen Kapazitäten $u(e) \in \mathbb{N}$ und Kosten $c(e) \in \mathbb{Z}$ für jede Kante $e \in E$. Außerdem sei b eine Balance. Ein **kostenminimaler Fluss** f ist ein b -Fluss mit minimalen Kosten. Die **Kosten** eines b -Flusses sind gegeben durch

$$c(f) = \sum_{e \in E} c(e) \cdot f(e).$$



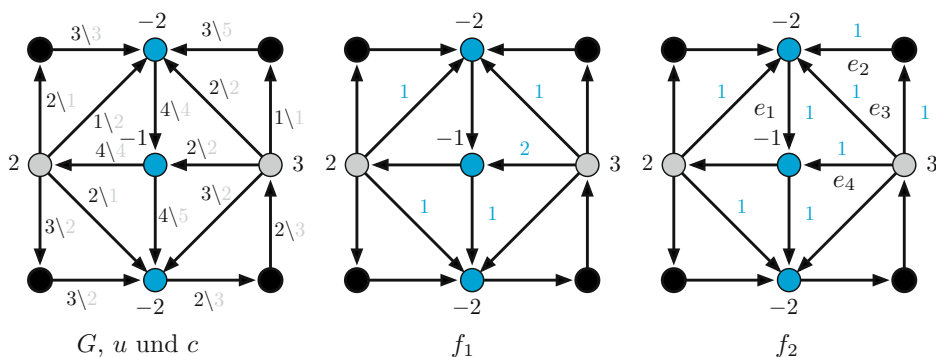


Abb. 11.3: Im linken Graphen sind die oberen Kapazitäten u und die Kantenkosten c angegeben. Die beiden Kopien von G enthalten die Flüsse f_1 und f_2 .



In der Abbildung 11.3 sind zwei unterschiedliche b -Flüsse f_1 und f_2 (Mitte und rechts) eingezeichnet. Welche Kosten ergeben sich für f_1 , welche für f_2 ? Wie man leicht sieht, ist der b -Fluss f_2 um einiges teurer als der linke. Um von dem b -Fluss f_2 zu dem b -Fluss f_1 zu kommen, muss man allerdings nur den Fluss entlang der Kanten e_1 , e_2 und e_3 zurückschicken und statt dessen entlang der Kante e_4 verschicken. Das Zurückschicken eines Flusses hatten wir in Kapitel 10 über den Residualgraphen modelliert (Definition auf Seite 171). Betrachtet man diese Kanten genauer, so ergeben sie einen Zykel im Residualgraphen von f_2 . Mit dieser Feststellung sind wir schon auf halbem Weg zu einem Optimalitätskriterium für kostenminimale b -Flüsse.

11.2 Ein Optimalitätskriterium

Für maximale Flüsse hatten wir das folgende Kriterium kennen gelernt (Satz 10.5): Ein (s, t) -Fluss f ist genau dann maximal, wenn es keinen f -augmentierenden Weg in G^f gibt. Mit G^f hatten wir den Residualgraphen bzgl. f bezeichnet (Definition auf Seite 171).

Ein ähnliches Kriterium erhalten wir für kostenminimale Flüsse. Statt eines f -augmentierenden Wegs definieren wir nun einen f -augmentierenden Zykel Z . Ein **f -augmentierender Zykel** ist ein gerichteter Kreis in dem Residualgraphen G^f . Verändert man einen gegebenen b -Fluss f entlang eines solchen Zyklus um den Wert $\gamma \leq \min_{e \in Z} u^f(e)$, so erhält man wieder einen zulässigen b -Fluss f' . Dabei definiert man

$$f'(e) = \begin{cases} f(e) & \text{falls } e \notin Z \\ f(e) + \gamma & \text{falls } e \in Z \\ f(e) - \gamma & \text{falls } \overleftarrow{e} \in Z. \end{cases}$$

Natürlich kann eine solche Veränderung sofort wieder rückgängig gemacht werden.

Der Begriff augmentierend ist in diesem Kontext etwas verwirrend, da der Fluss nur verändert wird und sich nichts erhöht. Allerdings wird er so in der Literatur verwendet. Betrachten wir noch schnell ein Beispiel, wie das Augmentieren entlang von Zykeln funktioniert, bevor wir zu einem Optimalitätskriterium kommen.

Beispiel. In der Abbildung 11.4 oben links ist ein Graph G mit oberen Kapazitäten (schwarz) und einem b -Fluss f (blau) gegeben. Der dazugehörige Residualgraph G^f mit einem f -augmentierenden Zykel Z_1 ist rechts davon eingezeichnet. Unten links ist nun die Kantenbewertung eingetragen, die durch die oben gegebene Formel entsteht, wenn man f entlang des Zyklus Z_1 um den Wert 3 verändert.

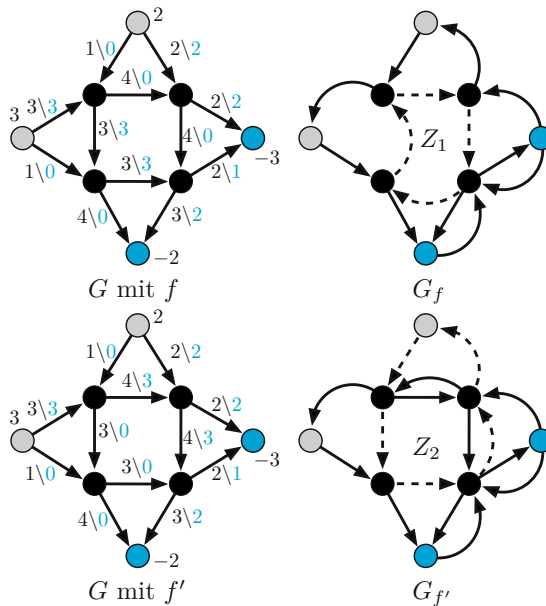


Abb. 11.4: Die grauen Knoten bilden die Quellen in diesem Graphen und die blauen die Senken. Ihre Balancewerte stehen neben den Knoten. Die schwarzen Zahlen an den Kanten geben die oberen Kapazitäten an, die blauen die b -Flüsse.

Wie sieht nun ein b -Fluss f'' aus, der aus dem Fluss f' durch das Augmentieren entlang des Zyklus Z_2 um den Wert 1 entsteht? ■



Damit haben wir gesehen, dass ein b -Fluss in einen anderen b -Fluss durch die Veränderung entlang eines f -augmentierenden Zyklus Z überführt werden kann. Allerdings ist damit noch nichts über die Kosten der beiden b -Flüsse gesagt. Wann lohnt es sich nun, entlang eines f -augmentierenden Zyklus den gegebenen b -Fluss zu verändern? Dazu erweitern wir die Definition des Residualgraphen und der Residualkapazitäten u^f noch um die **Residualkosten** c^f . Die Vorwärtskanten er-

halten dabei die ursprünglichen Kosten und die Rückwärtskanten den negativen Wert ihrer Vorwärtskanten, d.h.

$$\begin{aligned} c^f(e) &= c(e) & \forall e \in E \\ c^f(\overleftarrow{e}) &= -c(e) & \forall e \in E. \end{aligned}$$

Die Residualkosten sind im Gegensatz zu den oberen Kapazitäten unabhängig von dem gegebenen Fluss. Die Kosten für einen f -augmentierenden Zykel Z ergeben sich nun wieder über die Summe der Kosten der in ihm enthaltenen Kanten, d.h.

$$c(Z) = \sum_{e \in Z} c^f(e).$$

Mit diesen Kosten können wir nun das folgende Kriterium formulieren.

Satz 11.1 (Optimalitätskriterium für MCFP, Klein 1967). *Sei $G = (V, E)$ ein gerichteter Graph, u die oberen Kantenkapazitäten, c eine Kostenfunktion auf den Kanten und b eine Balance. Dann ist f genau dann ein kostenminimaler Fluss, wenn kein f -augmentierender Zykel Z mit negativen Kosten in G^f existiert.*



Ist der b -Fluss f in der Abbildung 11.5 nach diesem Kriterium minimal? Wenn nicht, welcher Zykel in dem Residualgraphen hat negative Kosten?

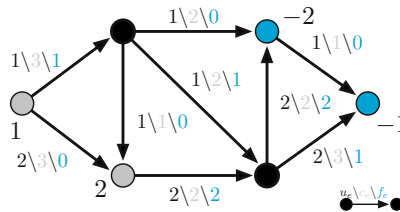


Abb. 11.5: Die schwarzen Werte geben die oberen Kapazitäten, die grauen die Kosten und die blauen einen b -Fluss an, die schwarzen Werte obere Kapazitäten und die grauen die Kosten.

Ob ein f -augmentierender Zykel auch negative Kosten hat, hängt stark von der Kostenfunktion ab. Wie sieht eine Kostenfunktion für das Beispiel auf Seite 191 aus, sodass Z_1 ein f -augmentierender Zykel mit negativen Kosten ist? Bei welcher Kostenfunktion wäre Z_2 ein Zykel mit negativen Kosten?



Ähnlich wie beim Algorithmus von Ford-Fulkerson zur Berechnung eines maximalen Flusses lässt sich auch aus diesem Optimalitätskriterium ein Algorithmus zur Berechnung eines kostenminimalen Flusses gewinnen. Wie genau, sehen wir im nächsten Abschnitt.

11.3 Zwei Algorithmen

Für das Maximale-Fluss-Problem hatten wir den Algorithmus von Ford-Fulkerson kennen gelernt. In jeder Iteration wird dabei ein f -augmentierender Weg berechnet und entlang dieses Wegs der bisherige Fluss f verändert. Startet man anstatt mit einem zulässigen (s, t) -Fluss mit einem b -Fluss und berechnet man in jeder Iteration einen f -augmentierenden Zykel mit negativen Kosten, so erhält man einen Algorithmus, der einen kostenminimalen Fluss berechnet. Wie kann ein solcher Zykel mit negativen Kosten berechnet werden?⁴⁷ Der sogenannte *Cycle-Canceling*-Algorithmus wurde 1967 von dem Mathematiker Klein vorgestellt.



Algorithmus 11.1 Cycle-Canceling-Algorithmus

Input: Gerichteter Graph $G = (V, E)$, obere Kapazitäten u , Kantenkosten c , Balance b .

Output: Kostenminimaler Fluss f .

Schritt 1: Berechnen Sie einen b -Fluss.

Schritt 2: Bestimmen Sie G^f , u^f und c^f .

Schritt 3: Konstruieren Sie einen f -augmentierenden Zykel Z in G^f mit negativen Kosten. Falls keiner existiert: STOPP.

Schritt 4: Verändern Sie den b -Fluss f entlang des Zyklus Z um $\gamma := \min_{e \in Z} u^f(e)$.

Schritt 5: Gehen Sie zu Schritt 2.

Leider ergibt sich bei dem *Cycle-Canceling*-Algorithmus 11.1 ein ähnliches Problem wie bei dem Algorithmus von Ford-Fulkerson: In jeder Iteration verbessern sich die Kosten des minimalen Flusses evtl. nur um eine Einheit, wie das folgende Beispiel zeigt.

Beispiel. Wir betrachten wieder den Diamantgraphen und erweitern ihn um eine Kante von s nach t (Abb. 11.6). Die Kosten auf dieser Kante werden auf 1 gesetzt, wohingegen sie auf den anderen Kanten 0 betragen. Die oberen Kapazitäten haben alle einen Wert von N bis auf diejenige Kante, die zwischen x und y verläuft und der (s, t) -Kante. Hier beträgt die obere Kapazität 1 bzw. $2N$. Gesucht ist ein kostenminimaler Fluss für die Balance $b(s) = 2N$, $b(t) = -2N$ und $b(v) = 0$ für die anderen beiden Knoten x und y . Wie sieht ein kostenminimaler Fluss in

diesem Beispiel aus? Angenommen, der Algorithmus wählt im ersten Schritt den zulässigen b -Fluss f mit

$$f(e) = \begin{cases} 2N & \text{für } e = (s, t) \\ 0 & \text{sonst.} \end{cases}$$



Welche Kreise könnte der *Cycle-Canceling*-Algorithmus wählen, sodass er tatsächlich $2N$ -mal entlang eines negativen Zyklus iteriert? ■

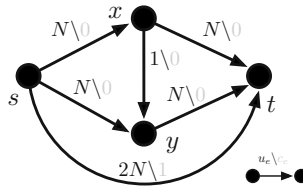


Abb. 11.6: Die oberen Kapazitäten sind wieder in schwarz und die Kosten in grau angegeben.

Auch hier kann man die Idee des *Cycle-Canceling*-Algorithmus verbessern, indem man nicht irgendeinen Zykel mit negativen Kosten wählt, sondern denjenigen mit den geringsten durchschnittlichen Kosten pro Kante. Natürlich wäre es noch besser, den Zykel mit den geringsten Kosten zu wählen. Aber dies entspricht dem Problem einen Hamilton-Kreis zu finden. Zykel mit den geringsten durchschnittlichen Kosten pro Kanten können effizient berechnet werden, was wir hier aber nicht machen möchten. Stattdessen wollen wir jetzt noch eine andere Methode betrachten, um das kostenminimale Fluss-Problem zu lösen.



Der neue Ansatz basiert darauf mit einem kostenminimalen Fluss zu starten, der jedoch die geforderten Balancen nicht einhält. Nach und nach wird dieser Fluss so verändert, dass sich am Ende die gewünschten Balancen ergeben. Dabei wird darauf geachtet, dass nie ein negativer Zykel in dem Residualgraphen zu den unterschiedlichen Flüssen entsteht.

Beispiel. Angenommen, uns ist ein kostenminimaler b -Fluss f zu den Balancen $b(s_1) = 3$, $b(s_2) = 2$, $b(t_1) = -1$ und $b(t_2) = -4$ gegeben (Abb. 11.7). Jetzt ändern sich die Balancen zu $b'(s_1) = 4$, $b'(s_2) = 2$, $b'(t_1) = -2$ und $b'(t_2) = -4$, d.h. s_1 produziert eine Einheit mehr und t_1 benötigt eine Einheit mehr. Wir müssen also zusätzlich eine Einheit von s_1 zu t_1 transportieren. Schon bei maximalen Flüssen haben wir gesehen, dass ein solcher Transport am besten entlang eines (s_1, t_1) -Wegs im Residualgraphen funktioniert. Jetzt könnte man natürlich einen beliebigen Weg aussuchen und hoffen, dass man anschließend immer noch optimal ist. Wählt man allerdings einen kürzesten Weg, so kann man die Kostenminimalität für die neuen Balancen sogar beweisen. Welchen (s_1, t_1) -Weg würden Sie in diesem Fall wählen und wie sieht der neue b' -Fluss aus? ■



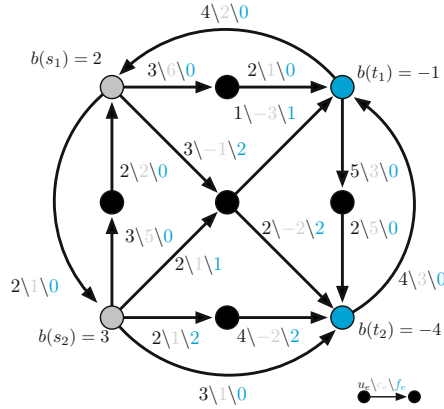


Abb. 11.7: Der erste Wert (schwarz) gibt die obere Kapazität an, der zweite (grau) die Kosten und der dritte (blau) den b -Fluss.

Diese Beobachtung halten wir im folgenden Satz fest.

Satz 11.2 (Jewell 1958, Iri 1969, Busacker & Gowen 1961). *Sei f ein kostenminimaler b -Fluss in dem Digraphen G mit oberen Kapazitäten u , Kantenkosten c und Balancen b . Sei p ein kürzester (s, t) -Weg in G^f bzgl. der Kosten c^f und f' der aus der Veränderung des Flusses f entlang des Wegs p um den Wert $\gamma \leq \min_{e \in p} u^f(e)$ entstandene b' -Fluss. Dann ist f' ein kostenminimaler Fluss zu den Balancen b' mit*

$$b'(v) = \begin{cases} b(v) & v \in V \setminus \{s, t\} \\ b(v) + \gamma & v = s \\ b(v) - \gamma & v = t. \end{cases}$$

Beweis: Nach Konstruktion ist f' ein b' -Fluss. Etwas formaler müsste man wieder die vier Fälle der eingehenden und ausgehenden Kantenarten in jedem Knoten aus p durchgehen.

Nehmen wir nun an, der b' -Fluss f' sei nicht kostenminimal. Dann gibt es einen Zykel Z in $G^{f'}$ mit negativen Kosten. Die restliche Beweisstruktur ist wie folgt: aus dem Zykel Z und p bilden wir einen Hilfsgraphen, der nur aus Kanten von G^f besteht. Außerdem enthält er einen (s, t) -Weg p' , der günstiger als p ist. Dieser ist somit auch in G^f enthalten, ein Widerspruch zur Wahl von p .

Wir konstruieren nun den Hilfsgraphen H aus den Kanten von Z und den Kanten von p abzüglich aller Paare von Vorwärts- und Rückwärtskanten (Abb. 11.8). Der Graph H muss kein einfacher Graph sein, sondern enthält parallele Kanten, wenn diese sowohl in p als auch in Z enthalten sind. Weiter ist jede Kante aus H auch in G^f enthalten: Jede Kante aus p ist auch nach der Wahl von p in G^f enthalten. Falls Z eine Kante e enthält, die nicht in G^f enthalten ist, dann muss diese Kante durch das Augmentieren entlang des Wegs p in den Graphen $G^{f'}$ gekommen sein.

Dann ist aber die umgedrehte Kante von e in p enthalten und wurde somit aus dem Graphen H gelöscht.

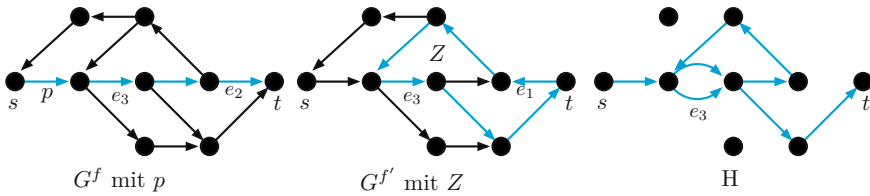


Abb. 11.8: Die Kanten e_1 und e_2 wurden aus der Vereinigung von Z und p gelöscht, um H zu bilden. Die Kante e_3 tritt in H doppelt auf.



Außerdem besteht H aus einem einfachen (s, t) -Weg p' und einem oder mehreren Zykeln. Diese Aussage kann mit vollständiger Induktion über die Anzahl der gelöschten Kantenpaaren von H bewiesen werden. Sowohl p' als auch diese Zyklen sind in G^f enthalten, da H ein Teilgraph von G^f ist. Da f ein kostenminimaler Fluss ist, haben diese Zyklen positive Kosten. Daraus erhalten wir für die Kosten von p' in G^f

$$c^f(p') \leq c^f(E(H)) = \underbrace{c^f(Z) + c^f(p)}_{<0} < c^f(p).$$

Das ist aber ein Widerspruch zur Wahl von p . □

Um aus dieser Beobachtung einen sinnvollen Algorithmus zu konstruieren, müssen wir nur noch in einem gegebenen Graphen für beliebige Balance einen kostenminimalen Fluss bestimmen. Bei positiven Kantenkosten können wir einfach mit dem Null-Fluss, d.h. $f(e) = 0$ für alle Kanten e , starten. Bei negativen Kosten geht dies nicht mehr, wie das nächste Beispiel zeigt.

Beispiel. Betrachten wir das folgende Netzwerk mit den oberen Kapazitäten u und den Kantenkosten c (Abb. 11.9). Der Fluss $f(e) = 0$ ist sicher ein zulässiger b -Fluss für $b(v) = 0$, aber er ist in diesem Fall nicht kostenminimal. Der Zyklus Z_1 hat negative Kosten. Wir könnten jetzt natürlich mit dem *Cycle-Canceling*-Algorithmus einen kostenoptimalen Fluss berechnen, aber dann könnten wir auch direkt das ursprüngliche Problem lösen.

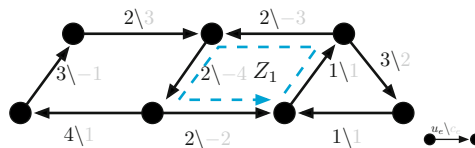


Abb. 11.9: Der Null-Fluss ist in diesem Netzwerk nicht minimal.

Stattdessen erhöhen wir den Flusswert auf jeder Kante mit negativen Kosten um die obere Kapazität (Abb. 11.10). Dabei gehen wir davon aus, dass die oberen

Kapazitäten beschränkt sind. Diese Kantenbewertung ist ein zulässiger b -Fluss für

$$b(v) = \sum_{\substack{e \in \delta^+(v) \\ c(e) < 0}} u(e) - \sum_{\substack{e \in \delta^-(v) \\ c(e) < 0}} u(e).$$

Außerdem ist sie kostenminimal, da in G^f alle Kantenkosten positiv sind und somit kein Zykel mit negativem Gewicht existieren kann. ■

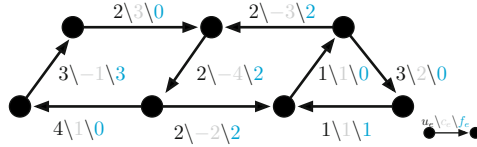


Abb. 11.10: Der Null-Fluss ist in diesem Netzwerk nicht minimal, dafür der hier angegebene b -Fluss. Welche Knoten bilden hier die Quellen und Senken?

Damit können wir den *Successive-Shortest-Path*-Algorithmus formulieren.

Algorithmus 11.2 *Successive-Shortest-Path*-Algorithmus

Input: Gerichteter Graph $G = (V, E)$, obere Kapazitäten u , Kantenkosten c , Balance b .

Output: Kostenminimaler Fluss f , falls existent, oder Aussage, dass keiner existiert.

Schritt 1: Setzen Sie $f(e) = \begin{cases} 0 & \text{falls } c(e) \geq 0 \\ u(e) & \text{falls } c(e) < 0 \end{cases}$ und

$$b(v) = \sum_{\substack{e \in \delta^+(v) \\ c(e) < 0}} u(e) - \sum_{\substack{e \in \delta^-(v) \\ c(e) < 0}} u(e).$$

Schritt 2: Wählen Sie einen Knoten s mit $b(s) - b'(s) > 0$, einen Knoten t mit $b(t) - b'(t) < 0$, der von s aus in G^f erreichbar ist, und gehen Sie zu Schritt 3. Falls kein solches Paar existiert und $b'(v) = b(v)$ für alle $v \in V$ gilt, dann ist f ein kostenminimal. Ansonsten gibt es keinen b -Fluss.

Schritt 3: Berechnen Sie einen kürzesten (s, t) -Weg p bzgl. c^f in G^f .

Schritt 4: Verändern Sie den b -Fluss f entlang des Wegs p um

$$\gamma := \min\{\min_{e \in Z} \{u^f(e)\}, b(s) - b'(s), b'(t) - b(t)\}.$$

Schritt 5: Gehen Sie zu Schritt 2.



