



Algorithmik

Notation · Effizienz · Suchen · Sortieren · erste Graphenalgorithmen

FÜR DEN INFORMATIKUNTERRICHT AM GYMNASIUM

INF school

Inhaltsverzeichnis

1	Einführung und Terminologie	5
	Eigenschaften von Algorithmen	6
	Beschreiben von Algorithmen	10
2	Suchen	19
	Lineare Suche	21
	Schneller Suchen	24
	Binäre Suche in Python	33
3	Sortieren	37
	Bubblesort	39
	Wie schnell ist Bubblesort?	43
	Selectionsort	46
	Effizient sortieren	50
4	Graphen und Graphenalgorithmen	63
	Grundbegriffe	66
	Eulerzüge und Eulertouren	78
	Graphen durchsuchen und erkunden	82
	Minimale Spannbäume	108
	Repetitionsaufgaben	115

Über dieses Skript

Es freut mich sehr, dass du¹ dich entschieden hast, dieses Lehrmittel zu verwenden. Ich hoffe, dass es dich auf deinem Weg zur gymnasialen Matura begleiten und unterstützen wird, indem es dein Interesse am Fachgebiet Informatik weckt, dich zur selbstständigen Erkundung der angesprochenen Themengebiete motiviert, und dir die Grundlagen bietet, deinen Weg in der digitalen Zukunft zu beschreiten.

Was ist der Youtube-Algorithmus, wie sucht Google, und wie findet mein Navi den Weg von Zürich nach Genf? Der Begriff des Algorithmus ist in aller Munde. Was man darunter versteht, und wie uns diese Algorithmen helfen, den Alltag besser zu bewältigen, soll in diesem Skript beleuchtet werden. Die Flughöhe versuche ich so hoch wie möglich, und so niedrig wie nötig zu halten. Ein intuitives Verständnis der Materie ist mir wichtig, genauso aber soll man wichtige Konzepte im Detail anschauen und beleuchten. Zudem sollen für interessierte SuS weiterführende Informationen für die eigenständige Vertiefung zur Verfügung gestellt werden.

Feedback

Egal ob du zu den Lehrpersonen oder zu den Lernenden gehörst: Wenn du Fehler oder Ungereimtheiten findest, Verbesserungsvorschläge hast, oder andere Ideen hast, wie man dieses Lehrmittel verbessern könnte, dann melde dich ungeniert bei mir! Eine kurze E-Mail an feedback@koehlf.ch genügt. Davon lebt dieses Skript, dadurch wird es besser, und du kriegst nebenbei auch einen Eintrag in der *List of bug hunters*.

Danke!

Vielen Dank an Sabina Schellenberg, für das Erkennen zahlreicher Typos und grammatikalischer Ungereimtheiten, das oft besser funktioniert als jegliche automatisierte Alternativen.

Ein grosses Dankeschön geht auch an die ganze $\text{\LaTeX}$ -Community auf StackExchange, ohne welche ich dieses Skript niemals in dieser Form hätte schreiben können.

Zu guter Letzt möchte ich mich bei meinen Eltern bedanken, die mich gross² gezogen haben, mich jederzeit in all meinen Zielen unterstützt haben und immer an mich geglaubt haben. Tusen takk! ♥

©2026 Florian Köhl

¹bzw. deine Lehrperson

²sehr gross

01110011011000110110100001101110110111101101100

Trotz sorgfältiger Recherche und mehrmaligem Durchlesen schleichen sich natürlich auch in diesem Skript Fehler ein. Danke an alle aufmerksamen Schüler*innen und Lehrpersonen, die durch das Melden von Fehlern oder Ungereimtheiten helfen, dieses Skript zu verbessern!

- Hugo Vetter
- Joel Spiess
- Seung Hee Ma
- Alexander Flick
- Flurina Bagattini

Einführung und Terminologie

Das Wort **Algorithmus** ist seit einiger Zeit in aller Munde. So hört man von Algorithmen, die Gesichtserkennung auf dem Smartphone ermöglichen, oder vom Youtube-Algorithmus, der bestimmt, wer wann welche Videos in welcher Reihenfolge zu sehen kriegt¹.

Algorithmen sind auch der Grund, weshalb ein Navigationssystem den schnellsten Weg von A nach B findet, ein Mensch keine realistische Chance mehr hat, einen Computer in einem Schachspiel zu schlagen, oder dass wir im Internet sicher kommunizieren und private und sensible Daten austauschen können.

