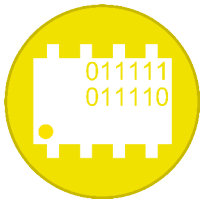
Deutsche JuniorAkademien

JuniorAkademie Adelsheim

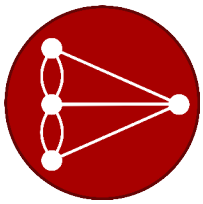
12. Science Academy Baden-Württemberg 2014



Digitaltechnik

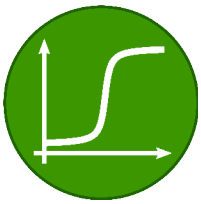
Geophysik

Geschichte/Germanistik



Mathematik

Physik



TheoPrax

H.W. & J. Hector Stiftung

Regierungspräsidium Karlsruhe
Abteilung 7 – Schule und Bildung

Dokumentation der JuniorAkademie Adelsheim 2014

12. Science Academy Baden-Württemberg

Träger und Veranstalter der JuniorAkademie Adelsheim 2014:
Regierungspräsidium Karlsruhe
Abteilung 7 –Schule und Bildung–
Hebelstr. 2
76133 Karlsruhe
Tel.: (0721) 926 4454
Fax.: (0721) 933 40270
E-Mail: georg.wilke@scienceacademy.de
petra.zachmann@scienceacademy.de
www.scienceacademy.de

Die in dieser Dokumentation enthaltenen Texte wurden von den Kurs- und Akademieleitern sowie den Teilnehmern der 12. JuniorAkademie Adelsheim 2014 erstellt. Anschließend wurde das Dokument mit Hilfe von LaTeX gesetzt.

Gesamtredaktion und Layout: Jörg Richter
Druck und Bindung: RTB Reprotechnkik Bensheim

Vorwort

Die Science-Academy Baden-Württemberg fand in diesem Jahr bereits zum 12. Mal am Landesschulzentrum für Umwelterziehung auf dem Eckenberg in Adelsheim statt. Gemeinsam mit einem fast 30-köpfigen Leiterteam verbrachten hier rund 70 Schülerinnen und Schüler aus ganz Baden-Württemberg die zweiwöchige Sommerakademie, das Eröffnungswochenende und das Dokumentationswochenende.

Am Eröffnungswochenende stehen sich die Teilnehmerinnen und Teilnehmer gegenüber, ohne sich jemals zuvor begegnet zu sein. Am Dokumentationswochenende hat sich jeder von ihnen nicht nur in die wissenschaftlichen Inhalte seines Kurses vertieft, sondern sich auch persönlich weiterentwickelt.

Während der gemeinsamen Zeit wurde aus diesen einzelnen Personen eine große Gemeinschaft. Es entstand eine Atmosphäre, die die Zeit zu einer sehr besonderen machte. Das Gefühl, im Kurs eine „bahnbrechende“ Erkenntnis gewonnen zu haben und etwas Neues ausprobiert zu haben, trägt ebenso dazu bei wie das Gefühl, seine Grenzen kennengelernt zu haben, vielleicht überwunden zu haben, zumindest aber daran gewachsen zu sein. Auch sind es der respektvolle Umgang miteinander und der Raum für Kreativität und Individualität, die diese besondere Atmosphäre entstehen ließen und sie prägten.

Um der Gemeinschaft zusätzlich einen gemeinsamen Rahmen zu geben, steht jede Akademie unter einem bestimmten Motto. In diesem Jahr war es das „Glück“, das uns über die Zeit hinweg begleitete. Glück ist ein weiter Begriff und hat unwahrscheinlich viele Facetten. Für uns standen die kleinen Glücksmomente, die Freundschaften, die entstehen und die Erfahrungen, die hier gemacht werden, im Vordergrund. Für einige Glücksmomente sorgte unser *geheimer Freund*, der

uns immer wieder mit kleinen Aufmerksamkeiten überrascht hat. Gemeinsam haben wir viele schöne Momente erlebt. Diese Momente werden uns immer begleiten und in Erinnerung bleiben.

Am Ende haben wir die Akademie wieder durch die Akademietür verlassen, und unsere gemeinsame Zeit ist zu Ende gegangen. Doch eines möchten wir euch mit auf dem Weg geben: Eure Wege werden sich wieder kreuzen! Die Freundschaften, die hier entstehen, halten oft noch über Jahre hinweg, und die Erfahrungen und die Erkenntnisse, die ihr hier gewonnen habt, gehen euch nie mehr verloren. Geht also mit offenen Augen durchs Leben und haltet Ausschau nach neuen „Türen“. Habt den Mut, sie zu öffnen und durch sie hindurch zu gehen.

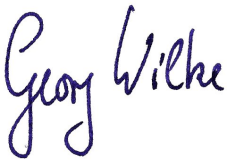
Wir wünschen euch alles Liebe und Gute für euren weiteren Weg und für das, was als nächstes auf euch zukommt. Wir freuen uns sehr darauf, euch bald – in egal welchem Zusammenhang – wieder zu sehen. Vielleicht ja sogar wieder hier in Adelsheim!

Viel Spaß beim Lesen und Schmökern!

Eure/Ihre Akademieleitung

Patricia Keppler (Assistenz)

Nico Röck (Assistenz)



Georg Wilke

Dr. Petra Zachmann

Inhaltsverzeichnis

Kurs 4 – Mathematik/Informatik: Über Routenplaner, eingefärbte Landkarten und das Problem des Handlungsreisenden

Vorwort

Die Kursleiter

Graphentheorie – viele Schüler, die diesen Begriff hören, können nicht viel damit anfangen. Er lässt höchstens bei manchen die Abscheu zur Mathematik hörbar werden. Dabei ist dieses Thema ganz anders als jene, die man in der Schule behandelt. In unserem Kurs ging es weniger um lange und komplizierte Formeln, als um verschiedene, anwendungsbezogene und anschauliche Sachverhalte. Was genau sind diese „Sachverhalte“? Spätestens wenn man sich wieder einmal verfahren hat und hilflos versucht, mit einer Landkarte den Weg aus dem tiefsten Stadtgetümmel oder dem dunkelsten Wald zu finden, hilft in der heutigen Zeit nur noch eins: der Griff zum Navigationsgerät.

Doch wie berechnet ein Navigationsgerät einen kürzesten, billigsten oder ökologischsten Weg? Hier kommt die Mathematik der Graphentheorie ins Spiel. Genau mit diesem Thema beschäftigte sich unser Kurs im Rahmen der Science-Academy 2014 in Adelsheim. Schon am Eröff-

nungswochenende wurde auf spielerische Weise der erste Graph mithilfe eines Wollknäuels zwischen den Teilnehmern zum Erfassen ihrer Wohnorte aufgespannt. Im Rahmen der Sommerakademie entwickelte sich dies zu der Herausforderung, eine möglichst kurze Rundreise zwischen den Wohnorten aller Teilnehmer zu finden, wofür unter anderem ein Computerprogramm in der Programmiersprache Java geschrieben wurde. Dies ist nur ein Beispiel für das, was während der Akademie im Sommer zustande kam: Es wurde gezeichnet, gezählt, programmiert und bewiesen, bis die Köpfe rauchten. Dabei wurden nicht nur fachliche Probleme gelöst, sondern vor allem auch neue Freundschaften geschlossen.

An dieser Stelle möchten wir uns bei den Teilnehmern, die immer offen für Neues waren und kontinuierlich, aktiv und motiviert mitgearbeitet haben, für die angenehme und lehrreiche Kurszeit bedanken. Es waren zwei schöne Wochen im Sommer, aus denen hoffentlich jeder, angesteckt von der Begeisterung zur Mathematik, Informatik und Graphentheorie, etwas mitnehmen konnte und angeregt wurde, sich weiter mit diesen Themen zu beschäftigen.

Wer nun Lust an diesen interessanten Themen gefunden hat, der sei herzlich dazu eingeladen die folgende Dokumentation über unsere Spaziergänge mit Euler, das Besuchen von sieben Brücken im schönen Königsberg oder das systematische Drehen von farbigen Würfeln und noch vielem mehr aus unserer Kursarbeit auf sich wirken zu lassen.

Unser Kurs

Johanna Buck, Johanna Kaps

Die Teilnehmer

Amelie brachte mit ihrem warmherzigen Lächeln und ihrer Fröhlichkeit immer gute Laune mit in den Kurs. Auch sonst hatte man mit ihr viel Spaß und konnte häufig mit ihr lachen. Der durch und durch liebenswürdige Lockenkopf war im Fußballspielen erste Klasse, was sie in den Mittagspausen exzessiv auslebte. Lustig und einfach nett war sie immer für uns da.

Benjamin war allen mit seiner lustigen Art meist schon einen Schritt voraus, sowohl im Denken als auch im Reden. Er rätselte gern an Problemen und kam meist schnell zu einer guten Lösung. War man nicht ganz seiner Meinung, musste man sich ganz schön anstrengen, um ihn vom Gegenteil zu überzeugen. Mit seinen leicht provozierenden Worten „Beweis es mir!“ brachte er uns häufig zum Nachdenken und wir fragten uns oft, wo er die viele Energie zum Reden, Rätseln und Diskutieren hernahm.

Christian H. half uns allen auf seine logisch denkende Art viel im Kurs weiter. Auf ihn war stets Verlass bei Gruppenarbeiten, bei denen er uns gerne Sachen noch einmal erklärte, wenn wir sie noch nicht ganz verstanden hatten. Total entspannt kann er super vortragen und war daher ein gutes Vorbild bei den Präsentationen. Wir hatten mit ihm viel Spaß in unserem Kurs, in den er jeden Tag aufs Neue gut gelaunt kam.

Christian M. Neben unserem Preis vom Sportfest war Christian unser großzügiger Süßigkeiten- und Kuchenlieferant. Er hatte während der Science-Academy Geburtstag und teilte seinen Kuchen und das Geburtstagspaket seiner Eltern freigiebig mit uns. Doch nicht nur das: Mit seinen mathematischen Fähigkeiten und vor allem seinem Wissen in der Informatik war er einer der besten Programmierer unseres Kurses. Seine produktiven Ideen und sein großes Interesse an unserem Thema bereicherten den Kurs in jeder Hinsicht.

David Unser Programmier-Ass war eindeutig David. Unser bester und schnellster Informatiker hatte geniale Ideen für jegliche Problemstellungen. Seine Erklärungen danach waren immer so verständlich, dass er der Lieblingsgruppenpartner aller Kursteilnehmer war. Außerdem steckt ein ehrgeiziger Mathematiker in ihm: Er hat immer solange nachgedacht, bis er eine Lösung fand. Mit seinen Beiträgen und Talenten war er ein wirklich nettes und hilfsbereites Mitglied unseres Kurses, und wir konnten uns als Anlaufstelle für jegliche Fragen immer auf ihn verlassen.

Gözde war immer zur Stelle, wenn es etwas auf das Flipchart zu schreiben galt, da ihre Plakate eindeutig am besten und übersichtlichsten waren. Nicht zuletzt lag dies an ihrer perfektionistischen und pragmatischen Ader und natürlich an ihrer schönen Handschrift. Sie war immer gut drauf, hilfsbereit, offen und man konnte sie zu Allem begeistern. Mit ihrer direkten, aber netten Art hat sie unseren Kurs extrem bereichert und um einiges produktiver gemacht.

Jana ist aufmerksam, nett, schlau und hilfsbereit. Mit ihrer sympathischen Art war sie eine super Stütze in unserem Kurs. Egal welche mathematische Frage kam, Jana konnte sie beantworten. Selbstbewusst und zuverlässig war sie ein tolles Kursmitglied mit ständig guter Laune und immer bereit, uns mit ihrem Wissen im Kurs sowohl bei den Beweisen als auch beim Programmieren weiterzuhelfen. Zu ihren Stärken gehörte auf alle Fälle auch Präsentieren, was ihr besonders in der Abschlusspräsentation zugute kam.

Johanna B. brachte sich mit ihrer netten und freundlichen Art viel im Kurs ein. Super sympathisch und lustig hielt sie uns alle mit ihren Späßen bei Laune. Von unseren Kursleitern auch „Barbara“ genannt – um sie leichter von der anderen Johanna zu unterscheiden – machte sie so ziemlich jeden Blödsinn mit. Zusammen mit besagter anderer Johanna tauschte sie Namensschilder, war immer gut gelaunt und fröhlich – so kennen wir Johanna.