Bevor Sie den Algorithmus an dem Graphen mit den oberen Kapazitäten u (schwarz), den Kantenkosten c (grau) und den Balancen b (Abb. 11.11) ausprobieren, betrachten wir noch mal Schritt 3 und 4. In Schritt 3 soll ein kürzester Weg in G^f bzgl. c^f berechnet werden. Dabei können durchaus negative Kantenkosten in G^f auftreten. Wie schon in Kapitel 9 besprochen, ist es im Allgemeinen schwer, einen einfachen kürzesten Weg mit negativen Kantengewichten zu berechnen. Eine Ausnahme bildet der Fall, in dem die Kantengewichte konservativ sind, d.h. keinen negativen Zykel enthalten. Da jeder Fluss, der im Laufe des Algorithmus bestimmt wird, kostenminimal ist, sind die Kantengewichte c^f im Graphen G^f immer konservativ. Ein kürzester Weg kann dann mit dem Algorithmus von Moore-Bellmond-Ford 9.2 berechnet werden. In Schritt 4 geht es dann darum festzulegen, wie viel entlang des Wegs p geschickt werden soll. Dabei gilt es drei Grenzen zu beachten:

$u^f(p)$ die Residualkapazitäten müssen eingehalten werden, damit über keine Kante mehr Fluss fließt, als erlaubt ist bzw. mehr zurückgeschickt wird, als bereits geflossen ist.

$b(s) - b'(s)$ aus der Quelle s darf nicht mehr herausgeschickt werden, als sie am Ende verschicken soll.

$b'(t) - b(t)$ in t sollen nicht mehr als $b(t)$ Einheiten ankommen.



Als Letztes sollten Sie den Algorithmus einmal an dem Graphen aus der Abbildung 11.11 ausprobieren.

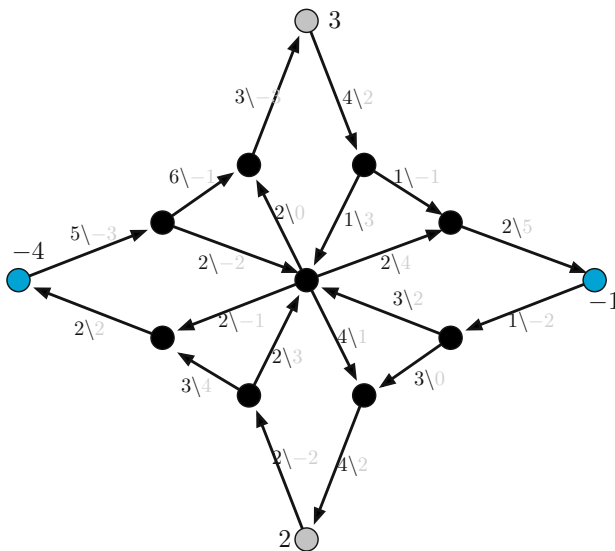


Abb. 11.11: Wie sieht ein kostenminimaler b -Fluss in diesem Netzwerk aus?

11.4 Aufgaben

Aufgabe 11.1. Betrachten Sie den gegebenen Graphen in Abbildung 11.12.

1. Welchen Wert hat der eingetragene b -Fluss f ?
2. Wie sieht der Residualgraph zu f aus?
3. Berechnen Sie einen kostenminimalen b -Fluss.

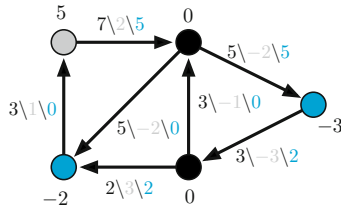


Abb. 11.12: Die oberen Kapazitäten sind schwarz, die Kosten grau und der b -Fluss blau eingezeichnet.

Aufgabe 11.2. Wie kann das Problem, einen kürzesten (s, t) -Weg bei konservativen Kosten zu finden, in ein kostenminimales Flussproblem übertragen werden?

Aufgabe 11.3. In den fünf großen Lagern eines Unternehmens sind bei weitem mehr Reifen gelagert, als zur Zeit Nachfrage bei den Kunden besteht. Einer der Lagerleiter hat nun die Möglichkeit, noch mehr Reifen zu erhalten. Die Kauf- und Lagerkosten sind dabei zu vernachlässigen. Wie wirkt sich die Anschaffung der neuen Reifen auf die Transportkosten des Unternehmens von den Lagerhallen zu den Kunden aus?

Aufgabe 11.4. Das *More-for-less-Paradoxon* von Charnes und Klingman zeigt, dass in einem Netzwerk durchaus mehr Fluss zu geringeren Kosten verschickt werden kann, auch wenn alle Kosten positiv sind. Betrachten Sie dazu das Netzwerk ohne obere Kapazitäten mit den grauen Kantenkosten in der Abbildung 11.13.

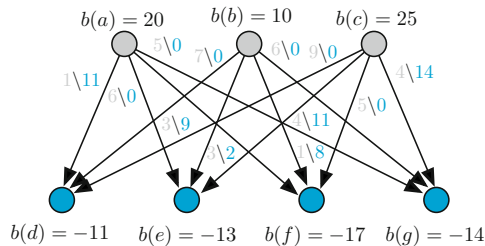


Abb. 11.13: Die Kosten sind grau und der b -Fluss blau eingezeichnet.

1. Zeigen Sie, dass $f_{(a,d)} = 11$, $f_{(a,f)} = 9$, $f_{(b,e)} = 2$, $f_{(b,f)} = 8$, $f_{(c,e)} = 11$ und $f_{(c,g)} = 14$ ein kostenminimaler Fluss ist. Welche Kosten ergeben sich für f ?

2. Welchen Wert hat ein kostenminimaler Fluss, wenn der Bedarf $b(b)$ auf 12 erhöht und die Nachfrage $b(d)$ auf -13 erniedrigt wird?

Aufgabe 11.5. In einem Netzwerk werden die Kosten mit $k \geq 0$ multipliziert. Ändert dies etwas an der Optimalität eines b -Flusses? Was passiert, wenn der Wert k zu den Kantenkosten addiert wird?

Aufgabe 11.6. In der Praxis kommt es selten vor, dass das Angebot und die Nachfrage in einem Netzwerk gleich sind. Wie können in einem solchen Fall trotzdem zulässige Balancen b mit $\sum_{v \in V} b(v) = 0$ bestimmt werden?

Aufgabe 11.7. In einem Netzwerk mit oberen Kantenkapazitäten und Kantenkosten sei e_1 die teuerste Kante und e_2 die billigste. Kann es vorkommen, dass ein kostenminimaler Fluss f keinen Fluss über die Kante e_1 schickt, d.h. $f(e_1) = 0$? Kann es andersherum passieren, dass jeder kostenminimale Fluss über die Kante e_2 Fluss fließen lässt?

Aufgabe 11.8. In einem Netzwerk $G = (V, E)$ mit oberen Kapazitäten u und Kantenkosten c ist ein kostenminimaler b -Fluss f zu Balancen b gegeben. Wie kann möglichst schnell ein kostenminimaler Fluss f' berechnet werden, wenn sich die obere Kapazität für eine Kante e um eine Einheit verringert?

Aufgabe 11.9. Sei $G = (V, E)$ ein gerichteter Graph, u obere Kapazitäten und c eine Kostenfunktion, wobei in diesem Fall die Kapazitäten auf einzelnen Kanten unbeschränkt sein kann. Zeigen Sie: Es gibt genau dann einen kostenminimalen Fluss in dem Netzwerk, wenn kein Zykel mit unbeschränkten Kapazitäten und negativen Kosten in G existiert.

Aufgabe 11.10. Sei $G = (V, E)$ ein gerichteter Graph, u obere Kantenkapazitäten, c eine Kantenkostenfunktion und b ein Knotenbewertung. Zeigen Sie: Falls $\sum_{v \in V} b(v) \neq 0$ gilt, gibt es keinen zulässigen b -Fluss.

Lösungen zu den Fragen im Text

- 46 Für eine solche Überprüfung erweitern wir den gegebenen Graphen um eine Super-Quelle s^* und eine Super-Senke t^* . Die Super-Quelle ist mit allen Quellen v mit einer (s^*, v) -Kante verbunden. Diese Kante erhält eine obere Kapazität von $b(v)$. Die Super-Senke t^* ist dann mit jeder Senke v über eine Kante mit oberen Kapazitäten von $-b(v)$ verbunden. Existiert nun ein maximaler (s^*, t^*) -Fluss vom Wert $0, 5 \cdot \sum_{v \in V} |b(v)|$, so gibt es einen zulässigen b -Fluss.
- 47 Der Algorithmus von Moore-Bellman-Ford 9.2 berechnet in einem Graphen entweder einen Kürzesten-Wege-Baum von einem Knoten s aus zu allen von ihm erreichbaren

Knoten oder einen Zykel mit negativen Kosten. Startet man nun mit einem Knoten s und berechnet keinen negativen Zykel, so bleibt noch die Knoten, die von s nicht erreicht wurden, zu überprüfen. Sind alle Knoten überprüft und es wurde kein negativer Zykel berechnet, so enthält der Graph keinen negativen Zykel.

12 Maximale Matchings

Übersicht

| | | |
|------|--|-----|
| 12.1 | Definition und ein Optimalitätskriterium | 203 |
| 12.2 | Matchings in bipartiten Graphen | 207 |
| 12.3 | Aufgaben | 218 |

12.1 Definition und ein Optimalitätskriterium

Frau Just, die Klassenlehrerin der 10b eines Mädcheninternates, ist schon seit einiger Zeit etwas unruhig: Die jährliche Klassenfahrt steht an. Dieses Mal soll es nach Berlin gehen, wo sie eine Jugendherberge ausschließlich mit Doppel- und Einzelzimmern gefunden hat.

| | Julia | Petra | Anne | Marie | Sophia | Carolin | Steffi | Beate | Pia |
|---------|-------|-------|------|-------|--------|---------|--------|-------|-----|
| Julia | | | | | | | | | |
| Petra | | | | | | | | | |
| Anne | | | | | | | | | |
| Marie | | | | | | | | | |
| Sophia | | | | | | | | | |
| Carolin | | | | | | | | | |
| Steffi | | | | | | | | | |
| Beate | | | | | | | | | |
| Pia | | | | | | | | | |

(: ja, : nein)

Tab. 12.1: Frau Just hat sich zu jedem möglichen Paar genau überlegt, ob es zwei Wochen auf einem Zimmer aushalten könnte.

Da letztes Jahr bei einer ähnlichen Veranstaltung die Zimmeraufteilung durch die Schülerinnen ziemlich im Zicken-Krieg geendet hatte (Julia und Pia reden seither nicht mehr miteinander), will sie dieses Mal selber die Mädchen auf die Zimmer verteilen. In den letzten Wochen hat sie eine Liste zusammengestellt, welche Zimmerpaarungen ihres Erachtens die zwei Wochen gut überstehen würden (Tab. 12.1). Da ein Einzelzimmer genauso viel kostet wie ein Doppelzimmer, möchte sie möglichst viele Schülerinnen auf Doppelzimmer verteilen. Wie viele Doppel- und Einzelzimmer soll sie buchen?

In der Graphentheorie gehört dieses Problem zu der Klasse der Matching-Probleme.

Definition. Sei $G = (V, E)$ ein ungerichteter Graph. Ein **Matching** in G ist eine Kantenmenge $M \subseteq E$, in der je zwei Kanten aus M keinen gemeinsamen Endknoten haben (Abb. 12.1).

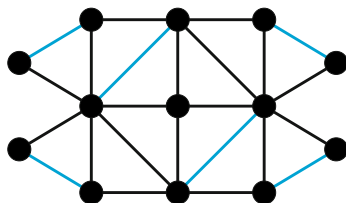


Abb. 12.1: Die blauen Kanten bilden ein Matching in G . Wie sieht ein anderes Matching in diesem Graphen aus?



Wie kann die Zimmerzuteilung als Matching-Problem formuliert werden?

Am günstigsten für die Klassenfahrt ist es, möglichst viele Doppelzimmer zu buchen. In unserem Matching-Modell ausgedrückt, bedeutet das, ein Matching mit möglichst vielen Kanten zu finden. Man spricht auch von maximalen Matchings.

Definition. Ein Matching M heißt **maximal**, wenn kein Matching M' mit mehr Kanten existiert.

In speziellen Fällen ist jeder Knoten zu einer Matching-Kante inzident. Wir nennen ein solches Matching **perfekt**.

Definition. Ein Matching M heißt **perfekt**, falls jeder Knoten in G zu einer Kante aus M inzident ist.

Das Problem, ein maximales Matching in einem Graphen zu finden, heißt auch **Maximales-Matching-Problem**. Sind die Matchings in den Graphen aus der Abbildung 12.2 maximal?⁴⁸ Wie sieht ein maximales Matching aus? Gibt es in den Beispielen nur ein maximales Matching?



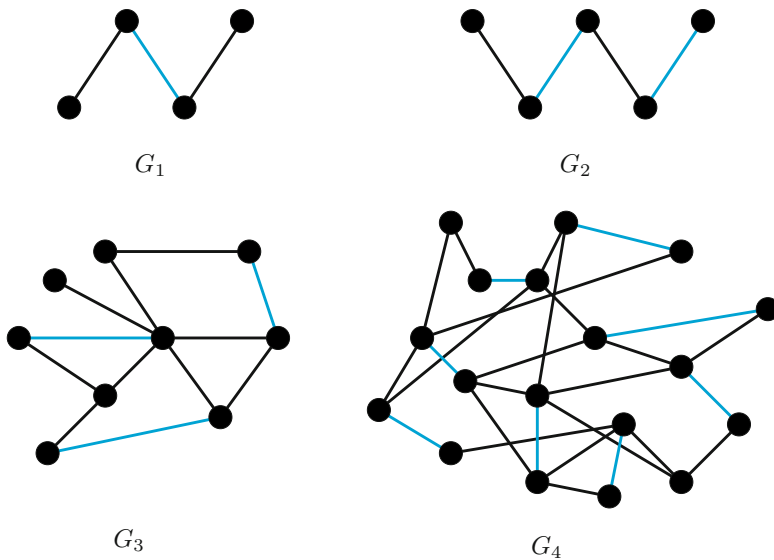


Abb. 12.2: Die Matching-Kanten sind blau eingezeichnet.

Zur Erleichterung der Suche nach einem maximalen Matching interessieren uns Kriterien, nach denen wir entscheiden können, ob ein gegebenes Matching maximal ist oder nicht. Im Graphen G_1 (Abb. 12.2) sieht man sofort, dass die Rollen von Matching- und Nicht-Matching-Kanten bloß zu tauschen sind, um ein maximales Matching zu bekommen. Im Graphen G_2 hingegen verändert ein solches Vertauschen nichts an der Größe des Matchings. Es lohnt sich auf jeden Fall, den Begriff des Vertauschens zu formalisieren.

Definition. Sei G ein Graph und M ein Matching in G . Ein M -alternierender Weg in G ist ein einfacher Weg, der abwechselnd eine Kante aus M und eine Kante, die nicht aus M ist, enthält. Ein Knoten in G heißt (bezüglich M) **exponiert**, wenn er zu keiner Kante aus M inzident ist. Einen M -alternierenden Weg nennt man **M -augmentierend**, falls beide Endpunkte exponiert sind.

Betrachten Sie noch einmal den Graphen G_2 aus der Abbildung 12.2. Dieser ist nach der Definition ein M -alternierender Weg, aber – im Gegensatz zu G_1 – nicht M -augmentierend, da nicht beide Endknoten exponiert sind. Wie sehen M -alternierende bzw. M -augmentierende Wege in den anderen Graphen aus?

Vielleicht haben Sie etwas gestutzt, als wir M -augmentierende Wege definiert haben. Schließlich hatten wir augmentierende Wege schon im Zusammenhang mit Flüssen kennen gelernt (Kapitel 10). Bei aller Unterschiedlichkeit der Definitionen soll mit „augmentieren“ auch hier auf eine Erhöhung hingewiesen werden: War es dort der Flusswert, so ist es jetzt die Anzahl der Kanten eines Matchings.



Satz 12.1. *Sei M ein Matching in G und p ein M -augmentierender Weg. Dann hat das Matching M' , das durch das Vertauschen der Matching- und Nicht-Matching-Kanten im Weg p entsteht, eine Kante mehr als M (Abb. 12.3).*

Beweis: Wir müssen als Erstes zeigen, dass die Kantenmenge M' wieder ein Matching ist. Für die Kanten bzw. Knoten, die außerhalb des Wegs p liegen, ändert sich nichts. Wie sieht es jetzt mit den Knoten auf dem Weg p aus? Kann es passieren, dass plötzlich zwei Kanten in der Menge M' enthalten sind, die einen gemeinsamen Knoten haben?

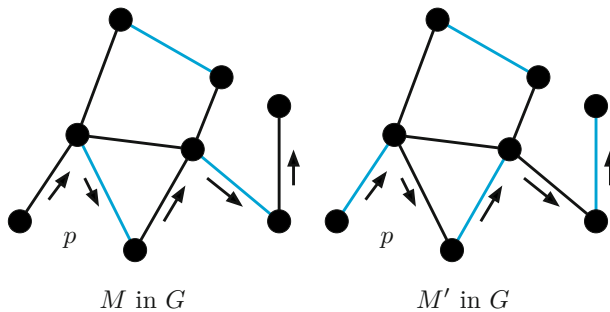


Abb. 12.3: Entlang des Wegs p werden die Nicht-Matching-Kanten zu Matching-Kanten und umgekehrt.

Die beiden Endknoten des Wegs p sind in M exponiert, d.h. in M sind sie zu keiner Kante inzident. Nach dem Vertauschen von Matching- und Nicht-Matching-Kanten sind sie nicht mehr exponiert und auch nur noch zu einer Kante inzident (Abb. 12.3). Bei anderen Knoten auf p gilt weiterhin: Jeder Knoten gehört zu genau einer Kante aus M' . Damit ist M' ein Matching und hat eine Kante mehr als M . \square

Der Satz lässt sich wieder zu einem Optimalitätskriterium erweitern, wie der Mathematiker Berge gezeigt hat.

Satz 12.2 (Berge 1957). *Ein Matching M in einem Graphen ist genau dann maximal, wenn es keinen M -augmentierenden Weg gibt.*

Beweis: Die erste Richtung haben wir schon mit dem Satz 12.1 bewiesen. Wir brauchen uns also nur um die andere Richtung zu kümmern.

Seien M und M' zwei Matchings und in G existiere kein M -augmentierender Weg. Wir werden zeigen, dass M mindestens so viele Kanten wie M' hat, d.h. $|M| \geq |M'|$. Dazu benötigen wir als Hilfsgraphen die symmetrische Differenz $M \Delta M'$ der beiden Matchings. Dieser Graph enthält nur die Kanten aus M und M' , wobei alle Kanten, die sowohl in M als auch in M' vorkommen, gelöscht werden (Abb. 12.4).

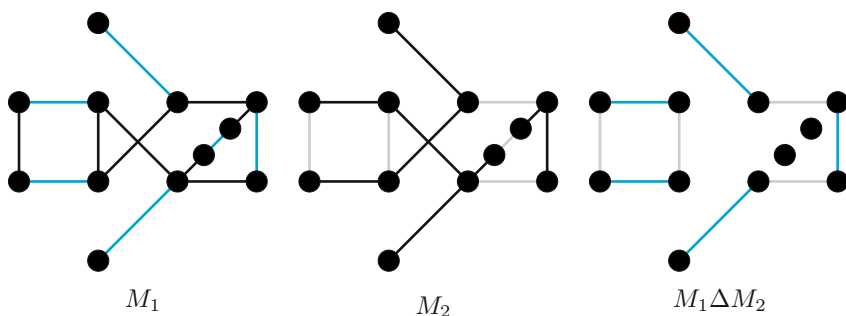


Abb. 12.4: Die symmetrische Differenz zweier Matchings M_1, M_2 enthält nur Kreise gerader Länge, alternierende Wege und isolierte Knoten.

Dann sind die Zusammenhangskomponenten von G' entweder

- isolierte Knoten
- einfache Kreise gerader Länge oder
- einfache Wege.

Wir wollen jetzt überlegen, wie viele Matching-Kanten von M und M' in den verschiedenen Komponenten vorkommen. Beginnen wir mit den elementaren Kreisen. Dort kommen genauso viele Kanten aus M wie aus M' vor, oder gibt es noch eine andere Möglichkeit, einen Kreis mit Kanten abwechselnd in M und M' zu bilden? Betrachten wir noch die einfachen Wege. Dabei ist wichtig, sich kurz klar zu machen, dass an einem Endknoten eines solchen Wegs kein „Löschvorgang“ in der symmetrischen Differenz vorgekommen sein kann. Wir unterscheiden jetzt die Wege bezüglich ihrer Anfangs- und Endkanten. Beginnt und endet ein solcher mit einer Kante aus M , dann hat M auf diesem Weg mehr Kanten als M' . Ein elementarer Weg, der mit einer Kante aus M' beginnt und endet, kann nicht existieren, da dieser Weg dann ein M -augmentierender Weg wäre. Solche Wege gibt es nach Voraussetzung in G aber nicht. Auf allen anderen Wegen befinden sich gleich viele Kanten, die zu M wie zu M' gehören. Damit hat M mindestens so viele Kanten wie M' bzw. wie jedes andere Matching. \square



Die Idee, nach und nach M -augmentierende Wege zu einem bestehenden Matching zu suchen, lässt sich wieder in einen Algorithmus umwandeln. Dieser Algorithmus wird auch als Edmonds-Matching-Algorithmus bezeichnet, ist aber etwas komplizierter. Im nächsten Abschnitt 12.2.2 werden wir einen etwas einfacheren Algorithmus kennen lernen, der in einem bipartiten Graphen ein maximales Matching berechnet.

12.2 Matchings in bipartiten Graphen

In vielen Matching-Problemen geht es darum, aus zwei verschiedenen Mengen Paare zu bilden. Typische Beispiele sind die Zuordnung von Busfahrern auf Busse

oder von Bewerbern auf Jobs oder (manchmal) von Frauen auf Männer. In dem folgenden Beispiel sollen sich Indianerstämme Totentiere aus den auf der Insel lebenden Schildkrötenarten aussuchen.

Beispiel. Auf einer einsamen Insel wohnen acht Indianerstämme. Schon vor etlichen Jahrhunderten haben die Indianer die Insel gerecht untereinander aufgeteilt: Jeder Stamm darf ein Achtel der Insel als sein Jagdgebiet benutzen (50 Quadratkilometer).