Doch was versteht man eigentlich unter einem Algorithmus? Wie beschreiben wir Algorithmen, und worauf müssen wir dabei achten? Was macht einen guten Algorithmus aus, und wann ist ein Algorithmus "besser" als ein anderer? Diese und weitere Fragen wollen wir gemeinsam anschauen und erkunden.

ALGORITHMUS



Ein **Algorithmus** ist eine präzise, eindeutige Handlungsvorschrift zur Lösung eines Problems oder einer Klasse von Problemen.

Ein Algorithmus besteht aus endlich vielen wohldefinierten Einzelschritten.

Auf der intuitiven Ebene können wir einen Algorithmus also als *präzise formuliertes Rezept* oder als *unmissverständlich beschriebene Strategie* beschreiben.

Als Analogie kann man sich das Zusammenbauen eines Möbelstücks vorstellen. Die Anleitung stellt dabei den Algorithmus dar. Die Person, welche das Möbelstück nach dieser Anleitung zusammensetzt handelt genau nach dieser Vorschrift². Heutzutage, im Kontext der Informatik, geht man meist davon aus, dass Algorithmen von Computern ausgeführt werden. Dies muss

¹Und welche/r content creator dadurch wie viel verdient

²Es lässt sich bekanntlich darüber streiten, ob die Anleitungen zum Zusammenbauen von Möbelstücken präzise und eindeutig definiert sind. Im Sinne der Analogie nehmen wir dies hier mal einfach so an.

aber nicht zwingend der Fall sein. Falls du schon einmal zwei Zahlen schriftlich zusammengezählt, den Satz des Pythagoras angewendet, oder den grössten gemeinsamen Teiler zweier Zahlen berechnet hast, bist du auch schon mal einem Algorithmus gefolgt! Da es sich um eine ausgearbeitete Anleitung handelt, sollen Algorithmen auch von Personen durchgeführt werden können, die keine Experten auf dem entsprechenden Gebiet sind.

Eigenschaften von Algorithmen

Jeder Algorithmus, der dem Namen gerecht werden will, muss bestimmte Eigenschaften erfüllen. Im folgenden Abschnitt werden die fünf wichtigsten definiert und anhand des Algorithmus zur Berechnung des grössten gemeinsamen Teilers (ggT) zweier positiven ganzen Zahlen erklärt.



BEISPIEL: GGT IN WORTEN

Gegeben sind zwei ganze positive Zahlen a und b .
Solange a nicht gleich b ist, ziehe die kleinere der beiden Zahlen von der grösseren ab, und überschreibe die letztere mit dem Resultat.
Wenn a und b gleich sind, so entsprechen diese genau dem grössten gemeinsamen Teiler.

- **Input:** *Jeder Algorithmus hat einen genau definierten (oder keinen) Input.*
Im Beispiel oben sind dies die beiden Zahlen, von denen man den ggT bestimmen möchte. Weiter ist angegeben, dass es sich um positive ganze Zahlen handelt, welche endlich gross sind.
- **Output:** *Jeder Algorithmus liefert ein genau definiertes, korrektes (oder kein) Resultat.*
Im obigen Beispiel ist das der ggT der beiden gegebenen Zahlen.
- **Terminierung:** *Für jeden Input muss der Algorithmus nach einer endlichen Anzahl Schritte terminieren.*
Da wir bei beim Berechnen des ggT immer die kleiner von der grösseren Zahl abziehen, kann keine der Zahlen je negativ werden. Gleichzeitig werden sie aber immer kleiner, müssen folglich also irgendwann gleich gross sein, worauf der Algorithmus terminiert. Gerät eine Handlungsvorschrift aber in eine Endlosschleife, so terminiert sie zum Beispiel nicht.
- **Finitheit:** *Der Algorithmus muss in endlich vielen Worten beschrieben werden können und darf während der Ausführung nur begrenzt (Speicher)platz in Anspruch nehmen*
Der erste Punkt ist bei unserem Beispiel klar erfüllt. Auch die zweite Bedingung ist schnell zu zeigen: Wir benötigen nur zwei Variablen, denen wir jeweils eine endlich grosse Zahl zuweisen.