Johanna K. Die sommersprossige Johanna ist frech, quirlig und spaßig. Sie fand, besonders mit unserer anderen Johanna, den ganzen Tag etwas zu lachen. Schon von Anfang an konnte die aufgedrehte Fünfzehnjährige ihre Bewegungslust kaum in Zaum halten und hat dies deshalb mit Turnen auf der Wiese, beim Sportplatz und Fußballspielen ausgeglichen. Man konnte sich meist auf ihre gute Laune verlassen und viel Quatsch mit ihr machen. Mit der anderen Johanna zusammen hat sie unserem Schülermentor Frank ein neues Namensschild mit seinem erfundenen Zweitnamen „Horst“ gebastelt.

Julia war stets gut gelaunt und freundlich. In Mathe hatte sie den totalen Durchblick und konnte uns deshalb immer helfen und uns etwas erklären. Mit ihr ging es lustig zu und es hat Spaß gemacht, mit ihr zusammenzuarbeiten. Sie fragte sofort nach, wenn etwas unklar war und beeinflusste durch ihr unkompliziertes und freundschaftliches Verhalten unser Kursklima sehr positiv. Man traf sie oft beim Sport oder mit ihren Freunden. Durch ihre ruhige Art war sie einfach ein tolles Kursmitglied.

Karen hat alle Kursinhalte und vor allem das Programmieren besonders schnell verstanden. Neben ihrer selbstbewussten und ruhigen Art sowie ihrem mathematischen Wissen hatte sie ein großes Präsentationstalent: Bei unserer Abschlusspräsentation trat sie ohne jegliche Angst, frei und souverän auf. Außerdem konnte sie wahnsinnig gut Graffiti zeichnen. Doch auch der Sport kam bei ihr nicht zu kurz: Egal in welcher Mittagspause, Karen war immer bei der Tischtennisplatte anzutreffen und hat dort leidenschaftlich gespielt.

Niklas war immer gut gelaunt und war uns vor allem im Bereich Mathematik oftmals schon einen Schritt voraus: Er hat die Kursinhalte immer schnell verstanden und konnte Beweise besonders gut nachvollziehen. Leider war es ihm aus gesundheitlichen Gründen nur möglich, unseren Kurs bis zur Mitte der ersten Woche mit seinen regen Beiträgen zu bereichern.

Die Kursleiter

Daniel ist von der Science-Academy nicht mehr wegzudenken. Er hatte stets anschauliche Beispiele parat und half uns mit seinen ausführlichen Erklärungen bei Fragen weiter. Er brachte durch seine lustige Art jeden Tag gute Laune und viel Spaß mit in den Kurs. Daniel lehrte uns vor allem das Programmieren und hatte viel Geduld mit denen, die dabei mehr Probleme hatten. Doch auch über unseren Kurs hinaus war er immer aktiv: Als selbst sehr guter und begeisterter Fotograf bot er die Foto-KüA an. Außerdem sorgte er beim Grillen und beim Bergfest

mit seiner Gitarre und seiner Stimme für gute Laune. Daniel war ein super Kursleiter und wir alle beneiden seine Schüler darum, dass sie jeden Tag von seiner lustigen und hilfsbereiten Art profitieren können.

Unsere Kursleiter Tina und Daniel.

Frank war stets hilfsbereit: Beim Programmieren nahm er sich meist der langsamsten Gruppe an, führte uns kompetent in einige Unterthemen der Graphentheorie ein und spielte immer wieder den Laufburschen ins Akademieleitungsbüro. Beim Sportfest motivierte er uns so einmalig und laut, dass wir dank ihm gewinnen konnten. Frank war immer für jeden Quatsch zu haben und nahm es mit Humor, als wir ihm ein neues Namensschild mit der Aufschrift „Horst“ verpassten. Er wischte stets zuverlässig und mehr oder weniger sauber die Tafel und überraschte uns mit seiner musikalischen Ader am Hausmusikabend im Orchester. Er war durch und durch geduldig und nett zu uns und daher ein unschlagbar toller Schülermentor.

Tina Unsere junge Kursleiterin Tina weiß definitiv alles über Graphentheorie, was man wissen sollte. Sie hatte es mit uns zwar nicht immer leicht, doch zielstrebig und hochmotiviert schaffte sie es jedes Mal, uns etwas Neues beizubringen. Sie erklärte uns Sachverhalte so lange und geduldig, bis es auch der Letzte verstanden hatte. Mathematische Induktionsbeweise waren ihre Lieblingsart, uns die Graphentheorie näher zu bringen. Dabei schrieb sie gern und oft die Tafel voll. Sie war immer gut gelaunt, für uns da und erklärte uns wieder und wieder gerne alles, was wir noch nicht verstanden hatten. Auf ihre ruhige und genaue Art war sie einfach eine tolle Kursleiterin.

Beweistechniken

Johanna Buck, Karen Wiegand

In die Graphentheorie fließen zahlreiche mathematische Sätze ein, die wir nicht nur anwenden, sondern auch beweisen wollten. Hierfür haben wir verschiedene Beweistechniken kennengelernt, die wir an einigen Beispielen erklären wollen. Dazu benötigen wir folgende Vorüberlegung: Da *gerade* Zahlen stets durch 2 teilbar sind, kann man diese auch als $2n$ darstellen, wobei n eine ganze Zahl ist. Addiert man zu einer beliebigen geraden Zahl die Zahl 1, dann erhält man eine *ungerade* Zahl. Somit lässt sich diese als $2n + 1$ darstellen.

Direkter Beweis

Beim direkten Beweis fängt man mit der Voraussetzung an und folgert daraus die zu zeigende Aussage.

Beispiel: *Wenn a und b zwei ungerade natürliche Zahlen sind, dann ist die Summe aus a und b gerade.*

Beweis.

Die Zahlen a und b sind ungerade, also schreiben wir $a = 2n + 1$ und $b = 2m + 1$. Dann gilt

$$\begin{aligned} a + b &= (2n + 1) + (2m + 1) \\ &= 2(n + m + 1). \end{aligned}$$

Also ist $a + b$ eine gerade Zahl. □

Direkter Beweis durch Gegenbeispiel

Durch ein passendes Gegenbeispiel wird gezeigt, dass eine Behauptung nicht stimmt.

Indirekter Beweis

Beim indirekten Beweis wird zunächst eine Umstellung des Satzes, die so genannte *Kontraposition*, gebildet, d. h. die Satzteile werden umgedreht und zusätzlich verneint. Zum Beispiel wird „Wenn n^2 gerade ist, dann ist auch n gerade.“ zu „Wenn n ungerade ist, dann ist auch n^2 ungerade.“ Die Kontraposition wird anschließend direkt bewiesen.

Äquivalenzbeweis

Die Aussage des Äquivalenzbeweises zeichnet sich durch die Verwendung des Ausdrucks „genau dann“ aus. Die Aussage muss deshalb in zwei Richtungen bewiesen werden: Die von dem ersten Teil des Satzes zum zweiten Teil und später genau umgekehrt. Bei den einzelnen Beweisrichtungen ist die Beweistechnik frei wählbar.

Beispiel: *Eine ganze Zahl n ist genau dann gerade, wenn n^2 gerade ist.*

Richtung 1: „Wenn n gerade ist, dann ist auch n^2 gerade.“

Beweis.

Da n eine gerade Zahl ist, können wir $n = 2m$ einsetzen und erhalten

$$n^2 = (2m)^2 = 4m^2 = 2(2m^2),$$

was eine gerade Zahl ist. □

Richtung 2: „Wenn n^2 gerade ist, dann ist auch n gerade.“

Beweis.

Diese Richtung wollen wir mit einem indirekten Beweis zeigen. Also bilden wir die Kontraposition: „Wenn n ungerade ist, dann ist auch n^2 ungerade.“

Da n eine ungerade Zahl ist, schreiben wir n als $2m + 1$. Dann ist

$$\begin{aligned} n^2 &= (2m+1)^2 \\ &= 4m^2 + 4m + 1 \\ &= 2(2m^2 + 2m) + 1, \end{aligned}$$

was eine ungerade Zahl ist. □

Widerspruchsbeweis

Beim Widerspruchsbeweis wird zunächst das Gegenteil des zu beweisenden Satzes angenommen und anschließend versucht, durch Umformungen oder Folgerungen auf einen Widerspruch zu stoßen.

Beispiel: *Die Quadratwurzel aus 2 ist irrational.*

Beweis.

Wir nehmen an, dass $\sqrt{2}$ rational ist, d. h. sie kann als Bruch geschrieben werden:

$$\begin{aligned} \sqrt{2} &= \frac{p}{q} && \text{mit } \frac{p}{q} \text{ gekürzt.} && \mid ()^2 \\ 2 &= \frac{p^2}{q^2} && \mid \cdot q^2 \\ 2q^2 &= p^2 && \Rightarrow\ p \text{ ist gerade.} \\ & && \Rightarrow\ p = 2a\ (a \in \mathbb{N}) \\ 2q^2 &= (2a)^2 \\ 2q^2 &= 4a^2 && \mid \cdot \tfrac{1}{2} \\ q^2 &= 2a^2 && \Rightarrow\ q \text{ ist gerade.} \end{aligned}$$

Dass p und q gerade sind, stellt einen Widerspruch dazu dar, dass $\frac{p}{q}$ ein vollständig gekürzter Bruch ist. Daraus folgt, dass $\sqrt{2}$ eine irrationale Zahl sein muss. □

Vollständige Induktion

Eine vollständige Induktion besteht aus drei Teilen: Dem *Induktionsanfang*, der *Induktionsannahme* und dem *Induktionsschritt*. Um diese drei Teile zu erläutern, betrachten wir eine unendlich lange Reihe Dominosteine und wollen erklären, dass rein mathematisch betrachtet, jeder Dominostein umfällt, wenn wir den ersten anstoßen und wenn wir wissen, dass jeder Dominostein seinen Nachfolger umstößt. Für den Induktionsanfang müssen wir zeigen, dass der erste Dominostein umfällt, wenn wir ihn anstoßen. Im Induktionsschritt müssen wir zeigen, dass der $(n + 1)$-te Dominostein umfällt und dürfen hierfür verwenden, dass der n-te Dominostein umfällt (Induktionsannahme). Wenn wir dies für ein beliebiges n machen, dann haben wir gezeigt, dass jeder Dominostein umfällt, also unendlich viele.

Beispiel:

$$1+2+3+\ldots+n \;=\; \sum_{i=1}^{n} i \;=\; \frac{n(n+1)}{2}.$$

Beweis.

Induktionsanfang: Hier wird gezeigt, dass die Aussage für $n = 1$ gilt. In unserem Beispiel:

$$\sum_{i=1}^{1} i = 1 \qquad \frac{1\cdot(1+1)}{2} = 1. \qquad \checkmark$$

Induktionsannahme (auch Induktionshypothese): Es wird angenommen, dass die Aussage für eine Zahl $n \in \mathbb{N}$ gilt. In unserem Beispiel:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Induktionsschritt: Hier wird gezeigt, dass die Aussage auch für $n+1$ gilt. In unserem Beispiel schreiben wir zunächst die Summe so um, dass wir das in der Induktionshypothese angenommene einsetzen können:

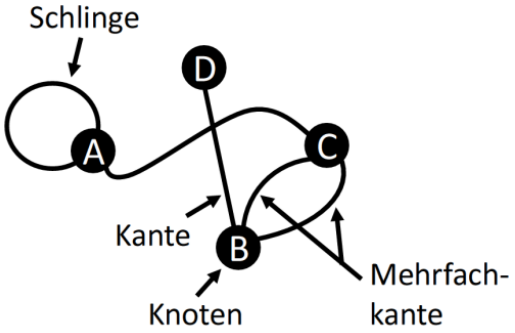
$$\begin{aligned}
\sum_{i=1}^{n+1} i \;&=\; \sum_{i=1}^{n} i + (n+1) \\
&\overset{\text{I.H.}}{=} \frac{n(n+1)}{2} + (n+1) \\
&= \frac{n^2+n+2n+2}{2} \\
&= \frac{(n+2)\cdot(n+1)}{2} \\
&= \frac{(n+1)\cdot((n+1)+1)}{2},
\end{aligned}$$

womit wir die Aussage für $n+1$ gezeigt haben. □

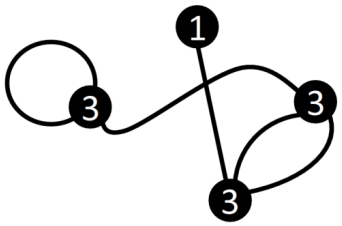
Graphen

Amelie Grimm, Jana Braunger, Johanna Buck

Ein *Graph* besteht aus *Knoten* und *Kanten*, wobei Knoten als Punkte und Kanten als Verbindungen zwischen zwei verschiedenen Knoten dargestellt werden. Zwischen zwei Knoten darf es höchstens eine Kante geben. Ein *Multigraph* darf außerdem noch *Mehrfachkanten* und/oder *Schlingen* besitzen.