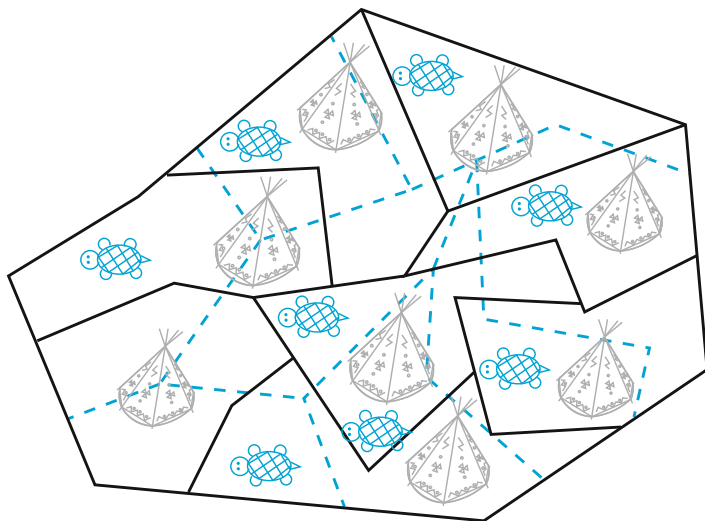


Abb. 12.5: Die Lebensräume der Schildkröten sind blau gestrichelt umrandelt und die Jagdgebiete der Indianerstämme schwarz.

Neben den Indianern haben sich auch acht unterschiedliche Schildkrötenarten auf der Insel niedergelassen. Jede dieser Art bevölkert genau 50 Quadratkilometer, wobei niemals zwei Schildkrötenarten an einer Stelle gleichzeitig vorkommen. Ein Jagdgebiet der Indianer muss natürlich nicht dem Lebensraum einer Schildkrötenart entsprechen (Abb. 12.5).

Auf ihrer letzten großen Versammlung haben die Indianer beschlossen, dass sich jeder Stamm eine eigene Schildkrötenart als Totentier wählt. Dabei sollen Teile des Lebensraums der ausgewählten Schildkröten in dem Jagdgebiet des jeweiligen Stammes liegen und natürlich können zwei Stämme nicht das gleiche Tier verehren. Ist eine solche Zuteilung möglich? ■

Diese Aufgabe kann auch wieder als Matching-Problem formuliert werden. Dazu repräsentieren die Knoten zum einen die Indianerstämme und zum anderen die Schildkröten. Ein Schildkrötenknoten wird mit einem Indianerknoten verbunden, wenn sich der Lebensraum der Schildkröte mit dem Jagdgebiet des Indianerstammes überschneidet. Wenn Sie sich den Graphen genauer anschauen, fällt Ihnen

bestimmt auf, dass dieser eine ganz spezielle Form hat: Seine Knotenmenge lässt sich in zwei Teilmengen aufteilen, sodass nur Kanten zwischen diesen Mengen, aber nicht innerhalb einer Menge verlaufen (Abb. 12.6). Sie werden auch als bipartite Graphen bezeichnet und wir hatten sie schon in Abschnitt 2.3.1 kennen gelernt. Hier zur Erinnerung die Definition.

Definition. Ein Graph $G = (V, E)$ heißt **bipartit**, wenn seine Knoten in zwei Teilmengen A und B zerlegt werden können, sodass für alle Kanten $e = (a, b) \in E$ ein Endknoten in der Menge A und einer in der Menge B liegt.

Ein bipartiter Graph $G = (V, E)$ mit den Teilmengen A und B wird auch häufig durch $G = (A \cup B, E)$ bezeichnet. Im ersten Kapitel 2.3.1 hatten wir schon eine einfache Methode kennen gelernt, einen Graphen auf diese Eigenschaft hin zu überprüfen. In den folgenden Abschnitten geht es um Matchings in bipartiten Graphen.

12.2.1 Existenz von überdeckenden Matchings

Bei dem Schildkrötenproblem wird ein perfektes Matching zwischen den Schildkröten (8 Stück) und den Indianerstämmen (8 Stück) gesucht. In anderen Fällen, z.B. bei der Zuteilung von Bewerbern auf Arbeitsplätze, muss die Anzahl der Elemente beider Mengen nicht gleich sein. Meist kommen auf ausgeschriebene Stellen mehr Bewerber als vorhandene Plätze. Gesucht ist dann eine Menge an Bewerbern, sodass jeder Arbeitsplatz besetzt werden kann. Übersetzt in die Sprache der Graphentheorie wird ein Matching zwischen den Arbeitsplätzen und den Bewerbern gesucht, das die Menge der Arbeitsplätze überdeckt (Abb. 12.6).



Definition. Sei $G = (V, E)$ ein Graph und $A \subseteq V$. Ein Matching M **überdeckt** die Knotenmenge A , wenn jeder Knoten aus A mit einer Kante aus M inzident ist.

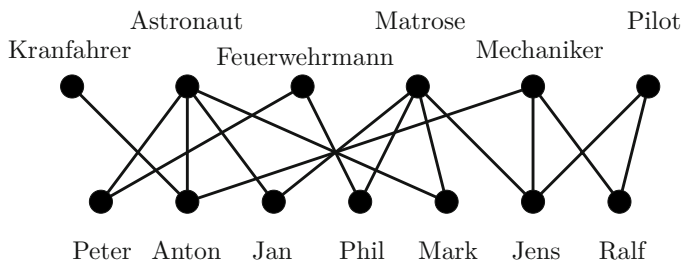


Abb. 12.6: Die Arbeitsagentur soll für die oben genannten Arbeitsplätze geeignete Kandidaten finden. Ist ein Kandidat geeignet, so ist er mit dem jeweiligen Angebot verbunden. Kann die Agentur alle Plätze vermitteln?



In einem bipartiten Graphen $G = (A \dot{\cup} B, E)$ wird häufig ein Matching gesucht, das eine der beiden Teilmengen A oder B überdeckt. Gibt es Beispiele, in denen zwar die Menge A , aber nicht die Menge B überdeckt werden kann? Welche Eigenschaft muss ein bipartiter Graph erfüllen, in dem sowohl die Menge A als auch die Menge B durch ein Matching überdeckt wird? Der Mathematiker Hall hat sich länger mit diesen Fragen auseinandergesetzt und schließlich ein allgemeines Kriterium gefunden, das besagt, wann ein überdeckendes Matching in einem bipartiten Graphen existiert. Die Idee dabei ist die Folgende: Wenn für jede Teilmenge $X \subseteq A$ genügend viele Partner in B vorhanden sind, dann existiert auch ein A überdeckendes Matching. Die **Partner** $\Gamma(X)$ einer Menge X sind alle Knoten in B , die mit X durch eine Kante verbunden sind, also

$$\Gamma(X) = \{y \in B \mid \text{es gibt ein } x \in X \text{ mit } (x, y) \in E\}.$$

Mit dieser Schreibweise lässt sich der Satz von Hall einfach formulieren.

Satz 12.3 (Hall 1935). *Sei $G = (A \dot{\cup} B, E)$ bipartit. Dann sind folgende Aussagen äquivalent:*

1. *Der Graph G hat ein Matching, das A überdeckt.*
2. *Für jede Menge $X \subseteq A$ gilt*

$$|X| \leq |\Gamma(X)| \quad (\text{Heiratsbedingung}).$$

Ein Spezialfall mit $|A| = |B|$ ist auch als Heiratssatz bekannt und wurde schon 1917 von Frobenius bewiesen.



Beweis des Satzes von Hall 12.3. (1) \Rightarrow (2): Wenn Sie davon ausgehen, dass Ihnen ein Matching gegeben ist, das A überdeckt, ist der Rest nicht mehr schwierig.⁴⁹ (2) \Rightarrow (1): Wir setzen voraus, dass die Heiratsbedingung, d.h. $|X| \leq |\Gamma(X)|$ für jede Teilmenge $X \subseteq A$ erfüllt ist, und konstruieren ein A überdeckendes Matching. Sei M ein beliebiges Matching, das A nicht ganz überdeckt. Wir werden nun zeigen, dass in diesem Fall ein M -augmentierender Weg existiert. Dazu erstellen wir eine Folge an Knoten. Diese Folge an sich ist noch kein M -augmentierender Weg, aber sie enthält einen. Wir bestimmen sie folgendermaßen:

Schritt 1: Wählen Sie einen Knoten a_0 , der zu keiner Matching-Kante inzidiert.

Schritt 2: Wählen Sie einen Knoten b_i , der zu einem Knoten $a_{f(i)} \in \{a_0, \dots, a_{i-1}\}$ benachbart ist und noch nicht in der Folge vorkommt, d.h. $b_i \neq \{b_1, \dots, b_{i-1}\}$.

Schritt 3: Falls b_i inzident zu einer Matching-Kante (a, b_i) ist, wählen Sie a als nächstes Folgeelement, d.h. $a_i = a$, erhöhen Sie i um eins und gehen Sie zurück zu Schritt 2. Ist b_i zu keiner Matching-Kante inzident, endet die Folge (Abb. 12.7).

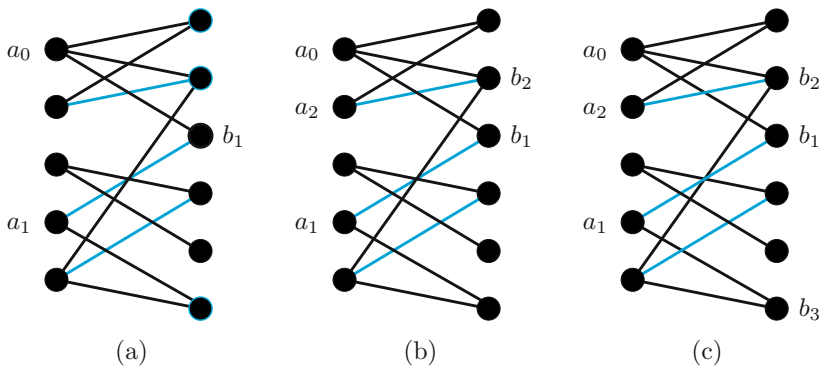


Abb. 12.7: Nachdem wir die Knoten a_0 und b_1 gewählt haben, ergibt sich der Knoten a_1 automatisch. Im nächsten Schritt können wir einen beliebigen der blau umrandeten Knoten auswählen (Teil (a)). Nehmen wir b_2 , dann erhalten wir sofort a_2 (Teil (b)). Nun kann man b_3 als Nachbar von a_1 wählen, also $f(3) = 1$. Mit dieser Wahl stoppt die Folge, da b_3 mit keiner Matching-Kante verbunden ist (Teil (c)).

Aus dieser Folge von Knoten bauen wir jetzt einen M -augmentierenden Weg. Dazu zeigen wir zunächst, dass die Folge wirklich in B endet und wir nicht schon irgendwann in Schritt 2 keinen geeigneten Knoten mehr finden: Die i Knoten $\{a_0, \dots, a_{i-1}\}$ haben nach der Heiratsbedingung mindestens i Nachbarn in B . Von diesen Nachbarn sind bisher nur $i - 1$ in der Folge enthalten, nämlich b_1, \dots, b_{i-1} . Also gibt es einen weiteren Knoten b_i , der die zweite Bedingung erfüllt. Somit muss der letzte Knoten der Folge aus B sein (sonst könnten wir die Folge noch einmal verlängern).

Sei $b_k \in B$ der letzte Knoten der Folge. Jetzt bestimmen wir den M -augmentierenden Weg einfach rückwärts aus der Folge, indem wir als Nächstes zu dem Knoten $a_{f(k)}$ gehen. Wenn $f(k) \neq 0$ gilt, dann wurde $a_{f(k)}$ in der $f(k)$ -ten Iteration ausgewählt, weil die Kante $(a_{f(k)}, b_{f(k)})$ in M enthalten ist. Wir gehen also zum Knoten $b_{f(k)}$. Dieser hatte wieder einen Nachbarn in der Menge $\{a_0, \dots, a_{f(k)-1}\}$ usw. (Abb. 12.8).

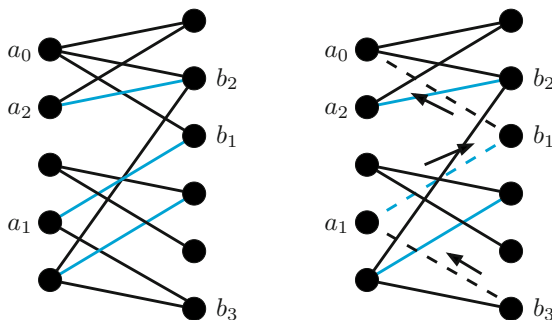


Abb. 12.8: Aus der Folge $a_0, b_1, a_1, b_2, a_2, b_3$ lässt sich einfach der augmentierende Weg b_3, a_1, b_1, a_0 bestimmen.

Irgendwann enden wir im Knoten a_0 , der nach der Wahl in Schritt 1 nicht überdeckt ist. Die Kanten, die beim Übergang von den a s zu den b s benutzt werden, sind immer Matching-Kanten. Die, die beim Übergang von den b s zu den a s benutzt werden, können hingegen keine Matching-Kanten sein. Somit sind die Endknoten des M -alternierenden Wegs a_0 und b_k exponiert, woraus folgt, dass dieser einen M -augmentierenden Weg bildet.

Wenn wir entlang dieses Wegs das Matching M verändern, d.h. die Matching- und Nicht-Matching-Kanten vertauschen, erhalten wir ein Matching, das einen Knoten mehr von A überdeckt. Wenn A immer noch nicht komplett überdeckt ist, so suchen wir wieder einen exponierten Knoten, bezeichnen diesen mit a_0 und starten die ganze Prozedur von vorne. Das machen wir so lange, bis ein Matching A vollständig überdeckt. \square

Die Schwierigkeit des Beweises liegt in den unterschiedlichen Konstruktionen. Doch die Mühe lohnt sich; schließlich entsteht eine konkrete Konstruktionsanleitung, ein A überdeckendes Matching zu berechnen. Fassen Sie die Anleitung doch noch einmal zusammen und wenden Sie sie auf den Graphen, das Matching und den Startknoten a_0 in Abb. 12.9 an. Wie viele Iterationen sind noch notwendig, um ein A überdeckendes Matching zu finden? Muss man dazu wirklich ein solches Matching berechnen und die Iterationen zählen?⁵⁰

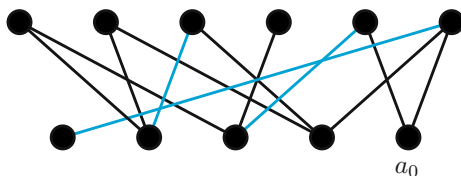


Abb. 12.9: Wie sieht eine Knotenfolge aus, die mit a_0 startet? Sind die Folgen immer eindeutig? Wenn nein, wie viele unterschiedliche finden Sie in diesem Fall?

Aus dem Satz von Hall folgt sofort eine Aussage für k -reguläre bipartite Graphen.

Definition. Ein Graph heißt **k -regulär**, wenn jeder Knoten den Grad k hat.

Ist ein bipartiter Graph k -regulär, dann wissen wir, dass er mindestens ein perfektes Matching besitzt.

Lemma 12.4. Sei $G = (A \cup B, E)$ ein k -regulärer bipartiter Graph. Dann hat G ein perfektes Matching.

Beweis: Damit in einem bipartiten Graphen ein perfektes Matching existieren kann, muss zunächst $|A| = |B|$ gelten. Wir zählen also die Kanten E_A , die von A ausgehen, und die Kanten E_B , die von B ausgehen. Da alle Kanten zwischen A und B verlaufen, gilt

$$|E_A| = |E_B|,$$

und da alle Knoten den Grad k haben, erhalten wir

$$k \cdot |A| = |E_A| = |E_B| = k \cdot |B| \quad \Rightarrow \quad |A| = |B|.$$

Jetzt können wir also den Satz von Hall bzw. den Heiratssatz anwenden. Falls ein A überdeckendes Matching existiert, ist es auch perfekt. Sind denn immer genügend Partner $\Gamma(X)$ für eine Teilmenge $X \subseteq A$ vorhanden? \square



Gehen wir noch mal zurück zu den Schildkröten. Wenn wir nun wissen würden, dass in jedem Jagdgebiet eines Indianerstammes k verschiedene Schildkrötenarten vorkommen und dass jede Schildkrötenart k Jagdgebiete streift, könnten wir eine Zuordnung vornehmen. Wie aber sieht es im allgemeinen Fall aus? Können Sie jetzt die Schildkrötenfrage beantworten? Wie kann man den Satz von Hall hier anwenden?⁵¹



Mit dem Satz von Hall haben wir schon einen wichtigen Satz zu Matching in bipartiten Graphen kennen gelernt. Was passiert aber, wenn es kein perfektes Matching gibt und wir ein maximales Matching bestimmen wollen? Können wir eine gute obere Schranke für die Anzahl der Matching-Kanten finden? Mit den beiden Fragen, beschäftigen wir uns im nächsten Abschnitt.

12.2.2 Der Satz von König

Im ersten Abschnitt über Matchings hatten wir gezeigt, dass ein Matching M genau dann maximal ist, wenn es keinen M -augmentierenden Weg gibt. Dazu wissen wir: In manchen Fällen findet man schnell einen M -augmentierenden Weg, in anderen Fällen kann das sehr lange dauern. Immerhin müssen wir allen M -alternierenden Wegen nachgehen. Wie sieht es mit dem Graphen in Abbildung 12.10 aus?

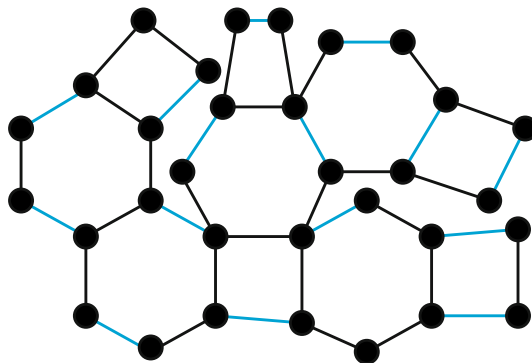


Abb. 12.10: Ist der Graph bipartit?

Es gibt noch eine andere Möglichkeit festzustellen, ob das gegebene Matching maximal ist oder nicht. Dazu betrachten wir, ähnlich wie bei maximalen Flüssen, ein anderes „gegenläufiges“ Problem.

Definition. Sei $G = (V, E)$ ein Graph. Eine **Knotenüberdeckung** (engl. *vertex cover*) ist eine Knotenmenge $V' \subseteq V$, sodass jede Kante von G mindestens einen Endpunkt in V' hat (Abb. 12.11).

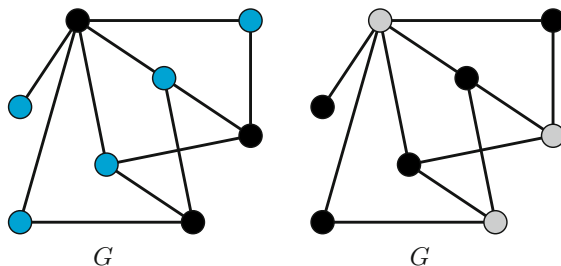


Abb. 12.11: Sowohl die blauen als auch die grauen Knoten überdecken alle Kanten.



Anstelle von Kanten werden jetzt Knoten gesucht. Eine Knotenüberdeckung mit möglichst vielen Knoten zu finden, ist einfach. Wie viele Knoten hat eine solche Menge? Wie viele Knoten werden aber mindestens benötigt? Die Frage ist viel spannender und nicht so einfach zu beantworten. Wie sieht es bei den Graphen aus Abbildung 12.2 oder 12.10 aus? Wir suchen jetzt also eine minimale Knotenmenge, die alle Knoten überdeckt.

Definition. Eine Knotenüberdeckung V' ist **minimal**, wenn es keine andere Knotenüberdeckung \tilde{V} gibt, die weniger Knoten hat als V' .



Fällt Ihnen eine Anwendung ein, in der die Suche nach einer minimalen Knotenüberdeckung Sinn macht?⁵² Neben vielen Problemen aus der Praxis, die mit Hilfe von Knotenüberdeckungen gelöst werden können, erhalten wir ein weiteres Optimalitätskriterium für ein maximales Matching in bipartiten Graphen.

Satz 12.5 (König 1931). *In einem bipartiten Graphen G ist die Kardinalität eines maximalen Matching $\nu(G)$ gleich der Kardinalität einer minimalen Knotenüberdeckung $\tau(G)$.*



Beweis: Als Erstes zeigen wir, dass $\tau(G) \geq \nu(G)$ gilt. Diese Richtung ist wieder relativ einfach. Wenn M ein maximales Matching ist, wie viele Knoten muss eine Knotenüberdeckung dann mindestens haben?⁵³ Damit sind wir auch schon fertig. Bleibt $\nu(G) \geq \tau(G)$ zu zeigen. Zur Konstruktion eines Matchings wandeln wir das Matching-Problem in ein Maximales-Fluss-Problem um. Sei $G = (A \dot{\cup} B, E)$ der bipartite Graph. Aus diesem ungerichteten Graphen generieren wir uns einen gerichteten, indem wir alle Kanten von A nach B richten. Anschließend fügen wir dem Graphen noch einen Knoten s und einen Knoten t hinzu und verbinden s mit allen Knoten in A und alle Knoten in B mit t . Die Kanten zwischen s und A sind

von s weggerichtet, die Kanten zwischen B und t zu t hin. Den neuen Graphen nennen wir G' und ergänzen ihn um obere Kapazitäten von 1 auf allen Kanten (Abb. 12.12).

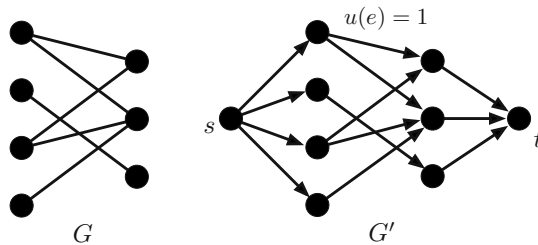


Abb. 12.12: Jedes Matching-Problem auf einem bipartiten Graphen kann einfach in ein Maximales-Fluss-Problem umgewandelt werden.

Anschließend berechnen wir einen maximalen (s, t) -Fluss f in G' so, dass er nur die Werte 0 und 1 an jeder Kante annimmt. Ein solcher Fluss existiert nach Lemma 10.3. Dann definieren die Kanten von A nach B , über die Fluss geschickt wird, ein Matching M . Warum ist das so? Der Flusswert von f entspricht dabei der Anzahl der Kanten in M .



Als Nächstes definieren wir die Knotenmenge S mit

$$S := \{s\} \cup \{v \in A \cup B \mid v \text{ ist von } s \text{ in } G'_f \text{ erreichbar}\}$$

(Abb. 12.13). Mit G'_f hatten wir den Residualgraphen bezeichnet (Definition auf Seite 10.2). Da hier die Flusswerte nur 0 oder 1 sind, enthält G'_f alle Kanten aus G' , über die kein Fluss fließt und die Rückwärtskanten zu den Kanten, über die Fluss fließt.

1. Beobachtung: Die Menge $V' = (A \setminus S) \cup (B \cap S)$ ist eine Knotenüberdeckung in G .

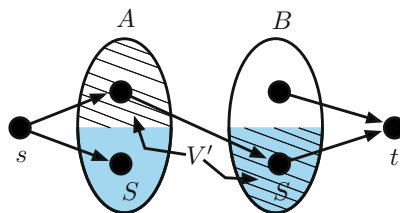


Abb. 12.13: Die Menge S (blau markiert) kann von s aus im Residualgraphen G^f erreicht werden. Die Menge V' enthält alle schraffierten Knoten.