- **Eindeutigkeit:** *Jeder Schritt muss genau und unmissverständlich formuliert sein.*
Die ausführende Partei muss genau wissen, was mit diesem Schritt gemeint ist. Bei unserem ggT-Beispiel ist dies sicherlich der Fall.
Beispiele für eine ungenaue Formulierung findet man zum Beispiel in Backrezepten: "Kuchen für 30-40 Minuten bei mittlerer Hitze backen." Da fragt man sich: Ja wie lange jetzt? Und was heisst mittlere Hitze?
- **Effektivität:** *Jeder Schritt muss durchführbar sein.*
Bei der Berechnung des ggT nutzen wir lediglich Subtraktionen und Vergleiche zweier Zahlen. Beide sind sicherlich durchführbar. Eine Division durch 0 hingegen wäre ein Beispiel für eine Operation, die nicht durchführbar ist.



AUFGABE 1: EIN LECKERER ALGORITHMUS?

Betrachte das folgende Rezept und überleg dir, welche Eigenschaften von Algorithmen dieses erfüllt bzw. eben nicht erfüllt. Begründe deine Antworten jeweils kurz.

Hallo,
hier ist als das Rezept fuer die Rueblitorte.
6 Eier
300 gr. Zucker
300 gr. geriebene Mandel oder Haselnuesse
250 gr. rohe, geriebene Karotten
50 gr. Mehl
2 Kaffeeloeffel Backpulver
gehackte Zitronenschale

Eigelb, Zucker und Zitronenschale 10 min. rühren.
Mandeln und Karotten begeben. Zuletzt Mehl, Backpulver
und Eischnee unterheben.
1 Stunde bei maessiger Hitze backen.

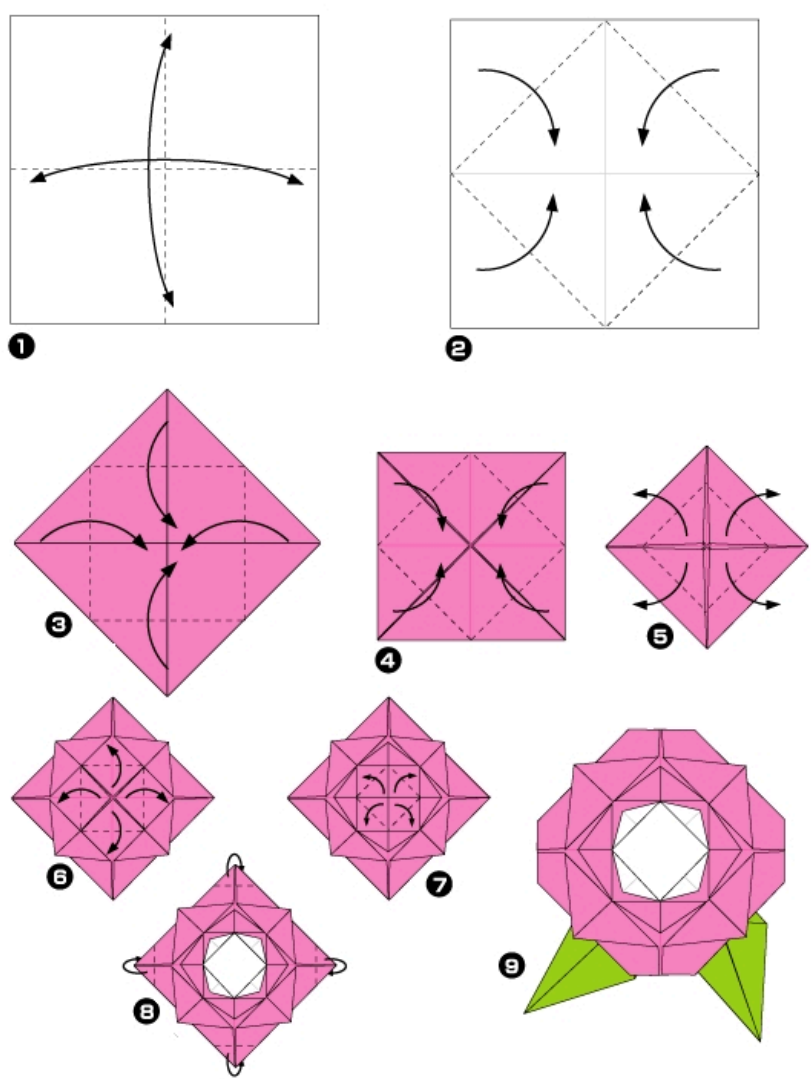
Glasur: 250 gr. Puderzucker, 1 Loeffel Kirsch-
wasser, 2 Loeffel Wasser

Viel Spass beim Backen!