Bei einem Graphen ist es egal, wo ein Knoten gezeichnet wird oder ob die Kanten durch gerade oder gebogene Linien dargestellt werden. Wichtig ist nur, welche Knoten vorhanden sind und welche Knotenpaare durch eine Kante verbunden sind. In einem Graphen wird normalerweise die Menge der Knoten mit V und die Menge der Kanten mit E bezeichnet. Man schreibt $G = (V, E)$ für den Graphen G mit der Knotenmenge V und der Kantenmenge E. Wenn zwei Knoten x und y durch eine Kante verbunden sind, dann sagt man, dass x und y *benachbart* sind, x ein *Nachbar* von y ist und umgekehrt. Außerdem schreibt man xy für diese Kante.



Die Zahlen in den Knoten stellen den Grad des jeweiligen Knotens dar.

Der *Grad* eines Knotens x bezeichnet die Anzahl der Nachbarn von x. Man schreibt dafür $\deg(x)$. Durch Zählen der Kanten, die aus x

herausgehen, kann man den Grad des Knotens x bestimmen. Der erste Satz, den wir für Graphen bewiesen haben, lautet:

Satz:
In jedem Graphen mit ungerader Knotenanzahl ist die Anzahl der Knoten mit geradem Grad ungerade.

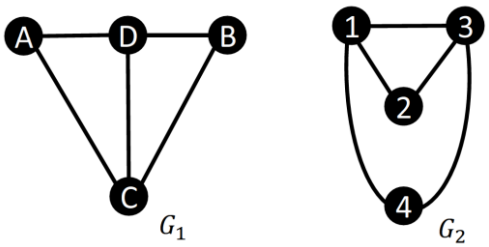
Subgraphen (Teilgraphen)

Ein beliebiger Graph H heißt *Subgraph* eines Graphen G, wenn alle Kanten von H auch Kanten von G sind und jeder Knoten von H auch Knoten von G ist. Man schreibt dann $H \subseteq G$.

Gewichtete Graphen

In einem *gewichteten Graphen* wird jeder Kante ein positiver Wert zugewiesen, wie beispielsweise die Entfernung zwischen zwei Orten oder der Ticketpreis einer Bahnverbindung. Diesen Wert $w(x, y)$ nennt man das *Gewicht* der Kante xy. Die Summe der Gewichte aller Kanten eines Graphen G wird mit $w(G)$ bezeichnet.

Isomorphe Graphen



Zwei isomorphe Graphen.

Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ heißen *isomorph*, wenn es eine Abbildung zwischen den Knotenmengen V_1 und V_2 gibt, die folgende Eigenschaften erfüllt:

- Die Abbildung ist eine *eins-zu-eins* Abbildung von V_1 nach V_2. Das heißt, jedem Knoten aus der Knotenmenge V_1 wird genau ein Knoten aus der Knotenmenge V_2 zugeordnet und jedem Knoten aus V_2 wird genau ein Knoten aus V_1 zugeordnet.
- Zwei Knoten x und y von G_1 sind genau dann benachbart, wenn die zugehörigen Knoten in G_2 benachbart sind.

Dabei können die Knoten andere Namen haben oder bei Zeichnungen anders angeordnet sein, doch zu jedem Knoten im ersten Graphen muss es genau einen Knoten im zweiten, isomorphen Graphen geben, der diesem entspricht und dieselben entsprechenden Nachbarn besitzt.

G_1	A	B	C	D
G_2	4	2	1	3

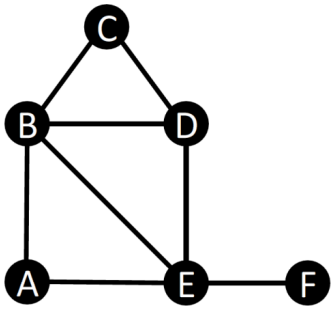
Die Tabelle zeigt eine Abbildung, die die Knoten von G_1 den Knoten von G_2 zuordnet und die geforderten Eigenschaften erfüllt. Beispielsweise entspricht der Knoten A aus Graph G_1 dem Knoten 4 aus Graph G_2: Sie haben die Nachbarn C und D, bzw. 1 und 3. Der Knoten D entspricht dem Knoten 3 und C dem Knoten 1. Ähnliches gilt für die restlichen Knoten: Zu jedem Knoten in G_1 gibt es einen entsprechenden Knoten in G_2 und die entsprechenden dazugehörigen Kanten. Wie die beiden Graphen dabei gezeichnet werden, ist egal.

Kantenzüge, Pfade und Kreise

Ein *Kantenzug* ist eine Abfolge von Knoten in einem Graph, wobei aufeinander folgende Knoten benachbart sind. Zum Beispiel ist

$$(A, E, B, C, D, E, F)$$

ein Kantenzug im nächsten Graphen. Die *Länge* des Kantenzugs ist die Anzahl der verwendeten Kanten. Beispielsweise hat der obige Kantenzug die Länge 6.



Ein *Pfad* ist ein Kantenzug mit der zusätzlichen Einschränkung, dass kein Knoten doppelt

verwendet werden darf. Zum Beispiel ist

$$(A, B, C, D, E, F)$$

ein Pfad. Dabei ist A der Start- und F der Endknoten des Pfads, der somit als A, F-Pfad bezeichnet wird. In unserem Beispiel gibt es keinen Pfad, der alle Kanten verwendet, weil Knoten B Grad 4 hat. Deshalb müsste der Pfad den Knoten B zweimal verwenden, um alle vier Kanten bei B zu benutzen.

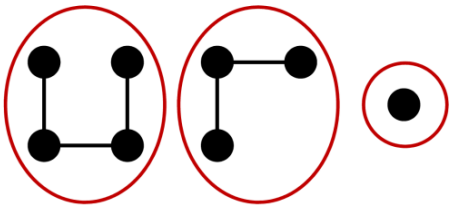
Ein *Kreis* ist ein Pfad auf mindestens 3 Knoten mit einer zusätzlichen Kante vom letzten Knoten des Pfades zum Startknoten. Diese Kante ist bei der Länge des Kreises mitzuzählen. Zum Beispiel ist

$$(A, B, C, D, E)$$

ein Kreis der Länge 5.

Zusammenhängende Graphen

Ein Graph $G = (V, E)$ heißt *zusammenhängend*, wenn es für alle Knoten $x, y \in V$ einen x, y-Pfad – also einen Pfad, der in Knoten x anfängt und in Knoten y aufhört – in G gibt. Das heißt, man kommt von jedem Knoten mit einem Pfad zu jedem anderen Knoten.



Dieser Graph besteht aus den drei rot umrandeten Zusammenhangskomponenten.

Ein Subgraph $H \subseteq G$ heißt *Zusammenhangskomponente*, wenn H zusammenhängend ist und kein Subgraph H' existiert, sodass H' zusammenhängend ist, $H \subseteq H'$ und $H' \neq H$. Das heißt, wenn ein Graph G nicht zusammenhängend ist, dann besteht er aus mehreren Zusammenhangskomponenten. Die einzelnen Zusammenhangskomponenten sind zusammenhängend und ergeben den Graph G.

Eulersche Graphen

Gözde Kabadayi, Karen Wiegand

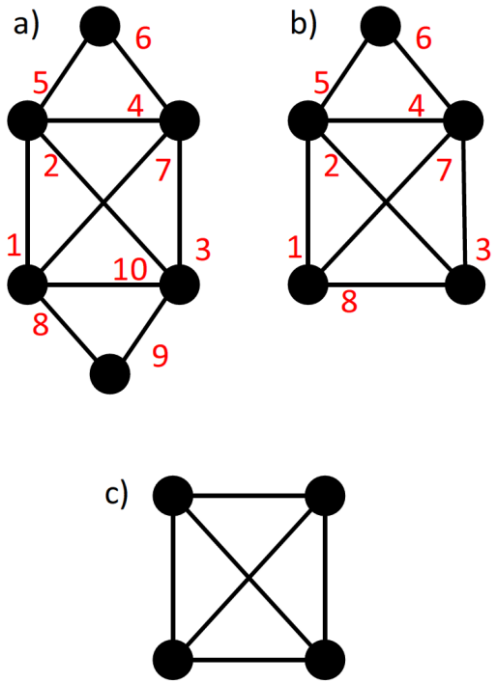
Ein *Euler-Kantenzug* in einem Graphen G ist ein Kantenzug $(v_1, v_2, \ldots, v_m)$, der jede Kante von G genau einmal verwendet. Falls dabei $v_1 = v_m$ ist, wird dieser als *Eulertour* bezeichnet. Ein Graph G heißt *eulersch*, wenn er eine Eulertour besitzt, bzw. *semi-eulersch*, wenn er keine Eulertour, aber einen Euler-Kantenzug besitzt.

Ob ein Graph eulersch, semi-eulersch oder keines von beidem ist, kann man herausfinden, indem man versucht, den Graphen nachzuzeichnen, ohne den Stift abzusetzen. Dabei darf keine Kante doppelt verwendet werden.

- Wenn ein Graph in einem Zug nachgezeichnet werden kann und dabei Start- und Endknoten derselbe sind, ist der Graph eulersch.
- Wenn ein Graph in einem Zug nachgezeichnet werden kann, jedoch Start- und Endknoten nicht derselbe sind, ist der Graph semi-eulersch.
- Wenn der Graph nicht in einem Zug nachgezeichnet werden kann, ist der Graph weder eulersch noch semi-eulersch.

In unserem Kurs suchten wir nach einer besseren Herangehensweise, um herauszufinden, ob ein Graph eulersch, semi-eulersch oder keines von beidem ist. Hierfür haben wir uns den Satz von Euler angesehen und bewiesen.

Satz (Satz von Euler)**:**
Ein zusammenhängender (Multi-)Graph ist genau dann eulersch, wenn jeder Knoten von G geraden Grad hat.



Läuft man die Kanten in der Reihenfolge der roten Zahlen ab, erhält man a) eine Eulertour und b) einen Euler-Kantenzug. Außerdem zeigt c) einen Graphen, der weder eulersch noch semi-eulersch ist.

Hierbei handelt es sich um eine Äquivalenzaussage. Einerseits gibt es immer eine Eulertour in einem zusammenhängenden Graphen G, wenn jeder Knoten von G geraden Grad hat. Andererseits ist der Grad jedes Knoten in einem eulerschen Graphen stets gerade. Um den Satz von Euler zu beweisen, müssen wir also zwei Richtungen zeigen.

Richtung 1: „Wenn G eulersch ist, dann hat jeder Knoten geraden Grad."

Wenn G eulersch ist, dann gibt es einen Kantenzug, der jede Kante genau einmal verwendet und dessen Start- und Zielknoten gleich sind.
$\Rightarrow$ Wenn wir diesen Kantenzug ablaufen, verwenden wir genau zwei Kanten eines Knotens x bei jedem Durchlaufen von x.
$\Rightarrow$ Jeder Knoten in G hat geraden Grad.

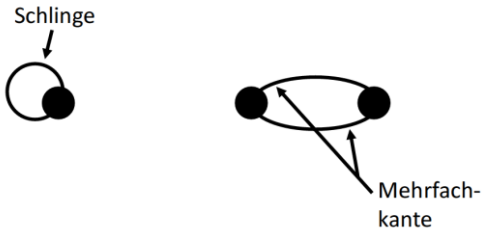
Richtung 2: „Wenn jeder Knoten geraden Grad hat, dann ist G eulersch."