Beweis: Wir müssen zeigen, dass jede Kante aus G einen Endknoten in der Menge V' hat (Abb. 12.13). Angenommen, die Kante $e = (a, b)$ mit $a \in A$ und $b \in B$ besitzt keinen Endknoten in V' . Dann liegt der Knoten $a \in S$ und der Knoten $b \in B \setminus S$. Ansonsten wäre $a \in A \setminus S$ oder $b \in B \cap S$ und somit wäre die Kante überdeckt (Abb. 12.14).

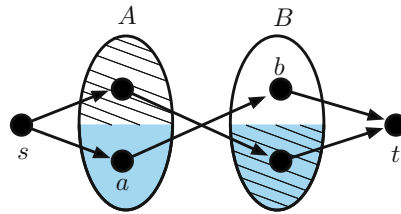


Abb. 12.14: Wir werden zeigen, dass die Kante (a, b) nicht existieren kann.

Wir unterscheiden jetzt zwei Fälle:

Fall 1: Über die Kante e fließt Fluss, d.h. $f(e) = 1$ (Abb. 12.15, Fall 1). Da $a \in S$, ist a von s aus im Residualgraphen G^f erreichbar. Das kann auf zwei unterschiedliche Weisen geschehen.

Fall a: Der Knoten a kann im Residualgraphen G^f direkt über die Kante (s, a) erreicht werden. Dann gilt $f((s, a)) = 0$, d.h. es fließt keine Einheit in den Knoten a rein. Aber es fließt eine Einheit über die Kante (a, b) aus dem Knoten heraus. Damit ist aber die Flusserhaltung in diesem Knoten verletzt. Ein solcher Fall kann nicht eintreten (Abb. 12.15, Fall a).

Fall b: Der Knoten a wird über die Kante (b', a) im Residualgraphen erreicht. Dann wird aber ebenfalls der Knoten b' von s aus erreicht und somit gilt $b' \in S$. Weiter ist (b', a) eine Rückwärtskante (alle Vorwärtskanten sind ja von A nach B gerichtet). Die Rückwärtskante ist nur in G^f enthalten, wenn Fluss über die Vorwärtskante fließt, d.h. $f(a, b') = 1$. Damit fließen aber mindestens zwei Flusseinheiten aus a heraus, während nur eine hineinfließt. Auch hier wird die Flusserhaltung wieder verletzt (Abb. 12.15, Fall b).

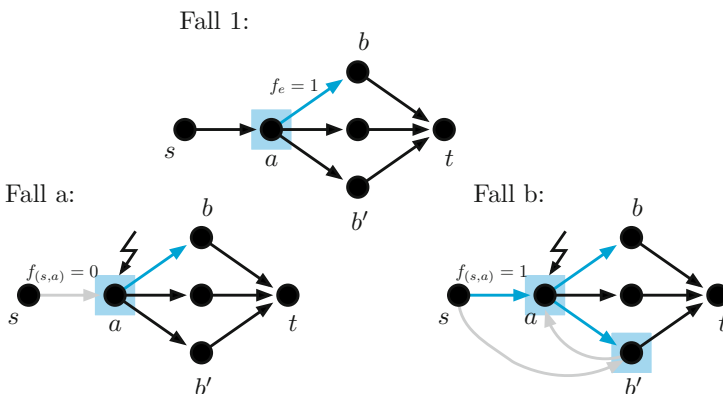


Abb. 12.15: Die blauen Kanten geben an, ob Fluss über die Kante fließt. Die grauen Kanten sind im Residualgraphen enthalten.

Fall 2: Über die Kante e fließt kein Fluss, d.h. $f(e) = 0$. Dann ist die Kante (a, b) in G^f enthalten. Da a von s in G^f erreichbar ist, kann auch der Knoten b von s aus erreicht werden. Wir hatten aber schon festgestellt, dass b nicht in S liegen darf (Abb. 12.16, Fall 2).

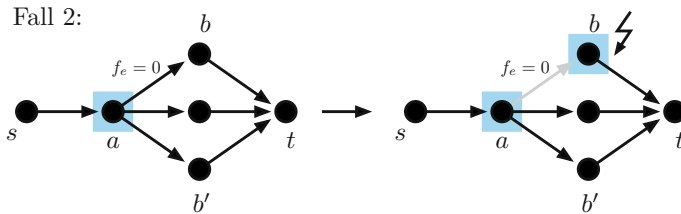


Abb. 12.16: Die blauen Kanten geben an, ob Fluss über die Kante fließt. Die grauen Kanten sind im Residualgraphen enthalten.

Damit haben wir gezeigt, dass die Menge V' eine Knotenüberdeckung in G ist. Sie muss nicht unbedingt auch eine Knotenüberdeckung für G' sein. Fällt Ihnen ein Beispiel dazu ein? □

Als Letztes müssen wir noch einen Zusammenhang zwischen der Anzahl der Knoten in V' und der Anzahl der Matching-Kanten finden. Auf den Kanten (s, a) mit $a \in A \setminus S$ (d.h. $a \in V'$) fließt Fluss, ansonsten wäre a im Residualgraphen von s aus erreichbar. Außerdem fließt Fluss auf den Kanten (b, t) mit $b \in A \cap S$, d.h. $b \in V'$, da sonst der Knoten t im Residualgraphen von s aus erreichbar und somit der Fluss f nicht maximal wäre (Satz 10.5). Weiter kann auf den Kanten (a, b) mit $a, b \in V'$ kein Fluss fließen, da sonst der Residualgraph die Kante (b, a) enthalten würde und somit auch der Knoten a von s aus zu erreichen wäre (Abb. 12.17).

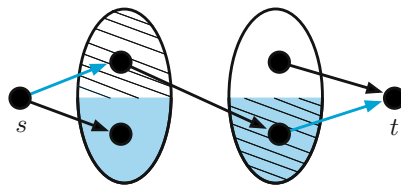


Abb. 12.17: Auf (s, a) mit $a \in A \setminus S$ und auf (b, t) mit $b \in B \cap S$ fließt Fluss.

Demnach entspricht der Flusswert von f der Anzahl der Knoten in V' und wir erhalten

$$\nu(G) = \text{value}(f) = |V'| \geq \tau(G).$$

Die letzte Ungleichung ist gültig, da wir an sich nicht sicher wissen, ob wir eine minimale Knotenüberdeckung mit V' gefunden haben oder nicht.

Da nun $\tau(G) \geq \nu(G)$ (Teil 1) und $\nu(G) \geq \tau(G)$ (Teil 2) gilt, folgt die Aussage des Satzes von König

$$\nu(G) = \tau(G).$$

□

Auch in diesem Beweis sind die einzelnen Konstruktionen etwas komplizierter. Außerdem muss man sich genau überlegen, wie der Residualgraph definiert war. Auf der anderen Seite haben wir eine Möglichkeit kennen gelernt, mit Hilfe eines maximalen Flusses ein maximales Matching zu berechnen.

Satz 12.6. *Sei $G = (A \dot{\cup} B, E)$ ein bipartiter Graph und $G' = (V', E')$ mit $V' = A \dot{\cup} B \cup \{s, t\}$ und $E' = E \cup \{(s, v) \mid v \in A\} \cup \{(v, t) \mid v \in B\}$. Sei f ein maximaler Fluss in G' mit den $f(e) \in \{0, 1\}$. Dann ist*

$$M = \{e \in E \mid f(e) = 1\}$$

ein maximales Matching.

Wenn Sie das nächste Mal also Freunde oder Geschäftspartner zum Essen einladen, Bewerber möglichst gut auf freie Stellen verteilen oder Aufgaben an andere delegieren wollen, können Ihnen die hier behandelten Methoden zur Seite stehen.

12.3 Aufgaben

Aufgabe 12.1. Ein Projekt soll von sieben Mitarbeiterinnen und Mitarbeitern durchgeführt werden. Sie haben unterschiedliche Kompetenzen, die (vom Aufwand her ähnlichen) Teilprojekte A bis G auszuführen. Wer für welche Aufgabe geeignet ist, sehen Sie an der folgenden Tabelle (Tab. 12.2). Welcher Mitarbeiter soll welches Teilprojekt leiten?

| Name | Projekt | Name | Projekt |
|-------------|----------------|--------------|----------------|
| Herr Baier | A, D und E | Frau Schmidt | A, D und F |
| Frau Müller | C und G | Herr Kunze | E und G |
| Frau August | A und B | Herr Hof | A und C |
| Frau Lase | D und F | | |

Tab. 12.2: Frau Müller könnte sowohl das Projekt C als auch das Teilprojekt G durchführen.

Aufgabe 12.2. Ist das Matching in Abbildung 12.18 maximal?

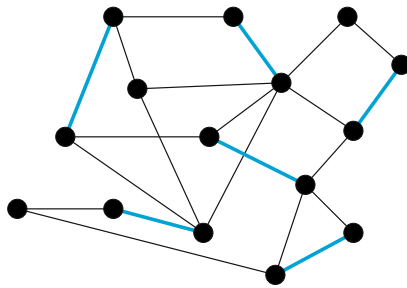


Abb. 12.18: Die Matching-Kanten sind blau gezeichnet.

Aufgabe 12.3. Welche Schachbretter lassen sich komplett mit Dominosteinen überdecken (Abb. 12.19)?

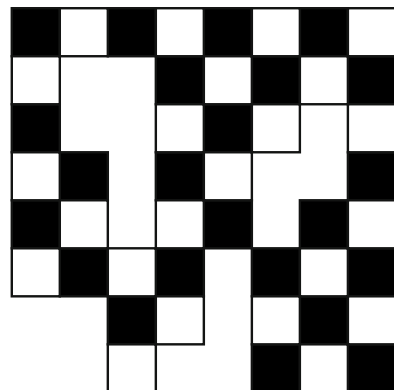
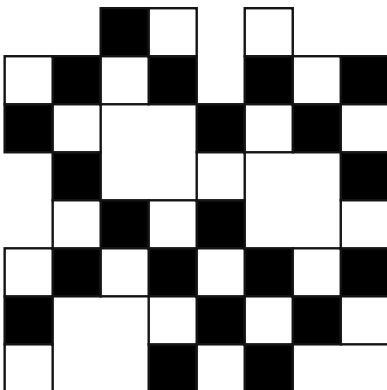
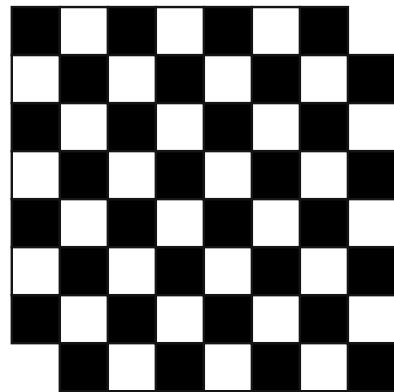
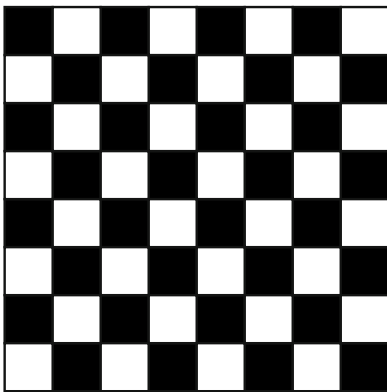


Abb. 12.19: Aus einigen Schachbrettern wurden Felder gelöscht.

Aufgabe 12.4. Von der Universität werden sechs unterschiedliche Workshops mit jeweils zehn Plätzen angeboten. Für die Workshops haben sich 80 interessierte Studenten beworben. Wie kann eine Zuordnung zwischen den Studenten und den

Workshops bestimmt werden, ohne ein Matching zwischen den Studenten und den einzelnen Plätzen der Workshops festzulegen?

Aufgabe 12.5. Sei $G = (V, A)$ ein Graph. Beweisen Sie:

1. Falls G ein perfektes Matching enthält, dann ist $|V|$ gerade.
2. Jeder hamiltonsche Graph mit einer geraden Anzahl an Knoten enthält ein perfektes Matching.
3. Der Petersen-Graph hat perfektes Matching.

Aufgabe 12.6. Beweisen oder widerlegen Sie:

1. In jedem Kreis C_n existiert ein perfektes Matching.
2. In jedem Baum gibt es maximal ein perfektes Matching.

Aufgabe 12.7. Gibt es einen k -regulären Graph mit $k \geq 2$ und einer geraden Anzahl an Knoten, der kein perfektes Matching enthält?

Aufgabe 12.8. Sei $G = (A \dot{\cup} B, E)$ ein bipartiter Graph und es gelte $d(a) \geq k \geq d(b)$ für alle Knoten $a \in A$ und $b \in B$ und eine Konstante $k \geq 1$. Dann existiert in G ein A -überdeckendes Matching.

Aufgabe 12.9. Auf eine Stellenausschreibung von 30 Tutorienjobs haben sich auch 50 Bewerber gefunden. Jeder Bewerber ist höchstens für zwei Jobs qualifiziert, allerdings kann jeder Job von mindestens drei Bewerbern ausgeführt werden. Können im nächsten Semester trotzdem alle Tutorien stattfinden?

Aufgabe 12.10. Seien M und N zwei Matchings in einem Graphen G mit $|M| > |N|$. Dann gibt es zwei Matchings M' und N' in G mit $|M'| = |M| - 1$ und $|N'| = |N| + 1$, sodass die symmetrische Differenz $M \Delta N$ und $M' \Delta N'$ gleich sind. (Die symmetrische Differenz zweier Matchings enthält alle Kanten, die entweder in dem einen oder in dem anderen, aber nicht in beiden enthalten sind.)

Aufgabe 12.11. Betrachten Sie das folgende Zwei-Personen-Spiel: Gegeben ist ein Graph $G = (V, E)$. Die erste Person wählt einen beliebigen Knoten $v \in V$. Der zweite Spieler darf jetzt einen der noch nicht besuchten Nachbarknoten von v wählen. Anschließend ist der erste Spieler wieder dran, einen von diesem neuen Knoten benachbarten Knoten auszusuchen (Abb. 12.20). Wer als letztes einen solchen Knoten auswählen kann, hat gewonnen. Zeigen Sie: Falls G ein perfektes Matching besitzt, kann der zweite Spieler immer gewinnen.

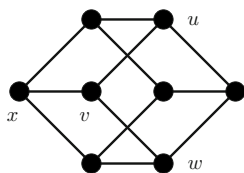


Abb. 12.20: Nachdem Spieler 1 den Knoten v gewählt hat, kann Spieler 2 entweder den Knoten u, w oder x nehmen. Die ausgewählten Knoten in G mit den dazugehörigen Kanten bilden nach und nach einen Weg.

Lösungen zu den Fragen im Text

- 48 G_1 ist nicht maximal, G_2 ist maximal bzw. perfekt, G_3 ist nicht maximal, es existiert kein perfektes Matching, G_4 ist nicht maximal (zwischen den beiden nicht gematchten Knoten gibt es einen M -alternierenden Weg.)
- 49 Sei X eine beliebige Teilmenge von A . Dann existiert zu jedem Knoten $x \in X$ eine Kante $(x, y) \in M$. Da M ein Matching ist, gilt

$$x \neq x' \in X \Rightarrow y \neq y' \in B$$

mit $(x, y), (x', y') \in M$. Daraus ergibt sich

$$Y = \{y \in B \mid (x, y) \in E, x \in X\} \subseteq \Gamma(X)$$

und damit

$$|X| = |Y| \leq |\Gamma(X)|.$$

- 50 Es sind noch 2 Iterationen notwendig. Allerdings muss dazu der Algorithmus nicht bis ans Ende durchgeführt werden. Pro Iteration wird ein Knoten mehr aus A gematcht. Da noch zwei Knoten keinen Partner haben, benötigt man nur noch zwei Iterationen. Die Folgen sind nicht immer eindeutig, da im Schritt 2 ein beliebiger Knoten gewählt werden kann.
- 51 Nach dem Schubfachprinzip gilt: Auf dem Gebiet von k Indianerstämmen müssen mindestens k unterschiedliche Schildkrötenarten vertreten sein. Damit ist die Bedingung vom Satz von Hall erfüllt und wir wissen, dass ein perfektes Matching existiert.
- Im zweiten Fall muss nicht unbedingt ein Matching existieren. Eine Art könnte z.B. zwei Gebiete der Indianer abdecken, aber nur einem Stamm könnten diese Art als Totem zugeordnet werden.
- Falls mehr als acht Schildkrötenarten auf der Insel wohnen, hängt es immer noch von ihrer Landverteilung ab. Sobald sie gleichmäßig verteilt sind, gibt es immer ein Matching. Ansonsten nicht unbedingt.
- 52 Man könnte sich ein Straßennetz und eine Menge an Polizisten vorstellen, die alle Straßen überwachen sollen. Wie viele werden dazu mindestens benötigt?
- 53 Jede Knotenüberdeckung muss mindestens einen Endknoten pro Matching-Kante enthalten.

13 Lösungshinweise

Lösungshinweise zu Kapitel 1

Zu 1.1: 1. ist kein Graph, da eine Kante nur aus zwei Knoten besteht, und 3. ist kein Graph, da der Knoten f nicht in V , sondern nur in E enthalten ist. 2. und 4. sind Graphen.

Zu 1.2: Andy streitet sich am häufigsten (als Graphen dargestellt, bestimmt er den Maximalgrad) und Mark ist am friedlichsten (er entspricht einem Knoten mit Minimalgrad).

Zu 1.3: Die zu v inzidenten Kanten sind blau eingezeichnet (Abb. 13.1). Die drei Nachbarn von w sind ebenfalls blau gefärbt. Der Knoten δ hat nur einen Grad von 1 und ist somit minimal. Der Knoten Δ hat den maximalen Grad von 5. Der Graph enthält eine Brücke (grau eingezeichnet). Die drei Artikulationsknoten sind grau eingezeichnet. Die grau gestrichelten Kanten sorgen dafür, dass der Graph zusammenhängend wird. Die minimale Anzahl entspricht der Anzahl der Zusammenhangskomponenten minus eins. Der dunkelblaue Kreis hat eine Länge von 4.

Zu 1.4: Wir erhalten den vollständigen bipartiten Graphen $K_{7,6}$, wobei die eine Menge die Jungen darstellt und die andere die Mädchen. Insgesamt ergeben sich somit $7 \cdot 6 = 42$ Kanten, wobei die Mädchenknoten einen Grad von 7 haben und die Jungenknoten einen Grad von 6.

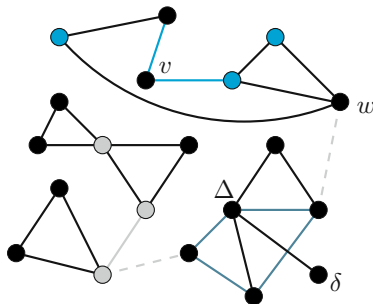


Abb. 13.1: Die Lösungen in dem Graphen.

Zu 1.5: 1. In den einfachen Kreisen C_k mit $k = 4, 5, 6$ hat jeder Knoten den Grad zwei. 2. Der K_4 erfüllt für vier Knoten das gewünschte. Für fünf Knoten existiert nach dem *Handshaking-Lemma* kein Graph, in dem alle Knoten einen Grad von 3 haben. Für 6 Knoten gibt es mehrere Möglichkeiten (Abb. 13.2).

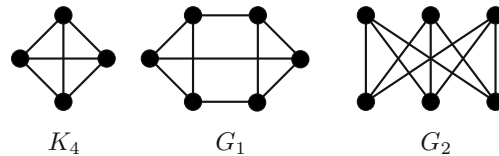


Abb. 13.2: Die Graphen G_1 und G_2 haben 6 Knoten und es gilt $d(v) = 3$ für alle $v \in V$.

Zu 1.6: 1. Ja; 2. nein, da die Anzahl der Knoten mit ungeradem Grad gerade ist; 3. Ja; 4. nein, weil der Graph nur 4 Kanten enthält und mindestens $n - 1 = 5$ Kanten für den Zusammenhang benötigt werden.

Zu 1.7: 2 Blätter: Weg der Länge 7; 4 Blätter: Weg der Länge 5 und zwei Kanten an den mittleren Knoten; 7 Blätter: ein Stern; 8 Blätter: existiert nicht.

Zu 1.8: In einem Baum bildet jede Kante eine Brücke. Da ein Baum $n - 1$ Kanten hat, gibt es in jedem Baum auch $n - 1$ Brücken. Auch jeder Knoten, außer den Blättern, ist ein Artikulationspunkt.

Zu 1.9: Sei v ein Knoten mit $d(v) = \Delta$. Dann kann jede zu v inzidente Kante zu einem in v startenden Weg verlängert werden, bis dieser in einem Blatt endet.

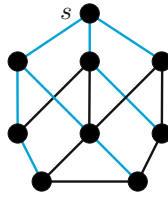
Zu 1.10: Ein Baum ist zusammenhängend. Da $a, b \in V$, gibt es einen Weg von a nach b . Angenommen, dieser wäre nicht eindeutig. Dann würden Teile der zwei Wege einen Kreis bilden. Das ist ein Widerspruch zur Kreisfreiheit von T .

Zu 1.11: Er kann maximal $\frac{(n-1) \cdot (n-2)}{2}$ Kanten enthalten. Begründung: Betrachten wir einen Graphen G mit maximal vielen Kanten, der nicht zusammenhängend ist. Dieser Graph enthält genau zwei Zusammenhangskomponenten mit n_1 und n_2 Knoten. In beiden Komponenten sind alle Knoten miteinander verbunden, d.h.

$$|E(G)| = \frac{n_1 \cdot (n_1 - 1)}{2} + \frac{n_2 \cdot (n_2 - 1)}{2}.$$

Diese Anzahl der Kanten ist für $n_1 = 1$ und $n_2 = n - 1$ maximal.

Zu 1.12: Nein, aufgrund des Schubfachprinzips. Angenommen, es gibt einen Graphen mit n Knoten. Falls ein Knoten v_0 den Grad $n - 1$ hat, so sind alle Knoten mit diesem verbunden. Deswegen kann kein Knoten vom Grad 0 in dem Graphen existieren. Damit bleiben nur $n - 1$ unterschiedliche Gradzahlen für n Knoten. Dann gibt es aber zwei Knoten mit gleichem Grad. Diese Argumentation gilt auch, falls ein Knoten den Grad 0 hat bzw. der Graph weder die Grade $n - 1$ noch 0 enthält.



Lösungshinweise zu Kapitel 3

Zu 3.1: 1. Die Stadt sucht einen minimal spannenden Baum. Dabei entstehen Kosten von 130.000,00 Euro. 2. Die Addition von einem konstanten Wert c zu jeder Kante ändert nichts an der Optimalität des Baums, da jeder Baum einen Zuschlag von $(n - 1) \cdot c$ Fixkosten erhält. Daraus ergeben sich die neuen Kosten für die Stadt von 141.000,00 Euro. 3. Auch die Multiplikation von Kantenkosten mit einem positiven Wert verändert nichts an der Optimalität des Baumes.