Für den Beweis dieser Richtung haben wir einen mathematischen Hilfssatz verwendet:

Lemma:
Jeder (Multi-)Graph, bei dem jeder Knoten mindestens Grad 2 hat, enthält einen Kreis.

Beweis.

Fall 1: Der Graph enthält eine Mehrfachkante oder eine Schlinge.



$\rightarrow$ Kreis bereits vorhanden.

Fall 2: Der Graph enthält keine Mehrfachkante und keine Schlinge.

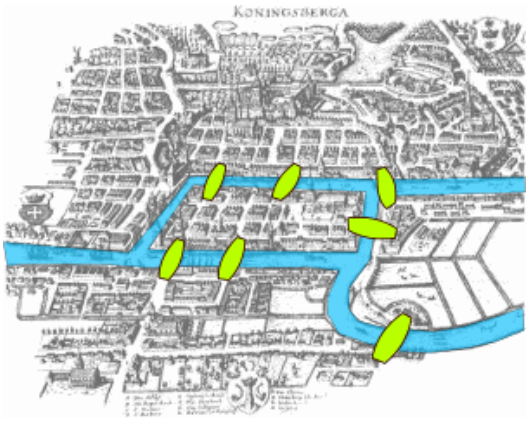
Von einem Startknoten v_1 wird ein Nachbar v_2 gesucht. Der Knoten v_2 hat einen weiteren Nachbarn v_3, da er mindestens Grad 2 hat und G keine Mehrfachkanten enthält. Dann hat der Knoten v_3 entweder v_1 als weiteren Nachbarn ($\rightarrow$ Kreis) oder einen weiteren Nachbarn v_4 und so weiter. Da G nur endlich viele Knoten enthält, muss schließlich ein Knoten v_i mit einem vorherigen Knoten einen Kreis schließen. □

Durch vollständige Induktion über die Anzahl der Kanten des Graphen konnten wir zeigen, dass jeder zusammenhängende Graph, dessen Knoten alle geraden Grad haben, eulersch ist.

Im Anschluss folgerten wir, dass ein zusammenhängender Graph G genau dann semi-eulersch ist, wenn er genau zwei Knoten mit ungeradem Grad besitzt. Dabei haben wir verwendet, dass jeder Graph eine gerade Anzahl an Knoten mit ungeradem Grad hat. Außerdem folgerten wir, dass ein zusammenhängender Graph, der weder eulersch noch semi-eulersch ist, mindestens vier Knoten mit ungeradem Grad haben muss.

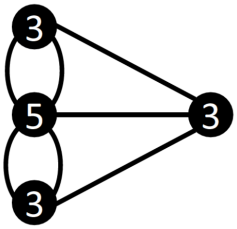
Das Königsberger-Brücken-Problem

Die Menschen aus Königsberg haben sich im 18. Jahrhundert gefragt, ob es möglich ist, über jede der sieben Brücken genau einmal zu gehen und am Ende wieder dort anzukommen, wo sie gestartet sind. Diese Frage konnten wir mithilfe unseres Wissens über eulersche Graphen beantworten.



Die Brücken in Königsberg im 18. Jahrhundert.[1]

Das Königsberger-Brücken-Problem lässt sich durch folgenden Multigraphen darstellen, der als unser Kurslogo auf dem Deckblatt dieser Dokumentation zu sehen ist.



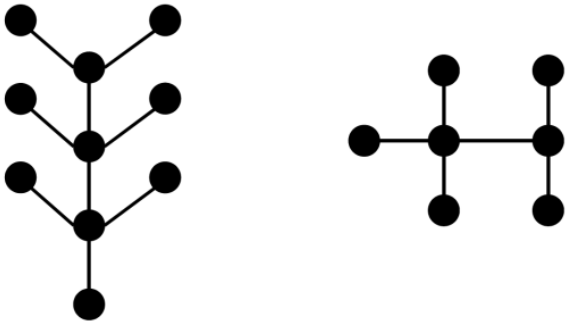
Hierbei stehen die Knoten für die Stadtteile und die Kanten für die Verbindungen durch eine Brücke. Im obigen Multigraphen haben wir die Grade der Knoten bestimmt und in die Knoten eingetragen. Man stellt fest, dass alle vier Knoten ungeraden Grad haben. Der Multigraph ist also nicht eulersch. Das bedeutet für die Bürger von Königsberg, dass es nicht möglich ist, über alle Brücken genau einmal zu gehen und anschließend wieder beim Startpunkt anzukommen.

Bäume

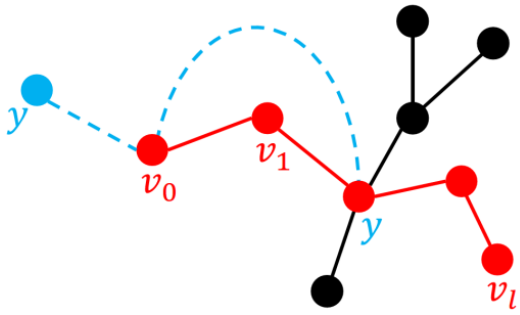
Amelie Grimm, Johanna Kaps

Bäume finden in der Graphentheorie vielseitige Anwendungen, um beispielsweise Schienen- oder Wegnetze zu optimieren. Ein Baum ist ein kreisfreier und zusammenhängender Graph. Ein kreisfreier Graph wird auch *Wald* genannt. Ein *Blatt* ist ein Knoten mit Grad 1.

[1] Abb.: Wikimedia (Bogdan Giuşcă, CC BY-SA 3.0)



Satz:
Jeder Baum T auf mindestens zwei Knoten hat mindestens zwei Blätter.



Beweis.

Sei T ein Baum und $P = (v_0, v_1, ..., v_\ell)$ ein längster Pfad in T. Wir nehmen an, dass v_0 kein Blatt ist. Da T zusammenhängend ist, muss $\deg(v_0) \geq 2$ gelten. Der Knoten v_0 hat demnach noch einen weiteren Nachbar $y \neq v_1$. Wenn y auf dem Pfad P liegt, dann hat T einen Kreis, was im Widerspruch dazu steht, dass T ein Baum ist. Demnach liegt y nicht auf P, was wiederum im Widerspruch dazu steht, dass P ein möglichst langer Pfad in T ist. Folglich ist v_0 ein Blatt. Analog beweist man, dass v_ℓ ebenfalls ein Blatt ist. Somit besitzt der Baum T die beiden Blätter v_0 und v_ℓ. □

Des Weiteren haben wir im Sommer gezeigt, dass Bäume eine Reihe von Eigenschaften haben, die mit oben genannter Definition äquivalent sind:

Satz (Charakterisierung von Bäumen)**:**
Folgende Aussagen sind für einen Graphen T auf n Knoten äquivalent:

(i) T ist ein Baum.

(ii) *T ist zusammenhängend und besitzt genau $n-1$ Kanten.*

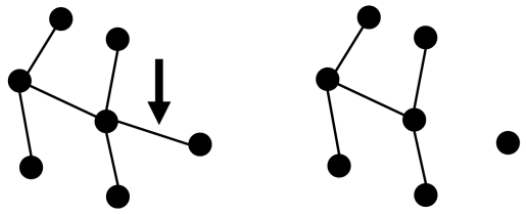
(iii) *T ist kreisfrei und hat genau $n-1$ Kanten.*

(iv) *Wenn x und y zwei beliebige Knoten von T sind, dann gibt es genau einen x,y-Pfad in T.*

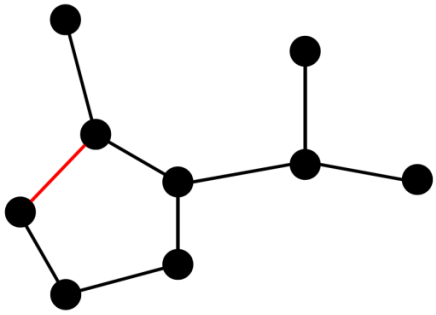
(v) *T ist kantenminimal zusammenhängend.*

(vi) *T ist kantenmaximal kreisfrei.*

Hierbei bedeutet *kantenminimal zusammenhängend*, dass der Graph T zusammenhängend ist, aber das Entfernen einer beliebigen Kante diesen Zusammenhang zerstören würde. Analog bedeutet *kantenmaximal kreisfrei*, dass T kreisfrei ist, aber das Hinzufügen einer beliebigen Kante einen Kreis schließt.



Wenn man die markierte Kante aus dem Baum entfernt, erhält man einen Graphen, der nicht zusammenhängend ist.



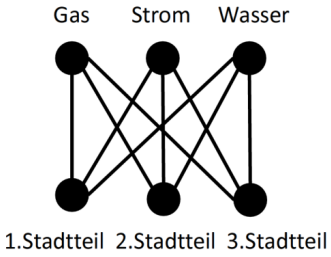
Wenn man die rote Kante in den Baum einfügt, entsteht ein Kreis.

Obiger Satz besagt, dass die sechs dort aufgelisteten Eigenschaften äquivalent sind. Wenn wir jede der geltenden Richtungen einzeln zeigen würden, wären das 30 Richtungen. Durch geschicktes Vorgehen konnten wir diese Anzahl deutlich reduzieren. Zum Beispiel haben wir am Anfang $(i) \Rightarrow (ii)$, $(ii) \Rightarrow (iii)$ und $(iii) \Rightarrow (i)$ bewiesen. Daraus folgt auch, dass $(i) \Rightarrow (iii)$, $(ii) \Rightarrow (i)$ und $(iii) \Rightarrow (ii)$ gelten. Außerdem konnten wir in den weiteren Beweisteilen verwenden, dass die Eigenschaften (i), (ii) und (iii) äquivalent sind. Die einzelnen Beweisteile haben wir im Kurs in Kleingruppen erarbeitet und anschließend dem gesamten Kurs vorgestellt.

Planare Graphen

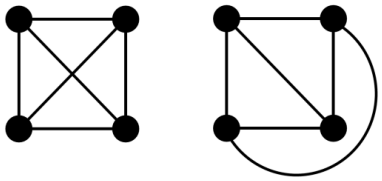
Christian Hirsch, David Ringhut

Wir beginnen zum Einstieg in die Thematik der *planaren Graphen* mit folgendem Beispiel: Die drei Stadtteile einer Stadt brauchen jeweils eine Leitung des städtischen Strom-, Wasser- und Gasversorgers. Aus Sicherheitsgründen dürfen sich, von oben betrachtet, keine der Leitungen kreuzen. Ist es möglich, alle drei Stadtteile unter dieser Bedingung zu versorgen? Um dieses Problem mithilfe der Graphentheorie zu untersuchen, verwenden wir folgenden Graphen:

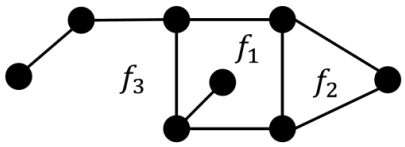


Die Aufgabe besteht also darin, diesen Graphen so zu zeichnen, dass sich keine seiner Kanten kreuzen. Ein Graph heißt *planar*, wenn er so in die Ebene gezeichnet werden kann, dass sich keine Kanten überschneiden. Eine solche Zeichnung, bei der sich keine Kanten überschneiden, wird auch als *ebene Zeichnung* des Graphen bezeichnet. Im folgenden Beispiel betrachten wir zwei Darstellungen desselben Graphen G.

Die linke Zeichnung ist nicht eben, da sich zwei Kanten überkreuzen. Die rechte Zeichnung ist



eben und daher ist G planar. In einer ebenen Zeichnung unterteilen die Kanten die Ebene in *Gebiete*. Das unbeschränkte Gebiet (im folgenden Beispiel f_3) wird *äußeres Gebiet* genannt. Die Menge der Gebiete wird mit R bezeichnet und $G = (V, E, R)$ beschreibt eine ebene Zeichnung des Graphen G mit den Gebieten R.



Über ebene Zeichnungen lässt sich folgender, zentraler Satz der Graphentheorie beweisen, was wir im Sommer gemacht haben.