Zu 3.2: 1. Der Algorithmus berechnet den schwarz eingezeichneten spannenden Baum (Abb. 13.6).

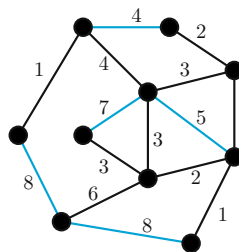


Abb. 13.6: Die blau eingezeichneten Kanten werden von dem Algorithmus nach und nach gelöscht.

3. Der Algorithmus berechnet eine optimale Lösung. Beweisidee: Zu dem Zeitpunkt, zu dem entschieden wird, dass die Kante e im Graphen bleibt und nicht gelöscht wird, ist sie die billigste in dem von ihr erzeugten Fundamentalschnitt. Damit erfüllt der gesamte übrig bleibende zusammenhängende Graph diese Eigenschaft und ist ein minimal spannender Baum.

Zu 3.3: Beide Bäume sind nicht minimal spannend.

Zu 3.4: Ein Bauunternehmer interessiert sich eher für große Aufträge und sucht dementsprechend nach einem maximal spannenden Baum. Multipliziert man die Kantengewichte mit minus eins, so entspricht ein minimal spannender Baum bezüglich dieser negativen Kantengewichte einem maximal spannenden Baum bezüglich der ursprünglichen Kantenbewertung. Für die Algorithmen von Prim und Kruskal stellen negative Gewichte keine Schwierigkeit da.

Zu 3.5: 1. Sortieren Sie die Kanten ihrer Gewichte nach und innerhalb derselben Gewichtsklasse, sodass die Baumkanten als Erstes kommen, dann die anderen

Kanten. Per Widerspruch lässt sich zeigen, dass damit genau der gesuchte Baum berechnet wird. 2. Ähnliches kann man auch für den Algorithmus von Prim zeigen. Wenn man ebenfalls von einer Sortierung in der Gewichtsklasse ausgeht, so stehen ebenfalls alle Baumkanten an erste Stelle.

Zu 3.6: 1. Die Aussage stimmt. Beweis: Angenommen, es gibt zwei minimal spannende Bäume T_1 und T_2 . Sei e eine Baumkante von T_1 , die nicht in T_2 enthalten ist. Dann gibt es eine weitere Kante f , die sowohl im Fundamentalschnitt von e bzgl. T_1 als auch auf dem Fundamentalkreis von e bzgl. T_2 liegt. Nach dem Kreis- bzw. Schnittkriterium gilt dann $c(e) = c(f)$, ein Widerspruch zur Voraussetzung. 2. Für den K_4 , in dem $n - 1$ Kanten, die einen Baum bilden, ein Gewicht von 1 und alle anderen Kanten ein Gewicht von 2 haben, stimmt die Aussage nicht.

Zu 3.7: In der Abbildung 13.7 links ist eine Verteilung der Gewichte angegeben, sodass der minimal spannende Baum eindeutig ist. Auf der rechten Seite kann sowohl die Kante (c, b) als auch die Kante (a, b) in dem minimal spannenden Baum enthalten sein.

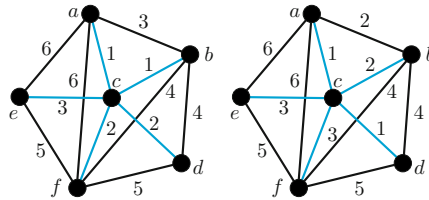


Abb. 13.7: Zwei Gewichtsverteilungen.

Zu 3.8: Der eindeutig bestimmte minimal spannende Baum T in dem gegebenen Graphen enthält alle zum Knoten 1 inzidenten Kanten und hat ein Gewicht von

$$c(T) = \sum_{i=1}^n i + 1 = \frac{(n+3) \cdot n}{2}.$$

Die Optimalität und Eindeutigkeit ergibt sich aus dem Kreiskriterium, da für jede Nicht-Baumkante (i, j) gilt

$$c((i, j)) = i + j > c((i, 1)).$$

Zu 3.9: Betrachten Sie die folgende Konstruktion der Folge und starten Sie mit $T_0 = T$: Sei T_i der i -te Baum der Folge und $e_i \in E(T') \setminus E(T_i)$. Da e_i eine Nicht-Baumkante von T_i ist und T' ein spannender Baum, gibt es eine Kante \bar{e}_i , die auf dem Fundamentalkreis von e_i liegt, aber nicht in T' enthalten ist. Konstruieren Sie $T_{i+1} = \{T_i + e_i\} - \bar{e}_i$. Der Baum T_{i+1} ist spannend und ist 1-benachbart zu T_i . Da in jedem Schritt eine Kante ausgetauscht wird, endet die Folge nach $n - 1 - k$ Schritten.

Zu 3.10: Die Aussage stimmt. Angenommen, es gibt einen minimal spannenden Baum T mit einem Knoten v , für den die Aussage nicht gilt. Dann ist die Nicht-Baumkante e_v günstiger als jede zu v inzidente Kante. Dies gilt insbesondere für

die zu v inzidenten Kanten des Fundamentalkreises zu e . Dies ist ein Widerspruch zur Optimalität des minimal spannenden Baumes T .

Zu 3.11: Sei T' ein minimal spannender Baum bzgl. c_1 . Dann erfüllt T' das Kreiskriterium für jede Nicht-Baumkante e' , d.h.

$$c_1(e') \geq c_1(e'') \quad e'' \in C_{e'}.$$

Damit ist aber auch das Kreiskriterium bzgl. der Kantengewichtsfunktion c_2 für T' erfüllt. Der Baum T' ist somit auch ein minimal spannender Baum bzgl. c_1 . Vertauscht man die Rollen von c_1 und c_2 erhält man die Äquivalenz.

Lösungshinweise zu Kapitel 4

Zu 4.1: Die Abzweigungen und Kreuzungen des Museums können als Knoten modelliert werden, die Seiten der Gänge als Kanten. Der Grad an jedem Knoten ist gerade, da die Kanten immer nur paarweise auftreten (jeder Gang hat zwei Seiten). Damit ist der Graph eulersch. Eine Euler-Tour bestimmt nun einen Gang durch das Museum, ohne zwei Seiten doppelt ablaufen zu müssen. Die Beschreibung der Modellierung ist nicht auf den gegebenen Grundriss fixiert. Allgemein existiert für jeden Museumsgrundriss ein Rundgang, der jede Wand genau einmal entlang führt.

Zu 4.2: Es gibt keinen solchen Rundgang. Betrachtet man den dazugehörigen Graphen, so ist er nicht eulersch (Abb. 13.8).

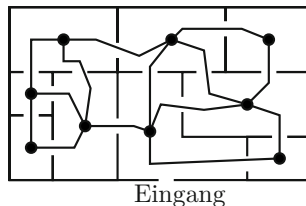


Abb. 13.8: Von dem Eckzimmer oben links gehen drei Türen ab.

Zu 4.3: Eine mögliche Lösung ist in Abbildung 13.9 zu sehen.

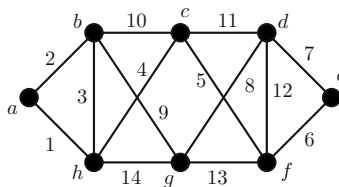


Abb. 13.9: Natürlich gibt es noch mehr Lösungen.

Zu 4.4: Der $K_{a,b}$ ist genau dann eulersch, wenn a und b gerade sind.

Zu 4.5: 1. Wir können mit Hilfe einer Euler-Tour das obige Problem lösen. Dazu modellieren wir einen Graphen G , der für jede Anzahl an Augen einen Knoten enthält. Zwischen je zwei Knoten verläuft eine Kante. Diese Kanten stellen einen Dominostein dar (Abb. 13.10). Eine Euler-Tour in dem vollständigen K_7 mit jeweils noch einer Schleife pro Knoten ergibt dann die gewünschte Kreiskette (und andersherum ergibt jeder Kette eine Euler-Tour im dazugehörigen Graphen). Eine Euler-Tour existiert, da jeder Knoten den Grad $d(v) = 6$ (bzw. $d(v) = 8$ mit Schleifen) hat.

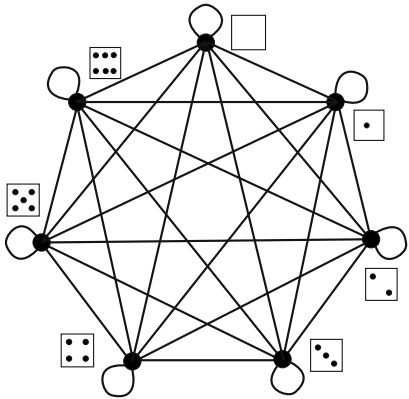


Abb. 13.10: Anstatt eine Kette aus Dominosteinen zu bestimmen, können wir in dem obigen Graphen eine Euler-Tour konstruieren.

- 2. Für den zweiten Fall existiert keine Kette, da auch keine Euler-Tour in dem Graphen K_4 existiert. Alle Knoten haben einen ungeraden Grad.
- 3. Wenn wir die oben genannten Steine entfernen, so entspricht dies in Bezug auf die Graphentheorie dem Entfernen einer Kante pro Knoten aus dem vollständigen Graphen K_{k+1} . Damit hat jede Kante wieder den Grad $k - 1$, also eine gerade Anzahl. Der Graph ist eulersch und von daher existiert auch eine Dominokette.

Zu 4.6: Mit 3 und 4 Knoten gibt es genau einen Graphen, den $C_3 = K_3$ und den C_4 . Bei fünf Knoten entsteht die folgende Situation: Da die Knotengrade gerade sein müssen, sind zu jedem Knoten entweder 2 oder 4 Kanten inzident. Für die folgenden Gradzahlentupel gibt es einen eindeutigen Graphen: $(2, 2, 2, 2, 2)$ (G_1), $(2, 2, 2, 2, 4)$ (G_2), $(2, 2, 2, 4, 4)$ (G_3) und $(4, 4, 4, 4, 4)$ (G_4) (Abb. 13.11). Für die Folgen $(2, 2, 2, 4, 4)$ gibt es keinen Graphen.

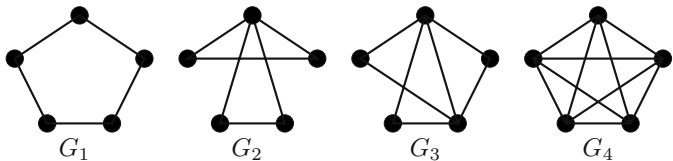


Abb. 13.11: Für fünf Knoten gibt es vier unterschiedliche Graphen.

Zu 4.7: Nein, diese Aussage stimmt nicht. Die Kanten g und f haben den Knoten b im Graphen G der Abbildung 13.12 gemeinsam. Angenommen, die Euler-Tour durchläuft zunächst die Kante g .

Fall 1: Sie hat den Kreis bed noch nicht durchlaufen. Dann muss dies erst geschehen, bevor die Kante f benutzt wird. Damit folgt f nicht direkt auf g .

Fall 2: Der Kreis bed wurde schon durchlaufen. Da g vor f durchlaufen wird, musste die Euler-Tour in b gestartet sein, dann den Kreis durchlaufen haben. Dann müssen aber zunächst alle weiteren Kanten benutzt werden, bevor die Kante f durchlaufen wird. Ansonsten könnte die Euler-Tour nicht mehr geschlossen werden.

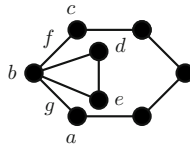


Abb. 13.12: Die Kanten f und g treten nie hintereinander in einer Euler-Tour auf.

Zu 4.8: 1. In einem bipartiten Graphen $G = (A \cup B, E)$ gilt

$$\sum_{v \in A} d(v) = m.$$

Da G eulersch ist, sind alle Grade gerade und somit ist auch die Summe bzw. m gerade. 2. Der Graph in Abbildung 13.13 ist einfach, eulersch, hat 6 Knoten und 9 Kanten. Also gilt die Aussage nicht.

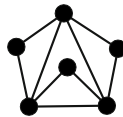


Abb. 13.13: Für G gilt $n = 6$ und $m = 9$, obwohl er eulersch ist.

Lösungshinweise zu Kapitel 5

Zu 5.1: Wir konstruieren einen Graphen G wie folgt: Die Knotenmenge repräsentiert die sechs Diplomaten, und zwischen zwei solcher Knoten existiert eine Kante genau dann, wenn es keinen Konflikt zwischen den Ländern gibt, die die beiden entsprechenden Diplomaten vertreten. Eine Sitzordnung entspricht einem Hamilton-Kreis in G (Abb. 13.14). Es ist leicht zu sehen, dass G hamiltonsch ist. Die folgende Abbildung zeigt eine von drei möglichen Sitzordnungen.

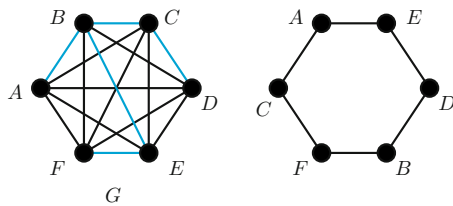


Abb. 13.14: Die blauen Kanten stellen die Konflikte dar. Rechts sieht man eine Möglichkeit, wie die Diplomaten in Ruhe diskutieren können.

Zu 5.2: Die Graphen zu a, d, f und g sind hamiltonsch.

Zu 5.3: Färbt man die Würfel abwechselnd schwarz und weiß, sodass Käsewürfel der gleichen Farbe keine gemeinsame Fläche haben, dann beginnt die Maus mit einem schwarzen Würfel und soll in einem weißen enden. Insgesamt gibt es 27 Würfel, d.h., wenn sich die Maus durch alle Würfel frisst, muss sie eine gerade Anzahl von Farbwechseln hinter sich bringen. Nach einer geraden Anzahl von Farbwechseln ist sie in einem schwarzen Käsewürfel, sie kann also niemals in der Mitte enden.

Wenn sie in einem weißen Würfel beginnt, würde eine Tour durch alle Würfel wegen der geraden Anzahl von Farbwechseln auch in der gleichen Farbe enden. Ein Weg, der weiße und schwarze Würfel abwechselt und in einem weißen Würfel beginnt und endet, enthält einen weißen Würfel mehr als schwarze Würfel. Es gibt in unserem Fall aber mehr schwarze als weiße Käsewürfel. Also kann die Maus auch in diesem Fall keinen geeigneten Fressweg durch alle Würfel finden.

Zu 5.4: Jeder Kreis C_n mit $n \geq 3$ ist sowohl ein Hamilton-Kreis als auch eine Euler-Tour. Weitere Graphen gibt es nicht.

Zu 5.5: Der $K_{a,b}$ ist genau dann hamiltonsch, wenn $a = b$ mit $a \geq 2$ gilt.

Zu 5.6: 4×4 und 5×5 geht nicht, aber 6×6 . Zu 4×4 : Löscht man die beiden Knoten aus den mittleren weißen Flächen, so zerfällt der Graph in drei Zusammenhangskomponenten (Abb. 13.15). Also kann er nicht hamiltonsch sein. Bei 5×5 gibt es mehr schwarze als weiße Felder, und da der dazugehörige Graph bipartit ist, kann es damit keinen Hamilton Kreis geben.

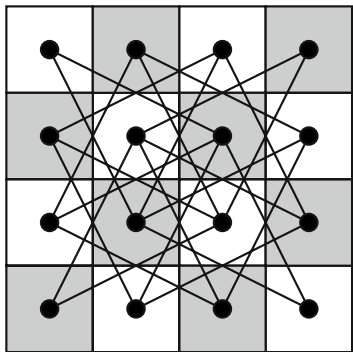


Abb. 13.15: Der Graph stellt die Bewegungsmöglichkeiten des Springers dar.

Zu 5.7: Beide Graphen aus Abbildung 13.16 haben die Gradfolge $(3, 3, 3, 3, 3, 3, 6)$. Allerdings ist nur der Graph G_2 hamiltonsch.

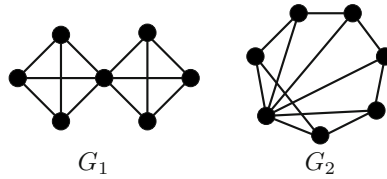


Abb. 13.16: Der Graph G_1 enthält keinen Hamilton-Kreis.

Zu 5.8: Da für einen solchen Graphen $d(v) \geq \frac{n}{2}$ gilt, folgt mit dem Satz von Dirac, dass er einen Hamilton-Kreis enthält.

Zu 5.9: Die Aussage folgt aus dem Satz von Dirac. Da jeder Knoten in \overline{C}_n den Knotengrad $d(v) = n - 3$ hat, gilt $d(v) \geq 0,5 \cdot n$ für $n \geq 5$.

Lösungshinweise zu Kapitel 6

Zu 6.1: 1. $f = 1$, d.h. der Graph ist ein Baum; 2. $m = 11$, 3. $n = 10$; Kreis mit 10 Knoten und einer Kante, die den Kreis in der Mitte teilt.

Zu 6.2: Es reicht irgendeine Kante zu löschen. Ansonsten wäre dies ein Widerspruch zum Satz von Kuratowski.

Zu 6.3: Mit drei Knoten nicht, aber der K_4 und der $K_{2,3}$ sind planare, aber nicht outerplanare Graphen.

Zu 6.4: Der $K_{2,n}$ ist planar, da 13.17 eine seiner planaren Darstellungen ist. Der $K_{n,m}$ ist nur für $n \in \{1, 2\}$ und $m \in \mathbb{N}$ (bzw. $m \in \{1, 2\}$, $n \in \mathbb{N}$) planar. Alle anderen vollständigen bipartiten Graphen enthalten den $K_{3,3}$ als Minor.

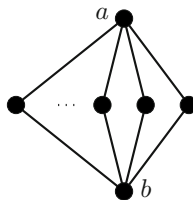


Abb. 13.17: Die zweielementige Menge besteht aus den Knoten a und b .

Zu 6.5: Nein. Auch Wälder (d.h. nicht zusammenhängende Bäume) haben nur eine Fläche.

Zu 6.6: Der Minor eines eulerschen bzw. hamiltonschen Graphen muss nicht wieder eulersch bzw. hamiltonsch sein. Beschränkt man sich auf die Kontraktion, so bleibt ein eulerscher Graph eulersch. Ein hamiltonscher Graph muss allerdings nicht hamiltonsch bleiben.

Zu 6.7: Ein solches Tupel existiert. Für $n = 5$, $m = 10$ und $f = 7$ kann es keinen planaren Graphen geben, da nur der K_5 die ersten beiden Parameter erfüllt, aber nicht planar ist.

Zu 6.8: Angenommen, es gäbe einen solchen Graphen G . Dann gilt für G nach dem *Handshaking-Lemma*

$$2 \cdot m = \sum_{v \in V} d(v) \geq 6 \cdot (n - 1).$$

Auf der anderen Seite wissen wir, dass

$$m \leq 3n - 6$$

gilt. Damit ergibt sich

$$3n - 3 \leq m \leq 3n - 6,$$

ein Widerspruch.

Zu 6.9: Sei n die Anzahl der Knoten, m die Anzahl der Kanten, f_4 die Anzahl der Flächen des Graphen mit der Länge 4 und f_6 die Anzahl der Flächen des Graphen mit der Länge 6. Da G ja 3-regulär ist, gilt $m = \frac{3}{2}n$. Jeder Knoten liegt genau zweimal an einer Fläche der Länge 4, d.h. $4f_4 = 2n$. Außerdem liegt jeder Knoten auf genau einer Fläche der Länge 6, d.h. $6f_6 = n$. Weiter erhalten wir aus der Euler-Formel

$$n - m + f_4 + f_6 = 2.$$

Löst man nun die vier Gleichungen, so ergibt sich für die vier Variablen $n = 12$, $m = 18$, $f_4 = 6$ und $f_6 = 2$ (Abb. 13.18).

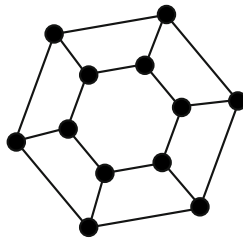


Abb. 13.18: Der Graph ist 3-regulär und erfüllt auch die restlichen Bedingungen.

Zu 6.10: Wir hatten gezeigt, dass jeder planare Graph einen Knoten vom Grad kleiner als sechs hat. Da in einem k -regulären Graphen jeder Knoten denselben Grad hat, muss $k \leq 5$ gelten.

Zu 6.11: Angenommen, dies ist nicht der Fall. Dann gilt

$$3 \cdot (f - 1) + 4 \leq \sum_{F \in G} d(F) \leq 2 \cdot m$$

Nach der Euler-Formel gilt $f = 2 - n + m$ und somit

$$2 \cdot m \geq 3 \cdot (f - 1) + 4 = 10 - 3n + 3m,$$

d.h. $m \leq 3n - 10$. Das ist ein Widerspruch zur Voraussetzung.

Zu 6.12: Es gilt

$$f \cdot 5 \leq 53 \cdot 5 \leq \sum_{F \in G} d(v) = 2m$$

und durch eine Umformung der Euler-Formel

$$n \geq m + 2 - f \geq \frac{53 \cdot 5}{2} + 2 - 53 = 81.5.$$

Zu 6.13: Für einen planaren Graphen gilt

$$m \leq 3n - 6.$$

Da ein vollständiger Graph $0,5 \cdot n \cdot (n - 1)$ Kanten enthält, müssen mindestens $0,5 \cdot n^2 - 3,5n + 6$ Kanten gelöscht werden, damit der neue Graph planar sein kann.

Lösungshinweise zu Kapitel 7

Zu 7.1: Die Kinder können als Knoten und ihre Streitigkeiten als Kanten modelliert werden. Die Anzahl der benötigten Zimmer entspricht nun der chromatischen Zahl des Graphen. In diesem Fall sind es drei.

Zu 7.2:

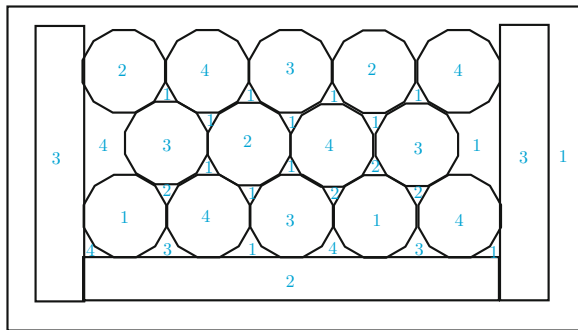


Abb. 13.19: Eine Färbung der Landkarte.

Zu 7.3:

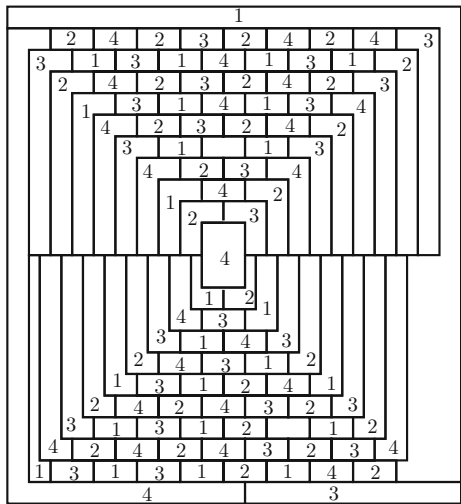


Abb. 13.20: Eine 4-Färbung der Karte.

Zu 7.4: Für (a): Der Graph ist bipartit, d.h. für a, b, c, d, e, f bekommt man eine Färbung von 2. Auf der anderen Seite ergibt a, d, b, e, f, c eine Färbung von $3 = \Delta(G_a) + 1$. Für (b): Optimal sind 3 Farben und mit der Knotenreihenfolge a, b, c, e, d, f, g erhält man 4 Farben. Graph (c): Mit der Knotenreihenfolge $g, f, b, \ell, j, k, i, h, a, c, d, e$ erhält man eine Färbung mit 6 Farben. Die chromatische Zahl ist 4. Für (d) (Petersen-Graph): Mit der Knotenreihenfolge a, b, f, h, g, e, j erhält man 4 Farben. Ansonsten benötigt man nur 3 Farben.

Zu 7.5: Die unterschiedlichen Verkehrsströme werden als Knoten und Konflikte als Kanten dargestellt (Abb. 13.21). In dem Graphen gibt es dann eine 3-Färbung, d.h. es werden drei Ampelphasen benötigt.

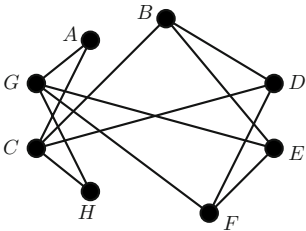


Abb. 13.21: Die Ampeln könnten folgendermaßen geschaltet werden: C, G , dann D, E und anschließend A, H, B, F .

Zu 7.6: Angenommen, es existiert eine solche Landkarte. Dann enthält der Dualgraph der Landkarte den K_5 als Minor. Der K_5 ist jedoch nicht planar, ein Widerspruch.

Zu 7.7: Ein Graph G ist 2-färbbar. $\Leftrightarrow G$ ist bipartit. $\Leftrightarrow G$ enthält nur Kreise gerader Länge. $\Leftrightarrow G$ enthält keine Kreise ungerader Länge.

Zu 7.8: Jedes Zahlenfeld wird zu einem Knoten. Innerhalb eines 3×3 -Blocks sind alle Knoten über eine Kante miteinander verbunden. Innerhalb einer Reihe/-Spalte auch. Dann ist jede Lösung des Sudokus eine 9-Färbung des Graphen und jede 9-Färbung eine Lösung des Sudokus (wobei bei Sudoku schon einige Farben vorgegeben werden).

Zu 7.9: Beweis über Induktion nach der Anzahl der Kreise k . Für $k = 1$ besteht die Zeichnung aus einem Kreis und kann somit entweder in blau oder weiß gefärbt werden. Induktionsschluss von $k - 1$ auf k : Wir wählen einen der k Kreise aus der Zeichnung aus und löschen ihn. Die restliche Zeichnung kann nach der Induktionsvoraussetzung mit blau und weiß gefärbt werden. Fügen wir nun den ausgewählten Kreis wieder in die Zeichnung ein, erhalten wir eine zulässige Färbung, wenn wir außerhalb des Kreises Färbung unverändert lassen, innerhalb davon vertauschen.

Zu 7.10: In jedem outerplanaren Graphen gibt es einen Knoten mit zwei Nachbarn. Wählen Sie diesen Knoten, färben Sie ihn (anders) als seine Nachbarn und löschen Sie ihn aus dem Graphen. Der neue Graph ist wieder outerplanar. Also kann die Prozedur wiederholt werden. Damit erhält man insgesamt eine Dreifärbung.

Zu 7.11: 1. Sei $e = (u, v)$ die gelöschte Kante. Dann können die Knoten u und v dieselbe Farbe bekommen. Es gilt also $\chi(K_n - e) = n - 1$. 2. Diese Graphen können mit $n - 1$ Farben gefärbt werden. 3. Hier reichen $n - 2$ Farben.

Lösungshinweise zu Kapitel 8

Zu 8.1: Von a aus kann jeder Knoten erreicht werden (Abb. 13.22), von g aus beispielsweise nicht.

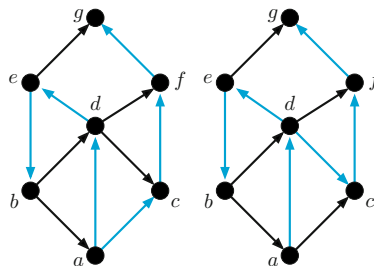


Abb. 13.22: Links sieht man einen Breitensuchbaum in G und rechts einen Tiefensuchbaum.

Zu 8.2: Der linke Graph ist stark zusammenhängend, da ein Kreis existiert $ifhebacdgi$, der alle Knoten enthält. Der rechte Graph hingegen ist nicht stark zusammenhängend. Der Schnitt bzgl. der Menge $X = \{f, g, h, i, \}$ ist gerichtet.

Zu 8.5: Der Weg $adfebcb$ ist ein Hamilton-Weg.

Lösungshinweise zu Kapitel 9

Zu 9.1: Modellierung: Jedes Blatt stellt einen Knoten dar und von jedem Knoten gehen genau zwei gerichtete Kanten aus, jeweils zu den Knoten, die von dem Blatt aus erreicht werden können. In diesem speziellen Fall reichen 6 Sprünge aus, indem Daniel zuerst 4 nach vorne, 2 nach vorne, 2 nach vorne, 1 zurück, 3 nach vorne, 2 nach vorne springt.

Zu 9.2: Teil 1: Der kürzeste Weg bzgl. der ersten Kostenfunktion mit 0,07 Euro pro Kilometer und der für 0,05 Euro pro Kilometer ist derselbe und in Abb. 13.23 blau eingezeichnet. Die gestrichelte Linie zeigt den kürzesten Weg, wenn die Regierung zusätzliche Maut-Gebühren erhebt.

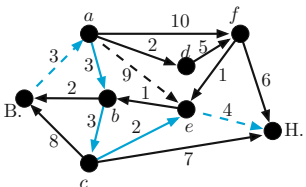


Abb. 13.23: Der kürzeste Weg ist unabhängig von den Benzinkosten pro Kilometer.

Teil 2: Für jeden (s, t) -Weg p' gilt

$$c(p') = \sum_{e \in p'} c(e) = \frac{1}{\alpha} \cdot \sum_{e \in p'} c_1(e) = \frac{1}{\alpha} c_1(p).$$

Seien p_1 und p_2 nun zwei Wege. Dann ergibt sich $c(p_1) \leq c(p_2) \Leftrightarrow c_1(p_1) \leq c_1(p_2)$. Somit ist p nicht nur ein kürzester Weg bzgl. c , sondern auch bzgl. c_1 . Für die Kostenfunktion c_2 gilt eine solche Äquivalenz nicht, da die Anzahl der Kanten in die neue Bewertung des Wegs mit einbezogen wird.

Zu 9.3:

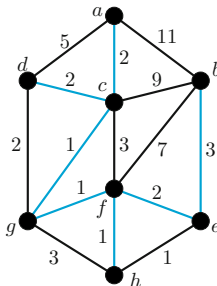


Abb. 13.24: Der über den Dijkstra berechnete Kürzeste-Wege-Baum von a aus.

Zu 9.4: Teil 1: Anne muss mindestens 17 Minuten auf dem Fahrrad verbringen. Die optimale Route ist in Abb. 13.25 eingetragen.

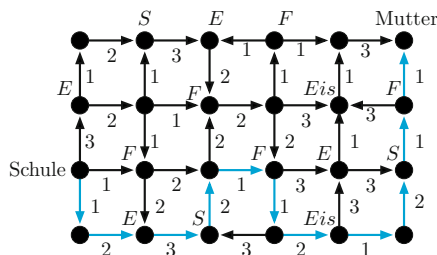


Abb. 13.25: Eine optimale Route für Anne.

Teil 2: Das Problem kann über k Kopien G_1, \dots, G_k des Graphen G modelliert werden. Sei $(u, v) \in G$ eine Kante und $u \in V_i$. Dann verläuft eine Kante zwischen der Kopie von u im Graphen G_i und einer Kopie von v in G_{i+1} mit demselben Kantengewicht wie die ursprüngliche Kante. Ein kürzester (s, t) -Weg in diesem erweiterten Graphen entspricht einer optimalen Lösung des Problems.

Zu 9.5: In einem Kettengraphen mit n Knoten gibt es 2^{n-1} unterschiedliche Wege von v_1 nach v_n .

Zu 9.6: Für $n \geq 2$ enthält der Graph 2^{n-2} einfache (v_1, v_n) -Wege.

Zu 9.7: Die Aussage stimmt nicht, wie das folgende Beispiel zeigt (Abb. 13.26).

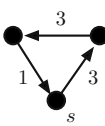


Abb. 13.26: Die günstigste Kante mit Kosten 1 ist nicht in dem Kürzesten-Wege-Baum von s aus enthalten.

Zu 9.8: 1. Seien e_1, \dots, e_i die günstigsten zu s inzidenten Kanten in G , d.h. $c(e_1) = c(e_2) = \dots = c(e_i) < c(e)$ für alle $e \in \delta(s)$ und $e \neq e_j$ für alle $j = 1, \dots, i$. Dann enthält T_{SP} alle Kanten e_1, \dots, e_i . Enthält T_{MST} keine dieser Kanten, so ist das Kreiskriterium bzgl. jeder dieser Kanten verletzt. Damit haben T_{SP} und T_{MST} mindestens eine gemeinsame Kante. 2. Für $c \equiv 0$ gilt die Aussage nicht, wie das Beispiel in Abb. 13.27 zeigt.

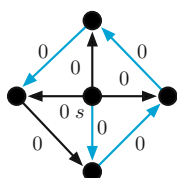


Abb. 13.27: Der minimal spannende Baum bestehend aus den blauen Kanten und der Kürzeste-Wege-Baum von s bestehend aus den schwarzen Kanten haben keine gemeinsame Kante.

Zu 9.9: Wir definieren die Kostenfunktion c mit

$$c(e) = -\log(w(e)) \quad \forall e \in E.$$

Da $0 < w(e) < 1$ gilt, erhalten wir $c(e) \geq 0$. Ein kürzester (s, t) -Weg bzgl. c ist dann ein Weg mit maximaler Zuverlässigkeit w . Berechnet werden kann dieser mit dem Dijkstra-Algorithmus.

Zu 9.10: 1. Es gilt

$$\sum_{i=1}^m \frac{1}{2} < 1$$

und für zwei disjunkte Teilmengen $I_1 \subseteq \{1, \dots, m\}$ und $I_2 \subseteq \{1, \dots, m\}$

$$\sum_{i \in I_1} \frac{1}{2} \neq \sum_{i \in I_2} \frac{1}{2}.$$

Daher ergibt sich für zwei (s, t) -Wege p_1 und p_2 immer

$$c'(p_1) \neq c'(p_2).$$

2. Sei p' ein kürzester Weg bzgl. c' . Angenommen, p' sei kein kürzester Weg bzgl. c , d.h. es existiert ein Weg p mit $c(p) + 1 \leq c(p')$. Dann gilt

$$c'(p) \leq \sum_{e \in p} c(e) + \sum_{i=1}^m \frac{1}{2} < \sum_{e \in p} c(e) + 1 \leq c(p') \leq c'(p'),$$

ein Widerspruch zu der Voraussetzung.

3. Zwei Wege p_1 und p_2 , die bzgl. c dieselbe Länge haben, dürfen bzgl. c' nicht dieselbe Länge haben (siehe 1.).

Lösungshinweise zu Kapitel 10

Zu 10.1: Der Fluss ist nicht maximal. Ein maximaler Fluss ist in Abb. 13.28 gegeben.

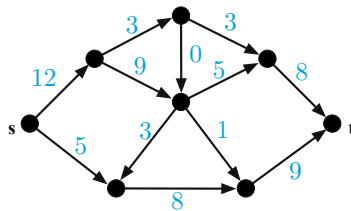


Abb. 13.28: In dieser Darstellung ist nur der Fluss und nicht die Kapazität pro Kante gegeben.

Zu 10.2: Teil 1: Der maximale Fluss in G zwischen HC und HT gibt die Anzahl der transportierbaren Container an. In diesem Fall sind es 10. **Teil 2:** Betrachten Sie die Modellierung von Teil 3. Maximal können jetzt noch 8 Container verschickt werden. **Teil 3:** Für jeden Knoten v wird eine extra Kante (v^+, v^-) zum Graphen

G hinzugefügt, wobei alle eingehenden Kanten in v mit dem Knoten v^+ und alle ausgehenden mit dem Knoten v^- verbunden werden. Diese Kante bekommt dann die Kapazität $d(v)$.

Zu 10.3: Verbindet man die blauen Knoten mit einer Quelle s und alle Randknoten mit einer Senke t , so gibt es genau dann genügend Fluchtwege, wenn ein maximaler Fluss mit einem Flusswert von 8 in dem Netzwerk mit Kapazitäten 1 existiert.

Zu 10.4: Teil 1: In dem dargestellten Graphen gibt es vier kantendisjunkte Wege. Ein Paket sollte somit sinnvollerweise geviertelt werden. **Teil 2:** Fügen Sie obere Kapazitäten $u(e) = 1$ für jede Kante e in das Netzwerk ein und berechnen Sie einen maximalen Fluss. Der Flusswert entspricht der maximalen Anzahl kantendisjunkter Wege in G . Die Wege erhält man durch die im Dekompositionssatz von Ford-Fulkerson vorgestellte Methode.

Zu 10.5: Der Fluss lässt sich in fünf (s, t) -Wege zerlegen: $p_1 = scet$ mit $x(p_1) = 3$, $p_2 = scbet$ mit $x(p_2) = 2$, $p_3 = sbet$ mit $x(p_3) = 5$, $p_4 = sbdet$ mit $x(p_4) = 5$ und $p_5 = sadet$ mit $x(p_5) = 5$. Zusätzlich enthält die Zerlegung noch den Kreis $p_6 = ded$ mit $x(p_6) = 3$.

Zu 10.6: Teil 1: Eine Modellierung funktioniert wie im Teil 2. Der maximale Fluss hat einen Wert von 11. **Teil 2:** Das Problem kann in ein maximales Flussproblem mit einer Quelle und einer Senke umgewandelt werden. Hierzu wird eine Super-Quelle, die mit allen Quellen über eine Kante verbunden ist, dem Netzwerk hinzugefügt. Jede Kante erhält als obere Kapazität die Sendekapazität der Quelle. Genauso verbindet man alle Senken mit einer Super-Senke. Ein maximaler Fluss entspricht dann dem maximalen Transportvolumen.

Zu 10.7: Teil 1: Insgesamt kommen 30 Fans pro Minute in der Station an. **Teil 2:** Sei e eine ungerichtete Kante mit der oberen Kapazität $u(e)$. In dem neuen Netzwerk ersetzen wir die Kante e durch zwei antiparallele Kanten e_1 und e_2 mit $u(e_1) = u(e_2) = u(e)$. Der berechnete Fluss f in dem neuen Netzwerk ergibt einen maximalen Fluss f' in dem ursprünglichen durch

$$f'(e) = \begin{cases} |f(e_1) - f(e_2)| & e \text{ ungerichtet} \\ f(e) & \text{sonst.} \end{cases}$$

Zu 10.8: Nicht jeder maximale Fluss muss ganzzahlig sein, auch wenn die oberen Kapazitäten dies sind. Dies zeigt das folgende Beispiel (Abb. 13.29).

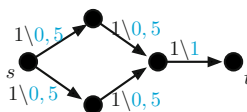


Abb. 13.29: Jeder maximale Fluss hat einen Flusswert von 1, so auch der blau eingezeichnete nicht ganzzahlige (s, t) -Fluss.

Zu 10.9: Sei die Kante (u, v) eine ausgehende Kante aus der Menge $S \cup T$ und u o.B.d.A. aus S . Dann ist die Kante $(u, v) \in \delta^+(S)$. Da $\delta(S)$ ein minimaler (s, t) -

Schnitt ist, gilt $f((u, v)) = u((u, v))$. Betrachten wir nun eine eingehende Kante (x, y) in die Menge $S \cup T$ und sei y o.B.d.A. aus S . Dann gilt $(x, y) \in \delta^-(S)$ und somit $f((x, y)) = 0$. Daraus ergibt sich

$$\begin{aligned} \text{value}(f) &= \sum_{e \in \delta^+(S \cup T)} f(e) - \sum_{e \in \delta^-(S \cup T)} f(e) \\ &= \sum_{e \in \delta^+(S \cup T)} u(e) - \sum_{e \in \delta^-(S \cup T)} 0 \\ &= \text{cap}(S \cup T). \end{aligned}$$

Damit ist $S \cup T$ ein minimaler Schnitt. Eine ähnliche Argumentation gilt für $S \cap T$.

Lösungshinweise zu Kapitel 11

Zu 11.1: Der gegebene b -Fluss erzeugt keine Kosten. Mit dem kostenminimalen Fluss ergibt sich sogar ein Gewinn von 12 (oder Kosten von -12).

Zu 11.2: Behalten Sie den Graphen $G = (V, E)$ und die Kostenfunktion bei. Erweitern Sie den Graphen um obere Kapazitäten $u(e) = 1$ für alle Kanten $e \in E$ und Balancen $b(v)$ mit

$$b(v) = \begin{cases} 1 & \text{für } v = s \\ -1 & \text{für } v = t \\ 0 & \text{sonst.} \end{cases}$$

Der berechnete b -Fluss setzt sich aus Zykeln der Länge 0 und einem kürzesten (s, t) -Weg zusammen.

Zu 11.3: Die Transportkosten sinken. Das Problem lässt sich als kostenminimales Fluss-Problem modellieren, in dem ein Angebotsüberschuss vorhanden ist. Dieser Überschuss wird durch das Hinzufügen einer Super-Senke mit der entsprechenden Nachfrage modelliert. Die Super-Senke ist mit allen Quellen über eine Kante verbunden. Die Kapazität dieser Kante ist unbeschränkt und die Kosten betragen Null.

Ein kostenminimaler Fluss in diesem Netzwerk hat dieselben Kosten wie ein Transportplan, der nur die notwendigen Einheiten verschickt. Außerdem kann er sofort in einen solchen Transportplan umgewandelt werden, indem der Fluss über die extra Kanten gelöscht wird.

Wir erhöhen nun das Angebot an einem Quellknoten. Dann ist der ehemals kostenminimale Fluss weiterhin ein zulässiger Fluss, erweitert man ihn um das Verschicken der Einheiten direkt zur Super-Senke. Damit kann ein kostenminimaler Fluss zu den neuen Balancen nicht mehr kosten als dieser.

Zu 11.4: Der b -Fluss ist kostenminimal, da kein negativer Zykel im Residualgraphen G^f existiert. Die Kosten betragen $c(f) = 152$. Ein kostenminimaler Fluss zu den erhöhten Balancen hat Kosten von 150.

Zu 11.5: Die Multiplikation von k ändert nichts an der Optimalität eines Flusses. Die Addition des Wertes k schon.

Zu 11.6: Je nachdem, ob das Angebot oder die Nachfrage größer ist, wird eine Super-Quelle oder eine Super-Senke eingeführt. Die Super-Quelle wird dabei mit allen Senken direkt verbunden, die Kapazität der Kanten ist unbeschränkt, aber die Kosten sind sehr groß. Die Super-Quelle s^* erhält dann den Balance-Wert $b(s^*) = -\sum_{v \in V} b(v)$. Eine Super-Senke wird ähnlich definiert.

Zu 11.7: Im Netzwerk G_1 enthält kein kostenminimaler Fluss die Kante e_1 und in G_2 jeder die Kante e_2 (Abb. 13.30).

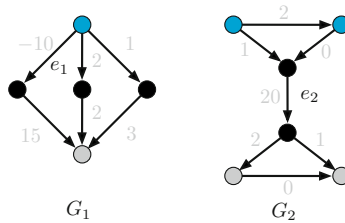


Abb. 13.30: Die oberen Kapazitäten seien in beiden Graphen unbeschränkt, die Kantenkosten sind grau eingezeichnet. Die blauen Knoten geben die Quellen und die grauen die Senken an. In G_1 wird kein Fluss über e_1 geschickt, in G_2 fließt hingegen immer Fluss über die Kante e_2 .

Zu 11.8: Falls $f(e) \leq u(e) - 1$ gilt, dann ist f weiterhin ein kostenminimaler b -Fluss. Ist dies nicht der Fall, so erhalten wir einen kostenminimalen Fluss f' , indem wir einen kürzesten Weg zwischen den Endknoten der Kante e im Residualgraphen G^f berechnen und entlang dieses Wegs den Fluss um eine Einheit verändern. Diese Vorgehensweise entspricht dem *Successive-Shortest-Path-Algorithmus*.

Zu 11.9: Sei f ein kostenminimaler Fluss. Angenommen, es existiert ein Zykel Z mit unbeschränkten Kapazitäten und negativen Kosten. Dann wäre

$$f'(e) = \begin{cases} f(e) & \text{falls } e \notin Z \\ f(e) + 1 & \text{falls } e \in Z \end{cases}$$

ebenfalls ein zulässiger b -Fluss (unabhängig von der Wahl von b). Es gilt aber

$$c(f') - c(f) = c(f) + c(Z) - c(f) = c(Z) < 0,$$

ein Widerspruch zur Kostenminimalität von f .

Sei Z ein Zykel mit unbeschränkten Kapazitäten und negativen Kosten. Angenommen, f sei ein kostenminimaler Fluss. Über dieselbe Argumentation wie eben erhalten wir auch hier einen Widerspruch.

Zu 11.10: Angenommen, ein zulässiger b -Fluss existiert in diesem Netzwerk. Erweitern Sie den Graphen um eine Super-Quelle und um eine Super-Senke. Die Super-Quelle s^* ist mit allen Quellen direkt verbunden. Jede Kante (s^*, v) erhält

eine obere Kapazität von $b(v)$. Die Super-Senke t^* wird mit allen Senken verbunden und jede Kante (v, t^*) erhält eine obere Kapazität von $-b(v)$. Dann ist

$$f'(e) = \begin{cases} f(e) & \forall e \in E \\ b(v) & \forall e = (s^*, v) \\ -b(v) & \forall e = (v, t^*) \end{cases}$$

ein zulässiger (s^*, t^*) -Fluss. Damit gilt

$$\sum_{\substack{v \in V \\ b(v) > 0}} b(v) = \sum_{e \in \delta^+(s^*)} f(e) = \text{value}(f) = \sum_{e \in \delta^-(t^*)} f(e) = \sum_{\substack{v \in V \\ b(v) < 0}} b(v),$$

ein Widerspruch zur Voraussetzung.

Lösungshinweise zu Kapitel 12

Zu 12.1: Das Problem entspricht der Suche nach einem perfekten Matching in einem bipartiten Graphen. In der Abbildung 13.31 ist eine mögliche Zuordnung angegeben.