Satz (Eulerformel)**:**
Für jede ebene Zeichnung $G = (V, E, R)$ eines zusammenhängenden Graphen gilt

$$|V| - |E| + |R| = 2.$$

Wenn man nun beispielsweise die 8 Knoten, 9 Kanten und 3 Gebiete des obenstehenden Graphen in die Formel einsetzt, erhält man die wahre Aussage $8 - 9 + 3 = 2$.

Die Eulerformel haben wir mithilfe einer vollständigen Induktion über die Anzahl der Kanten des Graphen bewiesen. Die Idee dabei besteht darin, dass man im Induktionsschritt die beiden Fälle, dass G ein Baum ist oder nicht, betrachtet.

Aus der Eulerformel lässt sich folgern, dass ein planarer Graph auf $n \geq 3$ Knoten höchstens $3n-6$ Kanten haben kann. Diese Aussage lässt sich wie folgt noch verallgemeinern.

Satz:
Jeder planare Graph auf $n \geq 3$ Knoten, in dem jeder Kreis mindestens die Länge r hat, kann höchstens

$$\frac{r}{r-2} \cdot (n-2)$$

Kanten haben.

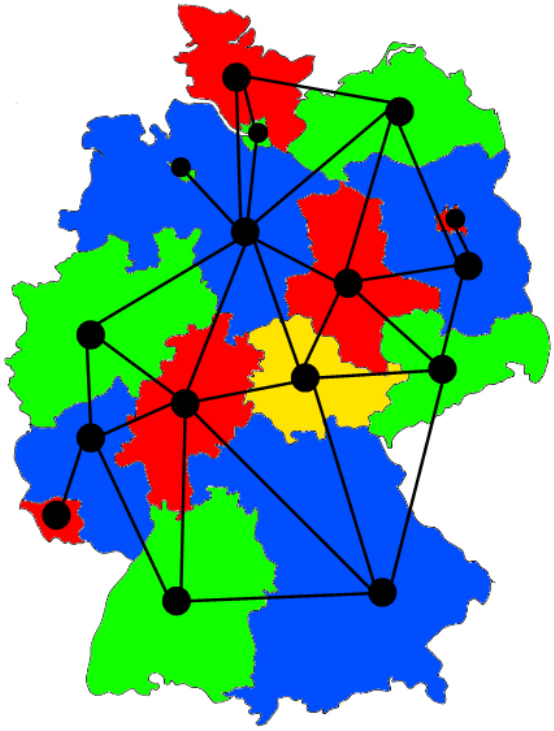
Kommen wir nun zurück zum Problem der städtischen Versorgungsleitungen und dem zugehörigen Graphen G. Alle Kreise von G haben mindestens Länge 4, weshalb wir obigen Satz mit $r = 4$ anwenden können. Daraus folgt, dass jeder planare Graph auf 6 Knoten, der keinen Kreis der Länge < 4 hat, höchstens 8 Kanten haben kann. Da der Graph aber 9 Kanten hat, kann er folglich nicht planar sein. Übertragen auf das Problem ist es also nicht möglich, die Leitungen nach der Sicherheitsvorschrift ohne Überkreuzungen zu verlegen.

Landkarten einfärben

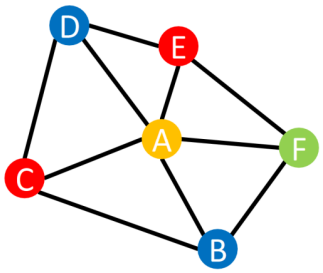
Benjamin Kloss, Johanna Kaps

Des Weiteren haben wir uns mit der Frage beschäftigt, wie viele Farben man braucht, um eine beliebige Landkarte so einzufärben, dass benachbarte Länder stets verschiedene Farben haben. Als erstes haben wir das bei einer Deutschlandkarte ausprobiert. Hier lag die Aufgabe darin, alle Bundesländer so zu färben, dass keine benachbarten Bundesländer die gleiche Farbe haben. Dies haben wir durch Ausprobieren mit vier Farben geschafft, was uns noch nicht gereicht hat. Wir wollten auch mithilfe der Graphentheorie beweisen, dass man mindestens vier Farben braucht, um die Deutschlandkarte auf diese Weise zu färben. Dazu haben wir jedes Bundesland mit einem Knoten dargestellt. Wenn zwei Bundesländer nebeneinander liegen, also benachbart sind, haben wir dort eine Kante in den Graphen gezeichnet. Dadurch erhält man, auch bei einer beliebigen Landkarte mit zusammenhängenden Ländern, einen planaren Graphen.

Man bezeichnet einen Graphen $G = (V, E)$ als *k-färbbar*, wenn die Knoten von G mit höchstens k Farben gefärbt werden können, sodass für alle Kanten $xy \in E$ gilt, dass die Knoten x und y verschieden gefärbt sind. Beispielsweise ist der zur Deutschlandkarte gehörige Graph 4-färbbar, da man mit vier Farben die Deutschlandkarte so färben kann, dass keine benachbarten Bundesländer die gleiche Farbe haben und dementsprechend auch die Knoten des zugehörigen Graphens. Um zu beweisen, dass man bei der Deutschlandkarte mindestens vier



Die mit vier Farben eingefärbte Deutschlandkarte und der zugehörige Graph.



Am Beispiel Thüringens sieht man, dass man für die Deutschlandkarte mindestens vier Farben benötigt.

Farben benötigt, betrachten wir Thüringen und seine Nachbarbundesländer näher. Wir färben zu Beginn Thüringen in gelb, und somit darf keines der Nachbarbundesländer ebenfalls gelb gefärbt werden. Um den Knoten A (Thüringen) herum existiert der Kreis (B, C, D, E, F), der eine ungerade Länge hat. Hätte er eine gerade Länge, dann könnte man ihn mit zwei Farben färben, beispielsweise blau, rot, blau, rot, usw. Das geht bei Kreisen mit ungerader Länge nicht. Wenn wir zum Beispiel B, C, D und E abwechselnd blau und rot färben, dann ist der Knoten F sowohl mit einem blauen, als auch mit einem roten Knoten, benachbart. Deshalb muss der Knoten F mit einer vierten Farbe gefärbt werden. In unserem Fall haben wir die Farbe grün gewählt.

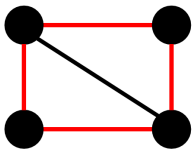
Der sogenannte *4-Farben-Satz* sagt aus, dass jeder planare Graph 4-färbbar ist. Daraus folgt, dass jede beliebige Landkarte mit vier Farben so gefärbt werden kann, dass keine benachbarten Länder die gleiche Farbe haben. Dieser Satz konnte aber bisher nur mithilfe eines Computers bewiesen werden, da sehr viele einzelne Fälle betrachtet werden müssen. Wenn man zeigen möchte, dass für jede Landkarte solch eine Färbung mit fünf Farben existiert, dann wird der Beweis viel einfacher und kürzer. Diesen Beweis haben wir im Sommer ausführlich besprochen. Für nicht planare Graphen gilt dieser Satz allerdings nicht, da beispielsweise ein Graph auf n Knoten, bei dem jeder Knoten mit jedem anderen benachbart ist, nur mit n verschiedenen Farben gefärbt werden kann.

Hamiltonsche Graphen

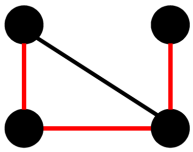
Christian Hirsch, Karen Wiegand

Ein *Hamiltonkreis* ist ein Kreis der Länge n in einem Graphen auf n Knoten. Das heißt ein Hamiltonkreis ist ein geschlossener Kantenzug, der jeden Knoten des Graphen genau einmal verwendet, was bedeutet, dass keine Kante mehrfach verwendet werden kann. Im Gegensatz zu Eulertouren müssen dabei nicht alle Kanten durchlaufen werden. Ob ein Hamiltonkreis vorhanden ist oder nicht, ist vor allem bei praktischer Anwendung der Graphentheorie interessant. So kann daraus z. B. geschlossen werden, ob eine Rundtour zwischen allen Zielen ohne doppelte Wege möglich ist.

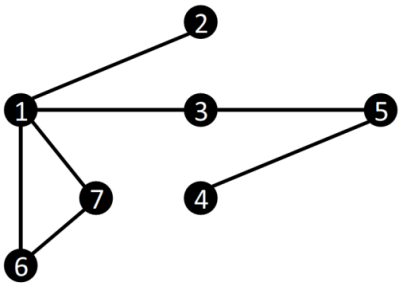
Ein Graph, der einen Hamiltonkreis besitzt, wird als *hamiltonscher Graph* bezeichnet.



Analog zu semi-eulerschen Graphen wird ein Graph als *semi-hamiltonsch* bezeichnet, wenn er keinen Kreis der Länge n, aber einen Pfad der Länge $n-1$ besitzt.



Der folgende Graph ist ein Beispiel für einen Graphen, der weder hamiltonsch noch semi-hamiltonsch ist, da es nicht möglich ist, jeden Knoten zu durchlaufen, ohne beispielsweise Knoten 1 doppelt zu verwenden.



Die Charakterisierung hamiltonscher Graphen ist noch heute ein ungelöstes Problem in der Graphentheorie. Es sind lediglich Theoreme bekannt, die entweder notwendige oder hinreichende Bedingungen für das Vorhandensein eines Hamiltonkreises liefern. Eine einfache notwendige Bedingung, die aus dem letzten Beispiel ersichtlich ist, lautet: Ein Graph, der einen Knoten von Grad 1 hat, ist nicht hamiltonsch.

Eine hinreichende Bedingung liefert der *Satz von Ore*, den wir uns im Kurs angeschaut haben.

Satz (Satz von Ore)**:**
Sei $G = (V, E)$ ein Graph auf $n \geq 3$ Knoten und $x, y \in V$ zwei beliebige, verschiedene, nicht benachbarte Knoten, die $\deg(x) + \deg(y) \geq n$ *erfüllen. Sei G' der Graph, der entsteht, wenn man die Kante xy zu G hinzufügt. Dann hat G genau dann einen Hamiltonkreis, wenn auch G' einen Hamiltonkreis hat.*

Dirac verfasste 1952 einen Satz, den man aus dem Satz von Ore folgern kann.

Satz (Satz von Dirac)**:**
Sei $G = (V, E)$ ein Graph auf $n \geq 3$ Knoten. Wenn für jeden Knoten x gilt, dass $\deg(x) \geq \frac{n}{2}$*, dann hat G einen Hamiltonkreis.*

Der Satz von Dirac besagt, dass ein Graph „mit vielen Kanten“ einen Hamiltonkreis besitzen muss. Dies ist aber lediglich eine hinreichende Bedingung, die nicht notwendigerweise erfüllt sein muss. So ist ein Kreis auf 100 Knoten natürlich hamiltonsch, erfüllt die Bedingung des Satzes jedoch nicht, da jeder Knoten nur Grad 2 hat.

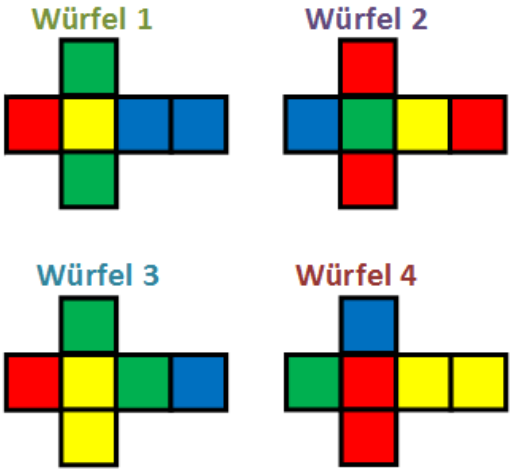
4-Würfel-Problem

Jana Braunger, Julia Högerle

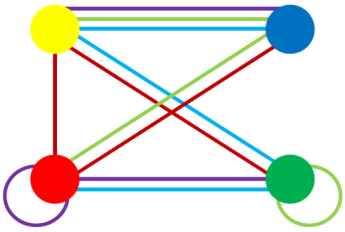
Wir haben unsere theoretischen Ergebnisse natürlich auch auf praktische Probleme angewandt, wie beispielsweise bei dem *4-Würfel-Problem*. Die Aufgabe besteht darin, vier Holzwürfel mit verschiedenfarbigen Außenflächen (rot, grün, gelb und blau) so aufeinander zu einem Turm zu stapeln, dass man auf allen vier Seiten (vorne, hinten, rechts und links) jede Farbe genau einmal sieht.

Beispielsweise könnten die Würfelnetze dabei so aussehen wie in der Abbildung auf der nächsten Seite.

Natürlich kann man diese Aufgabe durch Ausprobieren lösen. Einfacher ist es jedoch mithilfe eines Multigraphen: Dabei entspricht jede Farbe einem Knoten im Multigraph G. Dieser Multigraph hat also vier Knoten. Für jeden Würfel zeichnet man drei Kanten in den Multigraph, wobei jede Kante die auf dem Würfel gegenüberliegenden Farben verbindet. Dabei müssen wir uns merken, welche Kante zu welchem Wür-



fel gehört. Hier färben wir zum Beispiel die zu Würfel 1 gehörigen Kanten in grün. Zeichnet man diesen Multigraph, würde er in unserem Beispiel folgendermaßen aussehen:



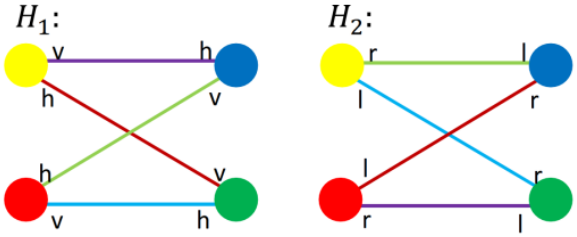
Um nun mithilfe des Multigraphen das Problem lösen zu können, muss man zwei Subgraphen H_1 und H_2 des Multigraphen G finden. Wie vorher erläutert, ist ein Subgraph ein Teilgraph aus einem Graphen G. Der Subgraph H_1 beschreibt dabei, welche Farben der Würfel man sieht, wenn man von vorne und hinten den Turm anschaut. Die Sicht auf die Würfel im Turm von rechts und links wird von H_2 beschrieben. Für die beiden Subgraphen H_1 und H_2 muss gelten:

1.) H_1 und H_2 besitzen von jedem Würfel genau eine Kante, da alle vier Würfel verwendet werden müssen, aber keiner doppelt benutzt werden kann.

2.) H_1 und H_2 dürfen keine gemeinsamen Kanten besitzen: Würde man eine Kante in beiden Subgraphen verwenden, müssten die beiden durch die Kante verbundenen, gegenüberliegenden Farben gleichzeitig vorne/hinten und rechts/links liegen, was natürlich nicht möglich ist.

3.) Jeder Knoten in den Subgraphen hat den Grad zwei, denn im Subgraphen H_1, der die Ansicht von vorne/hinten beschreibt, muss jede Farbe genau einmal vorne und einmal hinten auftauchen, also insgesamt zweimal. Ähnliches gilt für den Subgraphen H_2.

In unserem Beispiel können wir folgende Subgraphen H_1 und H_2 verwenden, um den Turm aufzubauen:



Nun markiert man im Subgraph H_1 ein Ende von jeder Kante mit einem „v“ (vorne) und das andere Ende mit einem „h“ (hinten), sodass an jedem Knoten (d. h. an jeder Farbe), einmal vorne und einmal hinten steht. Mit Subgraph H_2 wird dann mit rechts und links genauso verfahren. Um den Turm zu bauen, muss man nun die Informationen aus beiden Subgraphen ablesen: Zum Beispiel müsste man den ersten Würfel so aufstellen, dass vorne blau, hinten rot, rechts gelb und links blau ist. Stellt man nun die restlichen Würfel nach demselben Prinzip auf, hat man am Ende den fertigen Turm, bei dem man auf jeder Seite jede Farbe genau einmal sieht.

In diesem Beispiel gibt es für das Problem nur eine Lösung, da man keine anderen zwei Subgraphen von G mit den entsprechenden Eigenschaften finden kann. Es kann aber bei anderen Färbungen der Würfel durchaus auch mehrere zusammenpassende Subgraphen, also mehrere Lösungen der Aufgabe, geben.

Programmieren mit Java

Christian Hirsch, David Ringhut

Viele Verfahren, auch *Algorithmen* genannt, brauchen bei größeren Graphen sehr viel Zeit, wenn man sie von Hand ausführt. Hier kommt dann die Informatik ins Spiel: Eine Vielzahl der Eigenschaften von Graphen lassen sich mit

entsprechenden Computerprogrammen in Sekundenschnelle überprüfen und ausgeben. Kein Wunder also, dass das Programmieren in unserem Kurs ein nicht zu vernachlässigender Bestandteil war.

BlueJ ist eine Entwicklungsumgebung zum Erlernen der von uns genutzten Programmiersprache Java. Dabei unterstützt es den Programmierer mit zahlreichen Hilfestellungen. So markiert BlueJ beispielsweise den Quellcode auf sinnvolle Weise farbig und rückt bei Schleifen und Funktionen die Zeilen automatisch ein, um das Programm lesbarer zu gestalten.

Neben vielen vordefinierten Funktionen gehören die *Variablen* zu den wichtigsten Bestandteilen unserer Programme. Variablen ermöglichen das Speichern von Werten, die im Programm ermittelt werden. Dabei gibt es verschiedene *Typen* von Variablen, zum Beispiel Ganzzahl-, Kommazahl- und Textvariablen.

Hier eine Übersicht der wichtigsten Variablentypen:

Variablentyp	Bedeutung
`int`	ganze Zahl
`double`	Fließkommazahl
`String`	Zeichenkette
`boolean`	Wahrheitswert

Prinzipiell arbeitet der Computer einen Programmcode Zeile für Zeile von oben nach unten ab. Um diesen Programmfluss zu verändern, gibt es mehrere Möglichkeiten.

Verzweigungen

Eine *Verzweigung* oder *if-Abfrage* prüft eine Bedingung und führt, falls die Bedingung wahr ist, einen Anweisungsblock aus. Um bei Nichterfüllung der Bedingung eine andere Anweisung auszuführen, hängt man noch einen *else-Block* an:

```
if (i == 0) {
   ...
} else {
   ...
}
```

Schleifen

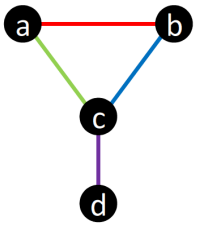
Schleifen benutzt man, wenn ein Programm denselben Befehl mehrmals wiederholen soll. Wir haben im Kurs for- und while-Schleifen kennengelernt. Die *for-Schleife* verwendet eine eigene Zählvariable. Man nutzt sie, wenn vorab bekannt ist, wie oft die Schleife durchlaufen werden soll. Ein Beispielprogramm, das die Zahlen von 1 bis 10 ausgibt:

```
for(int i = 1; i <= 10; i++) {
  System.out.println(i);
}
```

Die *while-Schleife* führt einen Block von Befehlen immer wieder aus, solange eine Bedingung erfüllt ist. Sie definiert keine eigene Zählvariable. Man verwendet sie, wenn vorab unbekannt ist, wie oft der Befehl ausgeführt werden soll, beispielsweise wenn man einen kürzesten Weg in einem Graphen sucht, ohne zu wissen, wie viele Kanten dieser Weg verwenden wird.

Adjazenzmatrizen

Mit dem Bild eines Graphen kann ein Computer nicht viel anfangen. Für ihn müssen wir den Graph in Zahlen übersetzen. Das gelingt uns mithilfe der *Adjazenzmatrix*. Die Adjazenzmatrix ist eine Art Tabelle, in der jede Zeile und jede Spalte mit einem Knoten beschriftet ist. Befindet sich in dem Feld in der Zeile i und der Spalte j eine 1, so sind die Knoten i und j benachbart; bei einer 0 sind sie nicht benachbart. Da es zu jedem Eintrag `[i][j]` den entsprechenden Eintrag `[j][i]` gibt, welcher dieselbe Kante repräsentiert, ist die Adjazenzmatrix symmetrisch aufgebaut. Im folgenden Beispiel ist ein Graph sowie die zugehörige Adjazenzmatrix dargestellt:



$$\begin{array}{c} \\ a \to \\ b \to \\ c \to \\ d \to \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \\ \downarrow & \downarrow & \downarrow & \downarrow \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Arrays

Variablen haben wir bereits oben kennengelernt. Will man mehrere Variablen desselben Typs erzeugen, kann man ein *Array* verwenden. Ein Array ist eine Art Regal mit durchnummerierten Fächern, in denen je eine Variable des gewünschten Typs gespeichert wird. Der Index eines Arrays gibt an, auf welches Fach zugegriffen wird.

Die Adjazenzmatrizen unserer Graphen haben wir mithilfe von zweidimensionalen Arrays realisiert. Diese verwenden immer zwei Indizes, um auf einen Wert zugreifen zu können. Die erste Zeile unserer Adjazenzmatrix von oben würde dann so aussehen:

```
A[0][0] = 0;
A[0][1] = 1;
A[0][2] = 1;
A[0][3] = 0;
```

Bei einem gewichteten Graphen speichert man statt einer 1 dann einfach das entsprechende Gewicht der jeweiligen Kante in der Adjazenzmatrix ab. Der Wert 0 steht weiterhin dafür, dass zwei Knoten nicht benachbart sind.

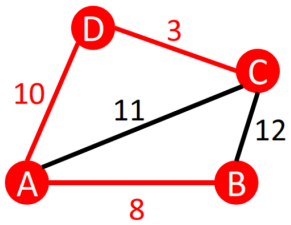
Minimale Spannbäume

Christian Martin, David Ringhut

Minimale Spannbäume finden in der Praxis Anwendung, wenn beispielsweise möglichst kurze Telefonleitungen verlegt werden sollen oder

wenn ein Eisenbahnnetz so reduziert werden soll, dass jeder Bahnhof von jedem anderen noch erreichbar ist, man aber möglichst viele Teilstrecken stilllegen kann. Klären wir zunächst den Begriff *Spannbaum*. Hierbei handelt es sich um einen Baum $T = (V', E')$, der ein Subgraph von einem Graphen $G = (V, E)$ ist und jeden Knoten aus G enthält, d. h. $V = V'$.

Ein *minimaler Spannbaum* ist ein Spannbaum $T = (V', E')$ in einem gewichteten Graphen $G = (V, E)$, sodass es keinen Spannbaum T' mit geringerem Gesamtgewicht $w(T')$ gibt.



Ein minimaler Spannbaum des Graphen G ist rot eingefärbt.

Konstruktion minimaler Spannbäume

Ein minimaler Spannbaum eines kleinen Graphen lässt sich in wenigen Schritten von Hand konstruieren. Dazu haben wir den *Algorithmus von Prim* kennengelernt, der einen minimalen Spannbaum in einem zusammenhängenden, gewichteten Graphen berechnet:

1.) Färbe alle Knoten und Kanten des Graphen $G = (V, E)$ blau.

2.) Färbe einen beliebigen Knoten $s \in V$ rot.

3.) Färbe eine Kante $xy \in E$ rot, die einen roten Knoten x und einen blauen Knoten y

verbindet, sodass $w(x, y)$ unter allen diesen Kanten minimal ist.

4.) Färbe den zugehörigen Knoten y rot.

5.) Wiederhole die Schritte 2 bis 4, bis alle Knoten rot gefärbt sind.

Der so eingefärbte rote Subgraph von G ist ein minimaler Spannbaum des Graphen G.

Implementierung

Wir haben den Algorithmus von Prim in unserem Kurs programmiert. Der schwierigste Teil ist hierbei das Finden der zu färbenden Kante. Dazu durchsucht man die gesamte Adjazenzmatrix mit zwei verschachtelten for-Schleifen:

```
for(int i = 0; i < n; i++) {
  for(int j = 0; j < i; j++) {
```

Jetzt überprüft man an der Stelle [i][j] der Adjazenzmatrix, ob hier eine Kante existiert, die einen roten mit einem blauen Knoten verbindet. Falls eine so gefundene Kante ein kleineres Gewicht als die bisher optimale Kante besitzt, wird ihre Position in zwei Variablen gespeichert. Als Vergleichswert nutzt man eine Variable „gewicht“, der vor jeder Suche einer minimalen Kante das Gesamtgewicht $w(G)$ des Graphen zugewiesen wird. Findet man nun eine Kante mit kleinerem Gewicht als das der bisher leichtesten gefundenen Kante, so wird der Variablen gewicht dieses Gewicht zugewiesen.