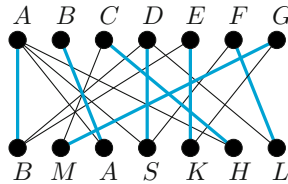


Abb. 13.31: Die oberen Knoten entsprechen den Projekten, die unteren den Personen.

Zu 12.2: Nein, da ein M -augmentierender Weg existiert.

Zu 12.3: Für die Schachbretter oben und unten links gibt es jeweils eine Belegung. Da jeder Dominostein ein weißes und ein schwarzes Feld abdeckt und mehr schwarze als weiße Felder vorhanden sind, kann keine Belegung für das Schachbrett oben rechts existieren. Für das letzte Schachbrett unten rechts gibt es keine Belegung, weil in der unteren rechten Ecke mehr schwarze als weiße Felder vorhanden sind.

Zu 12.4: Zur Berechnung eines maximalen Matchings in einem bipartiten Graphen $G = (A \cup B, E)$ wurde der Graph um eine Quelle s und eine Senke t erweitert, s mit allen Knoten aus A und t mit allen Knoten aus B verbunden und dann ein maximaler Fluss von s nach t bei oberen Kapazitäten von 1 berechnet. Erhöht man nun die Kapazitäten der Kanten von B nach t auf 10, so kann wiederum eine maximale Zuordnung berechnet werden, ohne alle Plätze explizit in B als Knoten zu modellieren.

Zu 12.5: 2. Sei C ein Hamilton-Kreis in G . Da G eine gerade Anzahl an Knoten hat, enthält der Kreis eine gerade Anzahl an Kanten. Löscht man jede zweite Kante, so bilden die restlichen Kanten ein perfektes Matching.

Zu 12.6: Beide Aussagen sind falsch. Für ungerade n enthält C_n kein perfektes Matching und ein Weg der Länge 4 ist ein Baum und enthält zwei unterschiedliche perfekte Matchings.

Zu 12.7: Ja, alle Graphen, die sich aus zwei Kreisen ungerader Längen zusammensetzen.

Zu 12.8: Sei $A' \subseteq A$ und seien $\Gamma(A')$ die Nachbarn von A' . Dann gilt

$$k \cdot |\Gamma(A')| \geq \sum_{b \in \Gamma(A')} d(b) \geq \sum_{a \in A'} d(a) \geq k \cdot |A'|.$$

Damit ist die Heiratsbedingung erfüllt und es existiert ein A überdeckendes Matching.

Zu 12.9: Modelliert man die Bewerber B und Jobs J als Knoten und die Qualifikationen Q als Kanten, so entsteht ein bipartiter Graph $G = (B \dot{\cup} J, Q)$. In diesem gilt $d(b) \geq 5 \geq d(j)$ für jeden Bewerber $b \in B$ und jeden Jobs $j \in J$. Nach der Aufgabe 12.8 existiert somit ein perfektes Matching in G , d.h. eine Jobzuteilung.

Zu 12.10: Wie in dem Beweis zum Satz von Berge gezeigt, besteht die symmetrische Differenz zweier Matchings aus Kreisen gerader Länge, Wegen und isolierten Knoten. Da $|M| > |N|$, existiert in $M \Delta N$ ein N -augmentierender Weg p . Sei N' das Matching, das durch das Augmentieren entlang des Wegs p aus dem Matching N entsteht, und M' das Matching, das durch das Hinzufügen der Kanten aus N in p und das Löschen der restlichen Kanten von p entsteht. Dann erfüllen M' und N' die gewünschten Bedingungen.

Zu 12.11: Die Strategie des zweiten Spielers ist die Folgende: Sei M ein perfektes Matching in G und v der von Spieler 1 als letztes gewählte Knoten. Wähle den zu v in M benachbarten Knoten w . Dieser Knoten kann von Spieler 2 noch nicht gewählt worden sein, da w nur zur Kante (v, w) in M inzident ist und auch Spieler 1 kann w noch nicht gewählt haben, da sonst Spieler 2 den Knoten v ausgesucht hätte. Spieler 2 gewinnt, da zu jedem von Spieler 1 gewählten Knoten eine solcher Partnerknoten existiert.

Anhang A Satz, Beweis, Definition

Übersicht

| | | |
|-----|--|-----|
| A.1 | Folgerungen und Äquivalenzen | 248 |
| A.2 | Negationen | 249 |
| A.3 | Beweis durch Widerspruch | 250 |
| A.4 | Vollständige Induktion | 251 |
| A.5 | Ringschluss | 253 |

Schon immer haben Mathematiker sich damit beschäftigt, aus bereits bekannten Aussagen neues Wissen mittels logischer Schlüsse zu ziehen. Dabei stehen am Anfang jeder Forschung gewisse Grundsätze, die als „wahr“ angenommen werden. Solche Grundsätze werden auch [Axiome](#) genannt. Eines der bekanntesten Axiome aus der Arithmetik ist das folgende: Jede natürliche Zahl n hat genau einen Nachfolger $n + 1$. Axiome bilden also das Fundament der Mathematik. Sind diese erst einmal festgelegt, so versuchen Mathematiker weiter Folgerungen aus ihnen zu ziehen. Die neuen Erkenntnisse werden in Theoremen, Sätzen, Lemmata oder Korollaren festgehalten. Eine klare Grenze, wann eine Erkenntnis ein Satz, ein Lemma oder ein Korollar ist, gibt es nicht. Die Übergänge sind fließend und Geschmackssache. Ganz grob ist jede größere Aussage ein [Satz](#). Als [Theorem](#) wird dann ein besonders wichtiger Satz bezeichnet. Ein [Lemma](#), oft auch als Hilfsatz übersetzt, wird dazu verwendet einen besonders langen Beweis in kleinere Teile zu zerlegen, oder eine wirklich grundlegende, aber dennoch einfache, Aussage zu treffen. Ein [Korollar](#) bezeichnet meist eine Folgerung aus dem vorangegangenen Satz oder Beweis.

Zur besseren Verständlichkeit vieler Aussagen, werden Definitionen verwendet. Eine [Definition](#) führt einen neuen Begriff oder ein neues Konzept in die Sprache der Mathematik ein. Dabei kann es kein „wahr“ oder „falsch“ geben. Es kommt nur darauf an, ob die Definition mit der bisherigen Sprechweise im Einklang ist oder nicht. Wir sprechen auch von [wohldefiniert](#).

Das Herzstück der Mathematik bilden die **Beweise**. Sie sind die logisch korrekten Herleitungen einer Aussage aus Axiomen oder vorherig bewiesenen Aussagen. Erst wenn ein Satz bewiesen ist, wird er von Mathematikern als wahr anerkannt. Diese Wahrheit kann anschließend auch nicht wegdiskutiert werden, sondern gilt für immer und ewig. Es gibt viele verschiedene Techniken einen Beweis zu führen. In den nächsten Abschnitten wollen wir uns mit den gängigsten beschäftigen.

A.1 Folgerungen und Äquivalenzen

Wir hatten gesagt, dass ein Beweis eine logisch korrekte Herleitung einer Aussage aus Axiomen oder vorherig bewiesenen Aussagen ist. Oder anders ausgedrückt ist ein Beweis der korrekte Nachweis, dass aus vorher bewiesenen Aussagen und festgelegten Axiomen eine weitere Aussage folgt. Stellt sich die Frage, was es heißt, dass eine Aussage aus irgendetwas Anderem *folgt*. Dazu haben wir eine gewisse Intuition, die jedoch bei komplexeren Aussagen nicht mehr ausreicht. Ein gewisses Maß an Formalismus ist deswegen hilfreich. Wenn wir zum Beispiel komplexe Aussagen aus mehreren Teilen verstehen wollen, so ist es eine gute Methode, jeden Teil mit einem Buchstaben (A, B, C, \dots) zu bezeichnen, um dann die logische Struktur des Satzes klarer vor sich zu sehen.

Hier ist ein Beispiel:

1. Poeten, die die Herzen der Menschen zu bewegen verstehen, sind clever.
2. Jeder meisterhafte Poet kann die Herzen der Menschen bewegen.
3. Shakespeare hat *Hamlet* geschrieben.
4. Nur ein meisterhafter Poet konnte *Hamlet* schreiben.

Wir bezeichnen jetzt jede Eigenschaft von Poeten, die in unserem Beispiel vorkommt, mit einem Buchstaben:

| | | | |
|-----|--------------------------------------|-----|-------------------------------|
| A | kann die Herzen der Menschen bewegen | D | ist Shakespeare |
| B | ist clever | E | hat <i>Hamlet</i> geschrieben |
| C | ist ein meisterhafter Poet | | |

Um die Aussagen der Sätze nun logisch darzustellen, verwenden wir das Symbol \Rightarrow . Dabei bezeichnen wir mit $A \Rightarrow B$, dass die Aussage B aus der Aussage A **folgt**. Damit lauten die Sätze von oben:

1. $A \Rightarrow B$, 2. $C \Rightarrow A$, 3. $D \Rightarrow E$ und 4. $E \Rightarrow C$.

Wie können wir aus diesen Aussagen den Satz „Shakespeare ist clever“ folgern? Wenn wir obige Aussagen nun geschickt anordnen, erhalten wir aus der Eigenschaft D (ist Shakespeare) die Eigenschaft B (ist clever), durch die folgende Argumentationskette $D \Rightarrow E \Rightarrow C \Rightarrow A \Rightarrow B$.

Manchmal implizieren sich sogar zwei Aussagen A, B . Dann gilt $A \Rightarrow B$ und $B \Rightarrow A$. Wir sagen dann, A und B sind **äquivalent**, in Zeichen $A \Leftrightarrow B$. Der deutsche Ausdruck dafür ist: A gilt **genau dann, wenn** B gilt.

In der Mathematik gibt es präzise Unterschiede zwischen den Ausdrücken *wenn* und *nur wenn* und *genau dann, wenn*. *Wenn A gilt, gilt B* heißt $A \Rightarrow B$. *Nur wenn A gilt, gilt B* bedeutet $B \Rightarrow A$. Und *A gilt genau dann, wenn B gilt*, heißt $A \Leftrightarrow B$.

Aufgabe A.1. Angenommen, es gelten die folgenden Gesetze:

1. Alle grünen Krokodile beißen und können fliegen.
2. Alle Moskauer Reptilien sind grün, beißen und können nicht fliegen.
3. Alle Frösche, die grün sind, sind Krokodile.

Folgern Sie:

1. Moskauer Reptilien sind keine Krokodile.
2. Grüne Frösche können fliegen.

Aufgabe A.2. Wie kann aus diesen Aussagen folgen, dass der Mensch ein böses Tier ist?

1. Nur böse Tiere fressen andere Tiere.
2. Elefanten haben keine Angst vor anderen Tieren, außer vor den Bezwingern der Natur.
3. Es ist unmöglich, die Natur zu bezwingen, ohne Tiere gefressen zu haben.
4. Elefanten haben Angst vor Menschen.

A.2 Negationen

Für alle Arten von Beweisen, vor allem aber den Widerspruchs-Beweis, ist es wichtig, Aussagen korrekt negieren (also verneinen) zu können. Und das ist nicht immer so einfach, wie man vielleicht meinen könnte. Was ist die Negation des Satzes „Köln und München sind Großstädte“? Der Satz „Köln oder München ist keine Großstadt“. Der wichtige Punkt ist hier, dass aus dem *und* ein *oder* wird. Andersherum wird bei der Negation aus einem *oder* ein *und*: Die Negation von „Peter oder Paul singt schief“ ist „Peter und Paul singen nicht schief“. In der Mathematik ist mit *oder* immer ein einschließendes oder gemeint, also ein *oder/und*. Andernfalls sagt man *entweder-oder*.

In ähnlicher Weise vertauschen sich bei der Negation eines Satzes auch All-Aussagen (für alle x gilt y) und Existenz-Aussagen (es gibt ein x mit y). Die Negation des Satzes „Über allen Gipfeln ist Ruh“ lautet beispielsweise „Es gibt einen Gipfel, über dem keine Ruh ist“. Es ist also nicht etwa direkt auf jedem Berg Lärm.

Aufgabe A.3. Wie lautet die Negation der folgenden Aussagen?

- Alle meine Entchen schwimmen auf dem See.
- Mein Hai und mein Delphin schwimmen im See.
- Grün grün grün sind alle meine Kleider.
- Mein Hemd oder meine Hose ist rot.
- Zu jedem Rechner gibt es jemanden, der ihn zum Absturz bringt.
- Es gibt Rechner, die nicht abstürzen.

A.3 Beweis durch Widerspruch

Der *Widerspruchs-Beweis* (auch indirekter Beweis oder *Reductio ad absurdum* genannt) ist eine der grundlegenden Beweismethoden. Kurz gesagt handelt es sich um die Methode, eine Aussage zu beweisen, indem man ihr Gegenteil zum Widerspruch führt. Man wendet sie oft sogar automatisch an, ohne es explizit zu kennzeichnen. So besagt der vierte Satz aus unserem Shakespeare-Beispiel eigentlich „Wenn ein Poet nicht meisterhaft ist, kann er nicht Hamlet geschrieben haben“, also $\text{Nicht-}C \Rightarrow \text{Nicht-}E$, wir haben ihn aber sofort in die äquivalente Aussage $E \Rightarrow C$ umgewandelt.

Der Widerspruchs-Beweis basiert auf dem fundamentalen logischen Prinzip des *Tertium non datur*: Für jede Aussagen A gilt entweder A oder Nicht- A , eine dritte Möglichkeit gibt es nicht. Außerdem wendet man die folgende Äquivalenz an:

$$(A \Rightarrow B) \Leftrightarrow (\text{Nicht-}B \Rightarrow \text{Nicht-}A)$$

In Worten: Aussage B folgt aus Aussage A genau dann, wenn aus Nicht- B Nicht- A folgt. Wichtig bei dieser Umwandlung ist erst mal, dass es dabei nicht darauf ankommt, ob die Aussagen A und B wahr oder falsch sind. Zum Beispiel ist der Satz *Wenn der Mond aus grünem Käse besteht, bin ich ein Mitteleuropäer* äquivalent zum Satz *Wenn ich kein Mitteleuropäer bin, besteht der Mond nicht aus grünem Käse* - und zwar unabhängig von der Beschaffenheit des Mondes und meiner Herkunft. Falsch wäre der Satz nur, wenn der Mond tatsächlich aus grünem Käse bestünde, aber ich nicht aus Mitteleuropa stammte.

Die oben beschriebene Äquivalenz machen wir uns nun in einem Widerspruchs-Beweis wie folgt zunutze: Wir haben A , und wollen B beweisen. Das heißt, wir möchten $A \Rightarrow B$ zeigen. Dazu nehmen wir Nicht- B an (dies ist die so genannte *Widerspruchsannahme*), und folgern daraus Nicht- A . An dieser Stelle entsteht der Widerspruch - denn dass A gilt, war ja gegeben, und aufgrund des *Tertium non datur* wissen wir, dass A und Nicht- A nicht gleichzeitig gelten können.

Wenn man solche Beweise nur abstrakt beschreibt, kommt man ziemlich schnell durcheinander. An konkreten Sätzen ist die Beweisstruktur einfacher zu sehen.

Satz A.1 (Euklid, 300 v.Chr.). *Es gibt unendlich viele Primzahlen.*

Beweis: Eine Primzahl ist eine Zahl, die nur durch 1 und sich selbst ohne Rest teilbar ist. Angenommen, es gäbe nur endlich viele Primzahlen (dies ist die Widerspruchsnahme). Dann könnte man sie in einer Liste als p_1, p_2, \dots, p_k aufzählen. Nun betrachten wir das Produkt $p_1 \cdot p_2 \cdot \dots \cdot p_k$ aller Primzahlen. Wir zählen 1 hinzu und nennen die entstehende Zahl N , also

$$N = p_1 \cdot p_2 \cdot \dots \cdot p_k + 1.$$

Dann ist N größer als alle Zahlen auf unserer Liste, also kann N keine Primzahl sein. Da jede Zahl, die keine Primzahl ist, eine Primzahl als Teiler hat, hat N eine Primzahl als Teiler. Dies kann aber keine Primzahl von unserer Liste sein, denn wenn wir N durch eine der Zahlen auf der Liste teilen, bleibt immer der Rest 1. Also kann unsere Liste nicht vollständig sein, im Widerspruch zur Annahme. Daher gibt es unendlich viele Primzahlen. \square

Aufgabe A.4. Beweisen Sie über einen Widerspruchsbeweis, dass die Wurzel jeder ungeraden Quadratzahl ungerade ist.

Aufgabe A.5. Beweisen Sie, dass $\sqrt{2}$ irrational ist. Eine Zahl heißt irrational, wenn sie nicht als Bruch zweier natürlicher Zahlen darstellbar ist. Hinweis: Nehmen Sie an, dass $\sqrt{2} = p/q$ gilt, für zwei teilerfremde Zahlen p und q .

A.4 Vollständige Induktion

Vollständige Induktion ist auch eine Beweismethode, die ständig in der Mathematik angewendet wird. Eine moderne Art das Prinzip darzustellen geht mit Hilfe von Dominosteinen. Dazu stellen wir alle Dominosteine in eine Reihe auf und wollen nun versuchen einen umzustößen, sodass auch alle anderen umfallen. Damit dies geschieht, müssen wir folgendes sicherstellen:

1. Der erste Domino-Stein wird umgeworfen. (Induktionsanfang)
2. Immer wenn ein Domino-Stein fällt, fällt auch der nächste. (Induktionsschritt)

Üblicherweise benutzt man dieses Prinzip überall dort, wo eine Aussage A für unendlich viele Objekte gezeigt werden soll - zum Beispiel für alle natürlichen Zahlen $n \in \mathbb{N}$. Die natürlichen Zahlen haben den großen Vorteil, dass sie schon der Reihe nach aufgestellt sind. Wir müssen nun noch den ersten Stein umwerfen, also A für $n = 1$ zeigen (Induktionsschritt - ist meistens einfach); dann nehmen wir an, dass der n -te Stein gefallen ist, also dass A für ein beliebiges $n \in \mathbb{N}$ gilt (Induktionsannahme), und müssen beweisen, dass es auch für $n + 1$ gilt (Induktionsschritt). Dafür verwenden wir die Induktionsannahme. Das berühmteste Beispiel ist wohl von Gauß.

Satz A.2. Für jede Zahl $n \in \mathbb{N}$ ist die Summe der ersten n natürlichen Zahlen gleich $0.5 \cdot n \cdot (n + 1)$, also

$$\sum_{i=1}^n i = \frac{1}{2} n \cdot (n + 1),$$

wobei $\sum_{i=1}^n i := 1 + 2 + \dots + n$.

Beweis: Induktionsanfang: für $n = 1$ gilt $1 = \frac{1}{2} \cdot 1 \cdot (1 + 1) = 1$. Die Aussage stimmt hier also.

Induktionsannahme: Es gelte

$$\sum_{i=1}^n i = \frac{1}{2} \cdot n \cdot (n + 1)$$

für ein $n \in \mathbb{N}$.

Induktionsschritt: Wir wollen zeigen, die Aussage gilt auch für die Zahl $n + 1$. Dazu fangen wir mit der Summe für $n + 1$ an, die wir berechnen wollen.

$$\sum_{i=1}^{n+1} i = \sum_{i=1}^n i + (n + 1) \stackrel{!}{=} \frac{1}{2} n(n + 1) + (n + 1).$$

Hier haben wir im zweiten Schritt (!) die Induktionsvoraussetzung angewendet. Nun müssen wir nur noch ein wenig umformen und erhalten

$$\frac{1}{2} n(n + 1) + \frac{1}{2} \cdot 2(n + 1) = \frac{1}{2} (n + 2)(n + 1)$$

wie gewünscht. □

Induktionen dieser Art benutzt man für alle Arten von Formeln für natürliche Zahlen. Aber man kann auch eine Induktion über die Anzahl der Knoten oder Kanten eines Graphen führen.

Aufgabe A.6. Beweisen Sie mit Hilfe vollständiger Induktion:

1.

$$\sum_{i=1}^n i^2 = \frac{n(n + 1)(2n + 1)}{6}$$

2. Die Summe der ersten n ungeraden natürlichen Zahlen ist das Quadrat einer natürlichen Zahl.

A.5 Ringschluss

Als letztes wollen wir noch die Methode des Ringschlusses vorstellen. Diese Technik kann immer dann sinnvoll angewendet werden, wenn mehrere äquivalente Aussagen A_i bewiesen werden sollen, z.B. $A_1 \Leftrightarrow A_2 \Leftrightarrow A_3 \Leftrightarrow \dots \Leftrightarrow A_k$. Anstatt jetzt „ A_1 ist äquivalent zu A_2 “, dann „ A_2 ist äquivalent zu A_3 “ usw. zu zeigen, kann man auch folgendes machen: Man zeigt, aus A_1 folgt A_2 , aus A_2 folgt A_3 , ..., aus A_{k-1} folgt A_k und aus A_k folgt wieder A_1 . Mit dieser Vorgehensweise kann man sich viele Beweisrichtungen sparen. Um das Prinzip besser zu verstehen, betrachten wir den folgenden Satz:

Satz A.3. *Seien A und B zwei Mengen. Dann sind folgende Aussagen äquivalent:*

1. $A \subseteq B$
2. $A \setminus B = \emptyset$
3. $A \cap B = A$
4. $A \cup B = B$

Beweis: Da wir insgesamt vier äquivalente Aussagen beweisen wollen, bietet sich ein Ringschluss an. Wir beginnen mit $(1) \Rightarrow (2)$ und führen einen Widerspruchsbeweis. Angenommen es existiert ein $x \in A \setminus B$. Dann gilt $x \in A$ und $x \notin B$. Damit ist $A \subseteq B$ aber falsch und wir haben somit einen Widerspruch.

Als nächste zeigen wir die Richtung $(2) \Rightarrow (3)$: Da $A \setminus B = \emptyset$, ist jedes $x \in A$ auch in B enthalten. Also gilt $A \cap B = A$. Auch die Richtung $(3) \Rightarrow (4)$ ist nicht kompliziert: Angenommen $x \in A \cup B$, aber $x \notin B$. Dann gilt $x \in A$ und die Voraussetzung $A \cap B = A$ ist nicht erfüllt. Damit haben wir einen Widerspruch. Jetzt fehlt nur noch die Richtung $(4) \Rightarrow (1)$, um den Satz vollständig zu beweisen. Sei $x \in A$. Dann gilt $x \in A \cup B$ und nach Voraussetzung somit $x \in B$. Da dies für alle $x \in A$ gilt, erhalten wir $A \subseteq B$. \square

Durch den Ringschluss haben wir uns also mindestens die Richtungen $(2) \Rightarrow (1)$, $(3) \Rightarrow (2)$ und $(4) \Rightarrow (3)$ gespart. Auch dieses werden wir uns häufig bei Beweisen zu Nutze machen.

Lösungshinweise

Zu A.1: Die Aussagen lassen sich in folgende Bestandteile zerlegen:

A ist ein grünes Krokodil D ist grün, beißt und kann nicht fliegen

B beißt und kann fliegen E ist ein grüner Frosch

C ist ein Moskauer Reptil

Dann gilt

1. $A \Rightarrow B$, 2. $C \Rightarrow D$, 3. $E \Rightarrow A$,

und somit $C \Rightarrow D \Rightarrow \text{nicht } B \Rightarrow \text{nicht } A$, d.h. Moskauer Reptilien sind keine Krokodile.

Außerdem gilt $E \Rightarrow A \Rightarrow B$, Grüne Frösche können fliegen.

Zu A.2: Wir können den einzelnen Aussagen wieder Buchstaben zuordnen:

A ist ein böses Tier D ist Bezwinger der Natur

B fressen andere Tiere E ist Mensch

C Elefanten haben Angst

Die Sätze stehen ergeben dann die folgenden Relationen:

1. $B \Rightarrow A$, 2. $C \Rightarrow D$, 3. $D \Rightarrow B$ und 4. $E \Rightarrow C$

Damit kann man beweisen: Menschen sind böse Tiere.

Zu A.3: Die Verneinungen lauten Folgendermaßen:

- Es gibt eine Ente, die nicht auf dem See schwimmt.
- Mein Hai oder mein Delphin schwimmt nicht im See.
- Ich habe mindestens ein Kleidungsstück, das nicht grün grün grün ist.
- Mein Hemd und meine Hose sind nicht rot.
- Es gibt einen Rechner, den keiner zum Absturz bringt.
- Alle Rechner stürzen ab.

Zu A.4: Angenommen, die Wurzel der ungeraden Quadratzahl k ist gerade, d.h. $\sqrt{k} = 2\ell$. Dann gilt $k = (2\ell)^2 = 4\ell^2$. Also ist k gerade. Damit haben wir einen Widerspruch.

Zu A.5: Wir nehmen an, dass $\sqrt{2} = p/q$, wobei p und q zwei teilerfremde natürliche Zahlen sind. Dann gilt $2q^2 = p^2$. Somit muss p eine gerade Zahl sein, d.h. $p = 2 \cdot \ell$ und wir können die Formel umschreiben in

$$2q^2 = p^2 = 4\ell^2.$$

Teilt man dieses durch 2 erhält man über dasselbe Argument, dass q eine gerade Zahl sein muss. Damit ergibt sich allerdings einen Widerspruch zur Annahme, dass p und q teilerfremd sind.

Zu A.6: 1. Induktionsanfang: Für $n = 1$ stimmt die Aussage. Induktionsbehauptung: Für ein $n \in \mathbb{N}$ gilt

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}.$$

Induktionsschritt (von n auf $n + 1$): Es gilt

$$\begin{aligned}\sum_{i=1}^{n+1} i^2 &= \sum_{i=1}^n i^2 + (n+1)^2 \\ &= \frac{n(n+1)(2n+1)}{6} + \frac{6(n+1)^2}{6} \\ &= \frac{[n(2n+1) + 6(n+1)](n+1)}{6} \\ &= \frac{[2n^2 + 7n + 6](n+1)}{6} \\ &= \frac{(n+2)(2n+3)(n+1)}{6}\end{aligned}$$

2. Induktionsanfang.: für $n = 1$ stimmt die Aussage. Induktionsbehauptung:
 $\sum_{i=1}^n (2i - 1) = n^2$. Induktionsschritt (von n nach $n + 1$):

$$\begin{aligned}\sum_{i=1}^{n+1} (2i - 1) &= \sum_{i=1}^n (2i - 1) + (2n + 1) \\ &= n^2 + (2n + 1) \\ &= (n + 1)^2\end{aligned}$$

Anhang B Zeichen und Symbole

Übersicht

| | | |
|-----|-----------------------------|-----|
| B.1 | Aus der Mengentheorie | 257 |
| B.2 | Aus der Arithmetik | 258 |
| B.3 | Zwei Quantoren | 259 |

Bei der Formulierung von Sätzen und Beweisen werden immer wieder mathematische Symbole und Kurzschreibweisen verwendet. Hier wurden die für dieses Buch wichtigsten zusammengestellt.

B.1 Aus der Mengentheorie

$M = \{a, b, c\}$ ist eine **Menge**, die aus den Elementen a, b, c besteht.

\mathbb{N} Die Menge der **natürlichen Zahlen**, $\mathbb{N} = \{0, 1, 2, 3, \dots\}$.

\mathbb{R} Die Menge der **reellen Zahlen**.

$a \in M$ Das Element a **ist** in der Menge M **enthalten**. Ist $M = \{1, 3, 4\}$, dann gilt $4 \in M$.

$a \notin M$ Das Element a **ist nicht** in der Menge M **enthalten**. Für $M = \{1, 3, 4\}$ ist $5 \notin M$.

$A \subseteq B$ Die Menge A ist eine **Teilmenge** von B . Sei $B = \{a, b, c, d\}$, dann ist $A = \{a, b\}$ eine Teilmenge von B . Die Menge $C = \{f, a\}$ ist keine Teilmenge von B .

$A \subsetneq B$ Die Menge A ist eine **echte Teilmenge** von B , d.h. es gilt $A \subset B$ und es existiert ein Element $a \in B$, dass nicht in A enthalten ist.

$A = B$ Zwei Mengen heißen **gleich**, wenn sie die gleichen Elemente enthalten. Im Fall $A = \{1, 2, 3\}$ und $B = \{1, 3, 2\}$ gilt $A = B$.

| | |
|----------------------|--|
| $A \cup B$ | Die Vereinigung von A und B enthält alle Elemente, die in A oder in B enthalten sind. Sei $A = \{1, 2, 3\}$ und $B = \{2, 4, 5\}$, dann ist $A \cup B = \{1, 2, 3, 4, 5\}$ |
| $A \cap B$ | Der Schnitt von A und B enthält nur Elemente, die sowohl in A als auch in B enthalten sind. Gilt $A = \{1, 2, 3\}$ und $B = \{2, 4, 5\}$, dann ist $A \cap B = 2$. |
| $A \setminus B$ | bezeichnet A ohne B und enthält alle Elemente, die in A aber nicht in B enthalten sind. Ist $A = \{1, 2, 4\}$ und $B = \{1, 2, 3\}$ gegeben, dann gilt $A \setminus B = \{4\}$. |
| $B \otimes C$ | Das Kreuzprodukt von B und C besteht aus allen 2-elementigen Teilmengen $\{a_1, a_2\}$ mit $a_1 \in B$ und $a_2 \in C$. Sei z.B. $B = \{1, 2, 3\}$ und $C = \{\text{Apfel}, \text{Birne}\}$, dann ist das Element $\{1, \text{Apfel}\}$ in $B \times C$. |
| $ M \in \mathbb{N}$ | gibt die Anzahl der Elemente in M an. Gilt $M = \{1, 2, 3\}$, dann ist $ M = 3$. |

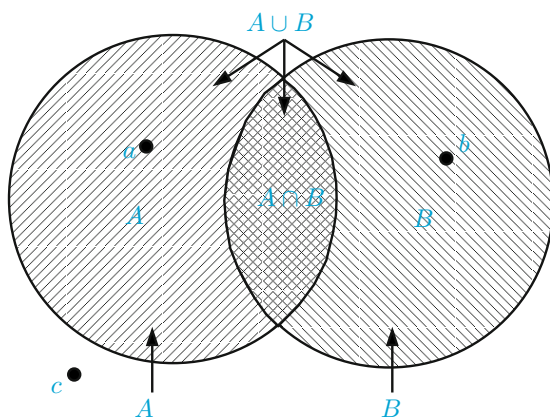


Abb. B.1: Die meisten mengentheoretischen Begriffe kann man sich leicht an Bildern veranschaulichen. Hier gilt $a \in A$, aber $c \notin A$.

B.2 Aus der Arithmetik

| | |
|---------------------|--|
| $\sum_{i=1}^n x_i$ | ist eine verkürzte Schreibweise für die Summe über alle Elemente x_1 bis x_n , d.h. $\sum_{i=1}^n x_i := x_1 + x_2 + \dots + x_n$. |
| $\prod_{i=1}^n x_i$ | ist eine verkürzte Schreibweise für das Produkt über alle Elemente x_1 bis x_n , d.h. $\prod_{i=1}^n x_i = x_1 \cdot x_2 \cdot \dots \cdot x_n$. |
| $3n$ | ist eine verkürzte Schreibweise für $3 \cdot n$. |

$f : X \rightarrow Y$ ist eine **Funktion**, die jedem Element $x \in X$ ein Element $y \in Y$ zuordnet. Zum Beispiel ordnet $f : \mathbb{R} \rightarrow \mathbb{R}$ mit $f(x) = x^2$ jedem Element $x \in \mathbb{R}$ den Wert x^2 zu.

$\sum_{x \in X} f(x)$ Verkürzte Schreibweise für die Summe aller Werte $f(x)$ der Elemente $x \in X$. Sei $X = \{1, 2, 3\}$ und $f(x) = x^2$, dann gilt

$$\sum_{x \in X} f(x) = \sum_{x \in \{1, 2, 3\}} x^2 = 1 + 4 + 9 = 14.$$

$x^* = \max\{X\}$ Das Element x^* ist der **maximale Wert** aller Werte aus X . Für $X = \{1, 3, 25\}$ gilt $x^* = 25$.

$f^* = \max_{x \in X} f(x)$ Der Wert f^* entspricht dem maximalen Wert von f bezüglich aller Elemente $x \in X$. Falls $f(x) = x^2$ und die Menge $X = \{-2, 1, 5\}$ gegeben ist, gilt

$$f^* = \max_{x \in \{-2, 1, 5\}} x^2 = \max\{4, 1, 25\} = 25.$$

B.3 Zwei Quantoren

Quantoren verknüpfen Variablen und Aussagen. Die beiden wichtigsten sind der Quantor \exists (es gibt) und der Quantor \forall (für alle).

$\exists x \in X : x > 0$ Gesprochen: Es **existiert** mindestens ein x aus der Menge X mit der Eigenschaft, dass x größer als Null ist.

$\forall x \in X : x > 0$ Gesprochen: **Für alle** x aus der Menge X gilt, dass sie größer als Null sind.

Literaturverzeichnis

- [1] Bang-Jensen J, Gutin G (2001). *Digraphs: Theory, Algorithms and Applications*. Springer, London
- [2] Berge C (2001). *Theory of Graphs*. Dover Publ. Inc.
- [3] Beutelspacher A (2009). *"Das ist o. B. d. A. trivial!": Tipps und Tricks zur Formulierung mathematischer Gedanken*. Vieweg + Teubner Verlag
- [4] Beutelspacher A, Zschiegner M (2007). *Diskrete Mathematik für Einsteiger: Mit Anwendungen in Technik und Informatik*. Vieweg + Teubner Verlag
- [5] Biggs N (2002). *Discrete Mathematics*. Oxford University Press
- [6] Bollobas B (2002). *Modern Graph Theory*. Springer, Berlin
- [7] Brandstädt A (1994). *Graphen und Algorithmen*. Teubner Verlag
- [8] Brualdi R (2008). *Introductory Combinatorics*. Prentice Hall
- [9] Chartrand G, Ollermann O (1993). *Applied and Algorithmic Graph Theory*. McGraw-Hill
- [10] Cormen T, Leiserson Ch, Rivest R, and Stein C (2001). *Introduction to Algorithms*. McGraw-Hill Book Company
- [11] Diestel R (2006). *Graphentheorie*. Springer, Berlin
- [12] Ford L, Fulkerson D (1962). *Flows in Networks*. Princeton University Press
- [13] Foulds L (1992). *Graph Theory Applications*. Springer, New York
- [14] Garey M, Johnson D (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman
- [15] Gondran M, Minoux M (1984). *Graphs and Algorithms*. Wiley
- [16] Grimaldi R (2003). *Discrete and Combinatorial Mathematics. An Applied Introduction*. Addison-Wesley Longman
- [17] Halin R (1989). *Graphentheorie*. Akademie-Verlag Berlin/Wissenschaftliche Buchgesellschaft Darmstadt
- [18] Holton D, Sheehan J (1993). *The Petersen Graph*. Cambridge University Press
- [19] Hußmann St, Lutz-Westphal B (2007). *Kombinatorische Optimierung erleben*. Vieweg
- [20] Jacobs K, Jungnickel D (2003). *Einführung in die Kombinatorik*. DeGruyter

- [21] Jungnickel D (1994). *Graphen, Netzwerke und Algorithmen*. Springer, Berlin
- [22] Korte B, Vygen J (2007). *Combinatorial Optimization: Theory and Algorithms (Algorithms and Combinatorics)*. Springer
- [23] Kreher D, Kocay W (2004). *Graphs, Algorithms, and Optimization*. Chapman & Hall
- [24] Lawler E (1976). *Combinatorial Optimization: Networks and Matroids*. Saunders College Publishing
- [25] Lovász L, Pelikán J, Vesztergombi K (2003). *Diskrete Mathematik*. Springer, New York
- [26] Läuchli P (1997). *Algorithmische Graphentheorie*. Birkhäuser Verlag
- [27] Matousek J, Nešetřil J, Milke H (2007). *Diskrete Mathematik: Eine Entdeckungsreise*. Springer, Berlin
- [28] Nitzsche M (2005). *Graphen für Einsteiger - Rund um das Haus vom Nikolaus*. Vieweg
- [29] Ruhe G (1991). *Algorithmic Aspects of Flows in Networks*. Springer Netherlands
- [30] Tucker A (2006). *Applied Combinatorics*. John Wiley & Sons
- [31] van Lint J, Wilson R (2001). *A Course in Combinatorics*. Cambridge University Press
- [32] West D (1996). *Introduction to Graph Theory*. Prentice-Hall Inc.
- [33] Wilson R (2002). *Graphs, Colourings and the Four-Colour Theorem*. Oxford University Press
- [34] Wilson R (2004). *Four Colors Suffice: How the Map Problem Was Solved*. Princeton University Press

Index

- Abstand
 - euklidischer Abstand, 79
 - Maximum-Abstand, 79
- Adjazenzmatrix, 33, 159
- Algorithmus, 84
 - Breitensuche, 21, 25, 34–36
 - Cycle-Canceling, 193
 - Dijkstra, 149, 154, 159
 - Doppelte-Baum-, 82
 - effizient, 84
 - Färbungsalgorithmus, 113
 - für gerichtete Euler-Touren, 133
 - Greedy-TSP-, 80
 - Laufzeit, 84
 - Moore-Bellman-Ford, 157
 - polynomiell, 84
 - Succesive-Shortest-Path, 193, 197
 - Tiefensuche, 21, 28, 34–36, 141
 - von Edmonds-Karp, 175
 - von Fleury, 61, 65, 68
 - von Ford-Fulkerson, 174, 181
 - von Hierholzer, 61, 62, 68, 130
 - von Kruskal, 39, 41, 51, 54
 - von Prim, 39, 43, 51, 53
- Artikulationspunkt, 12, 18, 19
- b*-Fluss, 189
- Balance, 188
- Baum, 14, 18, 19, 73, 108, 112, 127, 133, 141, 147, 220
 - Gewicht, 40
 - minimal spannender, 39, 40, 82, 161
 - spannender, 21, 22, 40
- Baumkante, 46
- bipartit, 7, 29, 69, 88, 102, 105, 108, 207, 209
- Blatt, 14
- Brücke, 19
- Breitensuche, 21, 25, 34–36, 127
- Brücke, 13, 18, 64, 130
- chromatische Zahl, 111, 112
- Cycle-Canceling-Algorithmus, 193
- Darstellung, 3, 92
 - outerplanare, 95
- Dekompositionssatz von Ford-Fulkerson, 163, 168
- Diamantgraph, 175, 193
- Digraph, 125
- Dijkstra-Algorithmus, 149, 154, 159
- Diktator, 138
- $\text{dist}(v)$, 149
- Doppelte-Baum-Algorithmus, 82
- Dreiecksungleichung, 81, 150
- eben, 92
- Endknoten, 9
- Entscheidungsproblem, 84
- euklidischer Abstand, 79
- Euler-Formel, 98, 99, 101
- Euler-Graph, 57, 58, 132
- Euler-Kreis, 57, 58, 72
- Euler-Tour, 58, 130, 132
- Euler-Weg, 65, 66, 133
- eulersch, 57, 58, 69, 72, 82, 87, 108, 132, 142
- exponiert, 205
- f*-augmentierender Weg, 173
- f*-augmentierender Zykel, 190
- Farbe, 112
- Färbung, 112
- Färbungsalgorithmus, 113
- Fläche, 92
 - äußere, 93
 - Grad, 93
 - innere, 93
 - Länge, 93
- Fluss, 163, 187
 - b*-Fluss, 189
 - Kosten, 189
 - kostenminimaler, 187, 189
 - maximaler, 169
 - (s, t) -Fluss, 167
 - Wert, 167
- Flusserhaltung, 167
- Fundamentalkreis, 44
- Fundamentalschnitt, 46
- Fünf-Farben-Satz, 118
- Fünf-Prinzen-Problem, 116
- Gas-Wasser-Strom-Problem, 91
- Grad, 5
 - Ausgrad, 126
 - Flächen, 93
 - Ingrad, 126
 - Knoten, 5
 - Maximalgrad, 5
 - Minimalgrad, 5, 9, 76
- Graph, 2
 - bipartiter, 7, 29, 69, 88, 102, 105, 108, 207, 209
 - Diamantgraph, 175, 193
 - Digraph, 125
 - dualer, 116
 - einfacher, 2
 - Euler-Graph, 57, 58, 132
 - eulerscher, 57, 58, 69, 72, 82, 87, 108, 142
 - gerichteter, 3, 125, 164
 - Hamilton-Graph, 71, 72

- hamiltonscher, 71, 72, 108, 134, 136, 220
- k -regulärer, 88, 109, 212, 220
- Komplementärgraph, 88
- kreisfreier, 14
- Multigraph, 58
- outerplanarer, 95
- Petersen-Graph, 7, 75, 102, 220
- planarer, 7, 91, 92
- unzusammenhängender, 11
- vollständig bipartiter, 7
- vollständiger, 6, 78
- Würfelgraph, 7, 92
- zugrunde liegender ungerichteter, 125
- zusammenhängender, 11, 128
- Greedy-TSP-Algorithmus, 80

- Hamilton-Graph, 71, 72
- Hamilton-Kreis, 71, 72, 97, 194
- Hamilton-Weg, 134
- hamiltonsch, 71, 72, 108, 134, 136, 220
- Handlungsreisenden-Problem, 78
- Handshaking-Lemma, 5, 103
- Haus vom Nikolaus, 65
- Heiratsbedingung, 210
- Hühner-Satz von Landau, 137

- inzidieren, 4

- k -färbbar, 112
- k -regulär, 88, 109, 212, 220
- Kante, 2
 - adjazent, 4
 - benachbart, 4
 - Nicht-Baumkante, 44
 - parallele, 2
 - Schlinge, 2
- kantendisjunkt, 66
- Kantenkapazität, 167
- Kantenzug, 9
- Klasse NP, 85
- Klasse P, 85
- Knoten, 2
 - adjazent, 4
 - Artikulationspunkt, 12
 - benachbart, 4
 - Maximalgrad, 5
 - Minimalgrad, 5, 9, 76
 - Nachbarn, 4
- Knotenfärbung, 111
- Knotenkapazität, 182
- Knotenüberdeckung, 214
 - minimale, 214
- Komplexitätstheorie, 84
- König, 137
- Königsberger Brückenproblem, 57
- konservativ, 156
- Kontraktion, 106
- konvex, 99

- Körper, 99
- Kreis, 9
 - einfacher, 9
 - Euler-Kreis, 57, 58, 72
 - Fundamentalkreis, 44
 - gerichteter, 126
 - Hamilton-Kreis, 71, 72, 97, 194
- Kreiskriterium, 48
- kürzester Weg, 31, 145, 146
- Kürzester-Wege-Baum, 145, 147

- Länge
 - Fläche, 93
 - Weg, 9
- Laufzeit, 84

- M -alternierender Weg, 205
- M -augmentierender Weg, 205
- Matching, 203, 204
 - maximales, 204
 - perfektes, 204
 - überdeckendes, 209
- Max-Fluss-Min-Schnitt-Satz, 176, 180
- Maximum-Abstand, 79
- minimal spannender Baum, 39, 40, 82, 161
- Minor, 106
- Moore-Bellman-Ford-Algorithmus, 157
- Multigraph, 58

- Netzwerk, 167

- Optimalitätskriterium
 - für Kürzesten-Wege-Baum, 150
 - für kostenminimale Flüsse, 192
 - für maximale Flüsse, 179
 - für maximale Matchings, 206
 - für MST, 48
- Optimierungsproblem, 84
- Orientierung, 125
- outerplanar, 95

- Partner, 210
- Petersen-Graph, 7, 75, 102, 220
- planar, 7, 91, 92
- $\text{pred}(v)$, 151

- Quelle, 188

- Rad, 112
- Residualgraph, 171
- Residualkapazität, 171
- Residualkosten, 191
- Rückwärtskante, 171, 176

- Satz

- zur Existenz eines
 - Kürzesten-Wege-Baumes, 148
- Dekompositionssatz von
 - Ford-Fulkerson, 163, 168
- Fünf-Farben-Satz, 118
- Hühner-Satz von Landau, 137
- Max-Fluss-Min-Schnitt-Satz, 176, 180
- Vier-Farben-Satz, 120
- von Berge, 206
- von Brooks, 114
- von Caley, 41
- von De Morgan, 116
- von Dirac, 76
- von Hall, 209, 210
- von König, 213, 214
- von Kuratowski, 104, 106
- Schnitt, 10, 14, 45
 - Fundamentalschnitt, 46
 - gerichteter, 126, 128
 - Kapazität, 177
 - (s, t) -Schnitt, 176
- Schnittkriterium, 48
- Senken, 188
- (s, t) -Fluss, 167
- (s, t) -Schnitt, 176
- stark zusammenhängend, 128, 136
- Succesive-Shortest-Path-Algorithmus, 193, 197
- symmetrische Differenz, 206

- Tailenweite, 102
- Teilgraph, 8
 - induzierter, 8
- Tiefensuche, 21, 28, 34–36, 141
- Travelling-Salesman-Problem, 78
- Turniergraph, 134, 136

- Untergraph, 8

- Vier-Farben-Problem, 115
- Vier-Farben-Satz, 120
- Vorwärtskante, 171, 176

- Weg, 9
 - $\text{dist}(v)$, 149
 - einfacher, 9
 - Euler-Weg, 65, 66, 133
 - f -augmentierender, 173
 - Gewicht, 146
 - Hamilton-Weg, 134
 - kürzester, 31, 145, 146
 - Länge, 9
 - M -alternierender, 205
 - M -augmentierender, 205
- Wegfluss, 164
 - Flusswert, 165
 - Wert, 165
 - zulässiger, 165
- Würfelgraph, 7, 92

- Zusammenhang, 10
- zusammenhängend, 11, 128
- Zusammenhangskomponente, 11, 29, 74, 100
 - starke, 128
- Zykel, 126
 - f -augmentierender, 190
 - negativer, 156