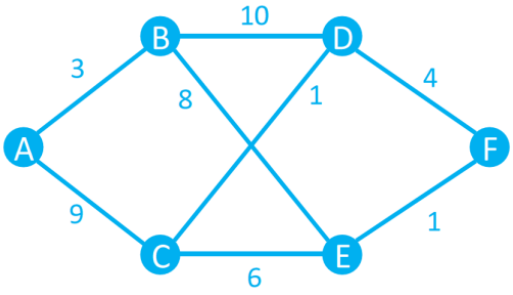
```
if(A[i][j] != 0) {
  if((Cknoten[i] == Color.blue
   && Cknoten[j] == Color.red)
   || (Cknoten[i] == Color.red
   && Cknoten[j] == Color.blue))
    {
     if(A[i][j] < gewicht) {
        e1 = i;
        e2 = j;
        gewicht = A[i][j];
    }
  }
}
```

Auf einem Graphen mit n Knoten arbeitet das Programm dabei mit drei ineinander geschachtelten for-Schleifen, die jeweils von 1 bis n zählen. Daraus ergibt sich die Komplexität $\mathcal{O}(n^3)$. Dies bedeutet, dass sich die Laufzeit unserer Implementierung verachtfacht, wenn man die Anzahl der Knoten verdoppelt. Das Verfahren von Prim lässt sich jedoch viel besser implementieren, wenn man für jeden blauen Knoten die anliegenden Kanten sinnvoll sortiert.

Algorithmus von Dijkstra

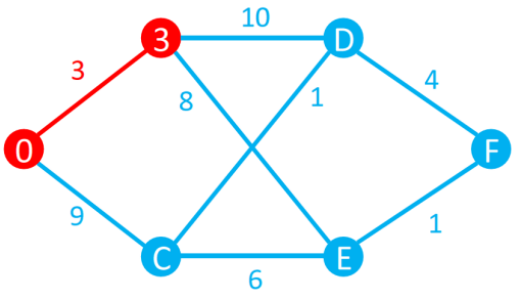
Christian Martin, Gözde Kabadayi

Das *Verfahren von Dijkstra* wird verwendet, um einen kürzesten oder schnellsten Weg zwischen zwei Orten zu finden. Dazu stellt man das Straßennetz zwischen den Orten als zusammenhängenden Graph dar.

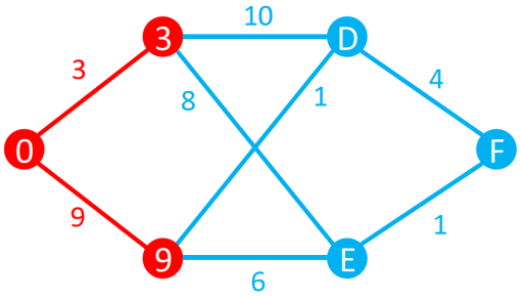


Die Knoten stellen Kreuzungen und die Kanten Straßen dar. Die Gewichte an den Kanten stehen beispielsweise für die Länge der Straße oder für die Zeit, die man zwischen den verschiedenen Kreuzungen benötigt. Wir wollen zeigen, wie man mithilfe des Algorithmus von Dijkstra einen kürzesten Weg zwischen den Knoten A und F im obigen Graphen bestimmen kann. Dabei werden wir jeweils einen kürzesten Weg von A zu allen anderen Knoten finden.

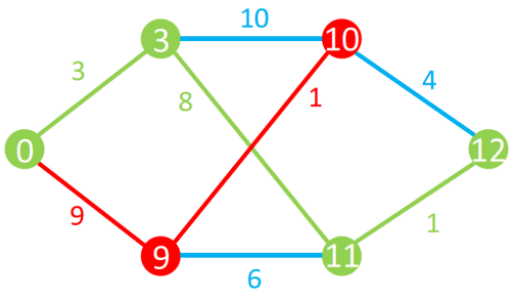
In jedem Schritt wird ein blauer Knoten rot gefärbt, bis alle Knoten rot sind und das Ziel erreicht ist. Hierfür färbt man am Anfang den Startknoten A rot. Dazu notiert man sich eine 0 in den Knoten, da man keine Zeit benötigt, um zum Startknoten zu kommen. Jetzt wird eine Kante xy gesucht, wobei x ein rot gefärbter Knoten (hier A) und y ein blau gefärbter Knoten (hier B oder C) ist. Die Kante xy muss so gewählt werden, dass die Entfernung von y zum Startknoten am kleinsten ist. Hier wählen wir also die Kante AB. Diese Kante wird ebenso wie der Knoten B rot gefärbt und das bisherige Gesamtgewicht 3, um Knoten B zu erreichen, notiert man in dem Knoten.



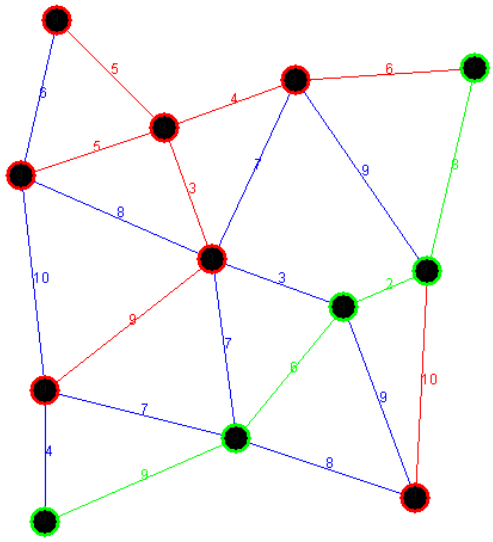
Nun wird die nächste Kante mit entsprechenden Eigenschaften gesucht. In unserem Graphen kommen jetzt drei Kanten (AC, BD und BE) in Frage. Wir erkennen, dass die Kante AC das minimale Gesamtgewicht 9 für Knoten C erzeugt. Somit wird sie, wie auch der Knoten C, rot gefärbt. Man notiert außerdem das bisherige Gesamtgewicht in den Knoten C.



Wenn man den Knoten y rot färbt, notiert man das Gesamtgewicht, um diesen Knoten zu erreichen, in y. Diese Zahl entspricht der Summe des Gewichts der Kante xy und des Gesamtgewichts von x. Jetzt sucht man die nächste Kante und wiederholt das Verfahren so lange, bis alle Knoten rot gefärbt sind. Dann führt vom Zielknoten aus nur ein rot gefärbter Pfad – ein kürzester A, F-Pfad – zum Startknoten, den wir grün färben.



Den Algorithmus von Dijkstra haben wir jetzt bei einem einfachen, kleinen Graphen gezeigt. Jedoch gibt es natürlich auch viel größere Graphen, bei denen es sehr lange dauern würde, die einzelnen Schritte von Hand durchzuführen. Deshalb haben wir diesen Algorithmus in Java programmiert. So kann unser Programm direkt die Lösung, also einen kürzesten Pfad zwischen zwei Knoten, die wir davor angegeben haben, liefern.

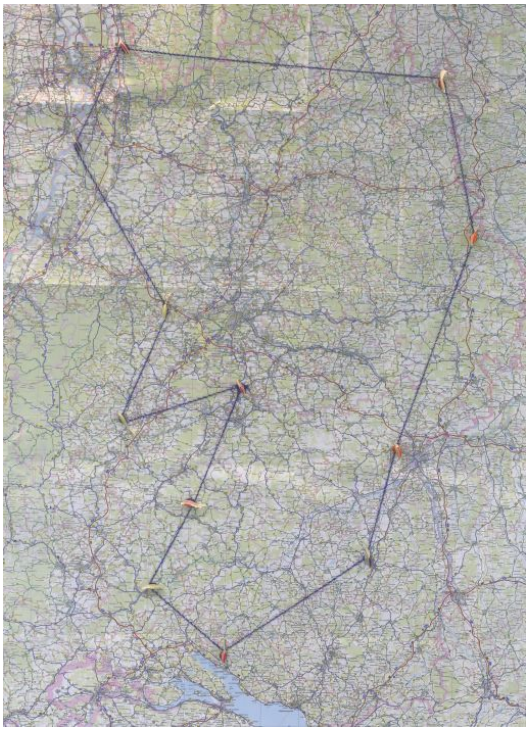


Ergebnis unseres selbst geschriebenen Dijkstra-Programms. Der gefundene kürzeste Weg ist grün gefärbt.

Das Verfahren von Dijkstra wird zum Beispiel in den Routenplanern von Navigationssystemen und Programmen wie Google Maps verwendet. Der folgende Abschnitt beschäftigt sich mit dem verwandten Problem, kürzeste Rundreisen zu finden.

Problem des Handlungsreisenden

Benjamin Kloss, Christian Martin



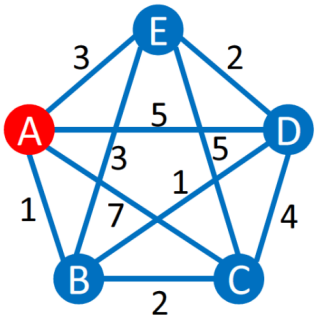
Eine kürzeste Rundreise zwischen unseren zwölf Kursteilnehmern auf einer Baden-Württemberg-Karte.

Früher standen Handlungsreisende vor dem Problem, dass sie viele Städte besuchen wollten, um dort ihre Waren zu verkaufen, aber dabei möglichst wenig Strecke zurücklegen wollten. In der Mathematik nennt man deshalb die Ermittlung einer kürzesten Rundreise zwischen mehreren Orten das *Problem des Handlungsreisenden*. Man kann das Problem durch einen gewichteten Graphen modellieren, in dem die Knoten für die Städte und die Kanten für die Verkehrsverbindungen zwischen den Städten stehen. Die Gewichte der Kanten stehen für die Entfernungen zwischen den Städten. Normalerweise kann man von jeder Stadt jede andere erreichen, sodass man einen sogenannten *vollständigen Graphen* erhält. Im Folgenden wollen wir Rundreisen in einem vollständigen, gewichteten Graphen G anschauen, d. h. wir suchen einen Hamiltonkreis in G mit kleinstmöglichem Gewicht.

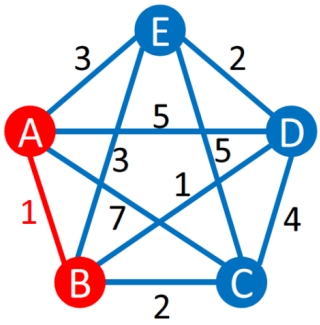
Eine Möglichkeit, eine kürzeste Rundreise zu ermitteln, ist es, alle möglichen Rundreisen im betrachteten Graphen auszuprobieren. Das einzige Problem dabei ist die Zeit, die das Verfahren braucht. Wir haben damit in unserem Kurs eine kürzeste Rundreise zwischen den Wohnorten der 12 Kursteilnehmer bestimmt, wofür der Computer 2 Minuten und 45 Sekunden benötigt hat. Das liegt daran, dass der Computer $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot \ldots \cdot 11$ Rundreisen ausprobieren muss, also etwa 40 Millionen Stück. Bei 13 Kursteilnehmern ist die Anzahl der möglichen Rundreisen 12-mal so groß, das heißt, die benötigte Rechenzeit ist ebenfalls 12-mal so groß. Für die 15 Wohnorte der Kursteilnehmer und der Kursleiter würde unser Computer mit diesem Verfahren sogar 4 Tage benötigen.

Nächste-Nachbar-Verfahren

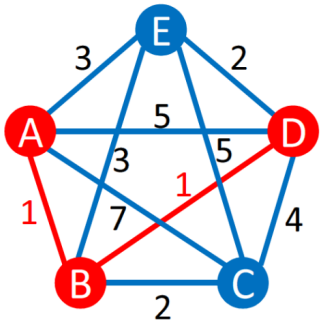
Ein anderer Lösungsansatz ist das *Nächste-Nachbar-Verfahren*. Dieses Verfahren benötigt deutlich weniger Rechenzeit als das Ausprobieren von allen Rundreisen, liefert allerdings nicht immer eine kürzeste Rundreise. Bei diesem Verfahren wird als erstes ein beliebiger Knoten, der Startknoten, rot gefärbt.



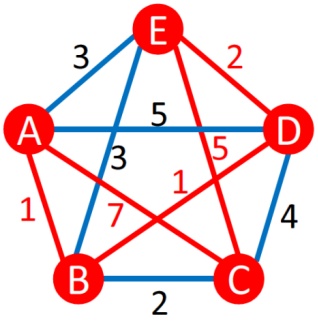
Dann wird eine Kante mit dem kleinsten Gewicht zwischen dem Startknoten und einem nicht gefärbten Knoten x, sowie der Knoten x, rot gefärbt.



Als nächstes werden eine Kante mit dem kleinsten Gewicht zwischen x und einem weiteren nicht gefärbten Knoten sowie dieser Knoten rot gefärbt.

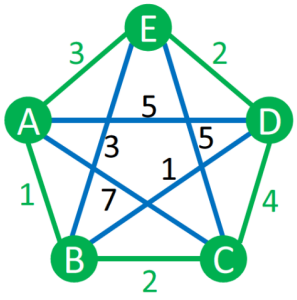


Das wiederholt man so lange, bis alle Knoten rot gefärbt sind. Wenn alle Knoten rot gefärbt sind, wird die Kante zwischen dem zuletzt gefärbten Knoten und dem Startknoten ebenfalls rot gefärbt. Die roten Kanten ergeben dann eine Rundreise.



In diesem Beispiel liefert das Nächste-Nachbar-Verfahren eine Rundreise der Länge 16. Dies ist keine kürzeste Rundreise, da die grün gefärbte Rundreise die Länge 12 hat.

Obwohl das Nächste-Nachbar-Verfahren nicht immer eine kürzeste Rundreise liefert, wird es trotzdem verwendet, weil man damit schnell eine kurze Rundreise bestimmen kann. Wie man auf andere Weisen kürzeste Rundreisen



Die grüne Rundreise ist kürzer als die mit dem Nächste-Nachbar-Verfahren bestimmte.

berechnet, hat uns unser Gast Wolfgang Riedl erklärt.

Unser Tag mit Wolfgang Riedl

Jana Braunger, Julia Högerle

Unser Gast Wolfgang Riedl.

Am Montag in der zweiten Akademiewoche fanden kursspezifische Exkursionen statt, für die unsere Kursleiter ein Treffen mit Wolfgang Riedl organisiert hatten.

Wolfgang erklärte uns ergänzend zu den behandelten Kursthemen deren Anwendungen im täglichen Leben. Im Rahmen seiner Promotion im Gebiet der Optimierung arbeitet er unter anderem an einem Projekt in München, bei dem er mit Graphentheorie zu tun hat. Um die Hintergründe dieses Projekts verstehen zu können, beschäftigten wir uns zunächst mit dem Problem des Handlungsreisenden und wie man beurteilen kann, ob eine gefundene Rundreise gut ist.

Bei dem Projekt geht es darum, die in den Außenbezirken geparkten Autos der Organisation *car to go* wieder in die Innenstadt zu fahren. Diese Route beginnt damit, dass die

Shuttle-Busse die Mitarbeiter der Organisation zu den – meist in verschiedenen Vororten Münchens geparkten – Autos fahren. Die Aufgabe der Mitarbeiter ist, die Autos an der nächstgelegenen Tankstelle zu tanken und dann wieder ins Stadtzentrum zu fahren, damit sie dort für die Kunden bereit stehen. Zur gleichen Zeit wie die Mitarbeiter soll der Shuttle-Bus im Stadtzentrum ankommen, um sie wieder in die Vororte zu bringen. Wolfgang hat uns seine Lösung zu diesem Problem gezeigt und wie sie in der Praxis angewandt wird.

Zum Schluss veranstalteten wir noch einen kleinen Wettbewerb, bei dem jeder versuchte eine möglichst kurze Rundreise durch 35 deutsche Städte zu finden. Unter dem Link http://www-m9.ma.tum.de/material/tsp-game/ kann man dieses Spiel ausprobieren. Wolfgangs Besuch war ein tolles Erlebnis, bei dem wir viel gelernt haben. Wir haben die Graphentheorie von einer ganz anderen Seite kennengelernt, was uns sehr begeistert hat.

Präsentationen

Amelie Grimm, Gözde Kabadayi

Rotation

Nach der halben Akademiezeit fand, wie üblich, der Austausch der Kurse über die jeweiligen Inhalte in Form der Rotation statt. Dazu verteilten wir die folgenden Themen gleichmäßig untereinander.

- Was sind Graphen?
- Eulersche Graphen und das Königsberger-Brücken-Problem
- Minimal aufspannende Bäume

Nach einer intensiven Übungsphase rückte der Zeitpunkt der Präsentation immer näher und die Aufregung vor unserer ersten gemeinsamen Präsentation stieg. Obwohl wir Teilnehmer uns bereits untereinander kannten, war es dennoch ein neues Erlebnis, vor so vielen Leuten zu präsentieren. Nicht nur die Teilnehmer, sondern auch andere Kursleiter kamen zu unseren Präsentationen. Später bekamen wir von unseren Kursleitern ein ausführliches Feedback, das uns half, unseren Präsentationsstil mit Hinblick auf die Abschlusspräsentation zu verbessern.

Abschlusspräsentation

Die Abschlusspräsentation am Ende der Akademie verlief ähnlich wie die Rotation. Jedoch waren diese Präsentationen für die Öffentlichkeit, unter anderem für unsere Eltern, Geschwister und ehemalige Teilnehmer, zugänglich, weshalb wir unsere Rotationspräsentation anpassten. Außerdem ergänzten wir zu unserem bisherigen Vortrag folgende Themen, da wir diese erst nach der Rotation behandelt hatten:

- Berechnung von kürzesten bzw. schnellsten Wegen
- Landkarten einfärben
- Bestimmen einer kürzesten Rundreise zwischen allen Kursteilnehmern

Während unserer Präsentation konnten wir die Verbesserungsvorschläge unserer Kursleiter direkt umsetzen. Dies führte dazu, dass auch dieser Vortrag erfolgreich gemeistert wurde und wir dadurch einen guten Eindruck beim Publikum hinterlassen konnten. Leider war mit der Abschlusspräsentation auch unsere gemeinsame Kurszeit zu Ende, weshalb wir, trotz der gelungenen Präsentationen, etwas betrübt waren.

Sportfest

Benjamin Kloss, Julia Högerle

Am Dienstag der ersten Woche fand das Sportfest statt, auf das wir uns alle sehr freuten. Dort bewiesen wir, dass wir nicht nur im Kurs ein

eingespieltes Team waren, sondern auch noch die verschiedensten sportlichen Aufgaben gemeinsam meistern konnten. Diese gingen von der Aufgabe, den Akademiebus einen Berg hoch zu ziehen, einen Slalom mit Taucherflossen zu absolvieren bis hin zum Teebeutelweitwurf.

Manche Aufgaben, wie beispielsweise einen Hula-Hoop-Reifen in einer Reihe zu transportieren, erledigten wir so, dass wir dachten, wir hätten die Aufgabe hervorragend gemeistert. Aber im Vergleich mit den anderen Kursen stellte sich heraus, dass wir nur im Mittelfeld rangierten. Am besten schnitten wir ab, als wir alle auf einem Tuch standen und dieses umdrehen mussten, ohne dass jemand den Boden berührte. Bei uns standen am Ende noch sechs Leute auf dem Tuch, während es bei vielen anderen Kursen nur noch drei Personen waren. Zusätzlich erledigten wir die Aufgabe am schnellsten.

Gruppenfoto beim Sportfest.

Unsere Paradedisziplin bestand zweifelsohne im Anfeuern unseres Kurses. Lauthals schrien wir unser Motto: „Induktions – Beweis“. Wir hatten uns aber auch für jeden anderen Kurs einen Schlachtruf überlegt. Beispielsweise begrüßten wir die Geophysiker mit dem Ruf „Atmos – Phäre“.

Beim Finale ging es darum, einen Eimer mit Wasser zu füllen, indem man das Wasser mithilfe eines Schwammes aus einer großen Wanne holen musste. Auch hier waren wir nicht die Besten, erzielten aber ein akzeptables Ergebnis. Dabei darf man nicht vergessen, dass wir dezimiert waren, da Niklas leider fehlte.

Da wir alle unser Bestes gegeben hatten, waren wir auch schon sehr gespannt auf das Ergebnis, welches am Bergfest bekannt gegeben wurde.

Unsere Siegerurkunde.

Zu unserer eigenen Überraschung gewannen wir vor TheoPrax das Sportfest. Als Preis gab es einen schönen Obstkorb, bei dem wir erst später feststellten, dass unter dem Obst vier Packungen Toffifee versteckt waren.

Nachwort

Die Kursteilnehmer

Nicht nur beim Sportfest hielten wir zusammen. Auch im Kurs spürte man zu jeder Zeit die gute und entspannte Atmosphäre. Wir haben uns die ganze Akademie über sehr gut verstanden, weshalb wir uns immer wohl fühlten. Rückblickend sind wir sehr begeistert von unserem Kurs und konnten viele neue Erfahrungen sammeln. Das haben wir vor allem unseren motivierenden und hilfsbereiten Kursleitern sowie unserem tollen Schülermentor zu verdanken. Diese haben uns nicht nur viel beigebracht, sondern auch bei Laune gehalten. Zusammen hatten wir unheimlich viel Spaß und haben dabei viel gelacht. Diese wunderschöne Zeit werden wir noch lange in Erinnerung behalten, da wir uns alle sehr ans Herz gewachsen sind.

Danksagung

Die JuniorAkademie Adelsheim / Science-Academy Baden-Württemberg fand in diesem Jahr bereits zum 12. Mal statt. Daher möchten wir uns an dieser Stelle bei denjenigen bedanken, die ihr Stattfinden überhaupt möglich gemacht haben.

Die JuniorAkademie Adelsheim ist ein Projekt des Regierungspräsidiums Karlsruhe, das im Auftrag des Ministeriums für Kultus, Jugend und Sport, Baden-Württemberg und mit Unterstützung der Bildung & Begabung gGmbH Bonn für Jugendliche aus dem ganzen Bundesland realisiert wird. Wir danken daher dem Schulpräsidenten im Regierungspräsidium Karlsruhe, Herrn Prof. Dr. Werner Schnatterbeck, der Referatsleiterin Frau Leitende Regierungsschuldirektorin Dagmar Ruder-Aichelin, Herrn Jurke und Herrn Rechentin vom Ministerium für Kultus, Jugend und Sport sowie dem Koordinator der Deutschen Schüler- und JuniorAkademien in Bonn, Herrn Volker Brandt.

Die Akademie wurde finanziell in erster Linie durch die H. W. & J. Hector Stiftung, durch die Stiftung Bildung und Jugend sowie den Förderverein der Science-Academy unterstützt. Dafür möchten wir an dieser Stelle allen Unterstützern ganz herzlich danken.

Wie in jedem Jahr fanden die etwas über einhundert Gäste sowohl während des Eröffnungswochenendes und des Dokumentationswochenendes als auch während der zwei Wochen im Sommer eine liebevolle Rundumversorung am Eckenberg-Gymnasium mit dem Landesschulzentrum für Umwelterziehung (LSZU) in Adelsheim. Stellvertretend für alle Mitarbeiter möchten wir uns für die Mühen, den freundlichen Empfang und den offenen Umgang mit allen bei Herrn Oberstudiendirektor Meinolf Stendebach, dem Schulleiter des Eckenberg-Gymnasiums, besonders bedanken.

Zuletzt sind aber auch die Kurs- und KüA-Leiter gemeinsam mit den Schülermentoren und der Assistenz des Leitungsteams diejenigen, die mit ihrer hingebungsvollen Arbeit das Fundament der Akademie bilden. Ein besonderer Dank gilt an dieser Stelle Jörg Richter, der auch in diesem Jahr für die Gesamterstellung der Dokumentation verantwortlich war.

Diejenigen aber, die die Akademie in jedem Jahr einzigartig werden lassen und die sie zum Leben erwecken, sind die Teilnehmerinnen und Teilnehmer. Deshalb möchten wir uns bei ihnen und ihren Eltern für ihr Vertrauen ganz herzlich bedanken.