AUFGABE 2: ORIGAMI

Wie sieht es mit der folgenden Bastelanleitung aus?



Quelle: Edwinsitho Asprilla, CC BY-SA 4.0 <<https://creativecommons.org/licenses/by-sa/4.0/>>, via Wikimedia Commons



★ AUFGABE 3: EIN GEWINN-ALGORITHMUS

In dieser Aufgabe sollst du einen eigenen Algorithmus entwickeln, um ein Spiel zu gewinnen.

Alice und Bob spielen eine abgewandelte Version des Spiels *Nim*. Auf dem Tisch liegen initial 21 Streichhölzer. Die Person, die am Zug ist, darf davon entweder 1, 2 oder 3 Hölzchen wegnehmen. Danach ist die zweite Person an der Reihe. Wer das letzte Hölzchen nehmen kann, hat gewonnen.

Alice darf den ersten Zug machen. Kannst du ihr helfen, das Spiel immer zu gewinnen?

- (a) Beschreibe zuerst eine Spielsituation, bei der man, sofern man als nächstes an der Reihe ist, immer verliert.
- (b) Überleg dir nun, wie man eine solche Situation erzwingen kann.
- (c) Beschreibe nun eine Strategie, mit der die Person, welche den ersten Zug macht, immer gewinnt. Achte auf eine eindeutige und präzise Formulierung.
- (d) Funktioniert deine Strategie auch, wenn zu Beginn 20 Hölzchen auf dem Tisch liegen?

Schneller Suchen

Geht es besser? Das ist eine Frage, die man sich im Bereich der Algorithmik oft stellt. Gemeint ist: Gibt es einen Algorithmus, der ein gegebenes Problem **effizienter** lösen kann? Aber was heisst eigentlich effizient, und wie können wir diese Effizienz messen?



AUFGABE 19: WAS HEISST EFFIZIENT?

Überleg dir, wie du die Effizienz eines Algorithmus messen würdest. Verwende als Beispiel den Algorithmus der linearen Suche.

Ein intuitiver Ansatz, wie wir die Effizienz eines Algorithmus messen könnten, ist via Zeitmessung. Wir nehmen uns eine Stoppuhr zur Hand und messen die Zeit, die der Algorithmus¹ benötigt, bis er das Problem gelöst hat. Wir wollen diesen Ansatz an einem Beispiel verdeutlichen.

Ein Supermarkt verkauft in einer Filiale 10000 verschiedene Artikel. Jeder Artikel hat eine eindeutige Artikelnummer, und die Daten zu einem Artikel sind in einer Datenbank abgespeichert:

ARTIKELNUMMER	PREIS	BESCHREIBUNG
...	...	...
3080191	3.30	Eier (Freiland)
3081654	1.80	Milch
3090581	4.40	WC-Papier
4859939	3.60	Butterzopf
...	...	...

Du tätigtst einen Einkauf in diesem Supermarkt und hast 20 Artikel in deinem Warenkorb, die du nun an der Kasse scannen möchtest. Dabei wird die Artikelnummer des eingescannten Produktes in dieser Datenbank gesucht. Das System des Supermarktes setzt dabei auf die lineare Suche, geht also jeden Eintrag in der Datenbank durch, bis die gesuchte Artikelnummer gefunden wurde. Wir nehmen an, dass das System 1 ms benötigt, um eine einzige Artikelnummer in der Datenbank zu lesen und mit der gesuchten zu vergleichen.

¹bzw. eine Implementation davon, beispielsweise als Programm



AUFGABE 20: WARTEZEIT AN DER KASSE

Wie lange dauert es im Durchschnitt, bis...

- (a) ...ein bestimmter Artikel im System gefunden wird?
- (b) ...dein gesamter Warenkorb eingescannt ist?

Nun, wenn wir annehmen, dass das System im Durchschnitt die Hälfte aller Artikel anschauen muss, bis die gesuchte Artikelnummer gefunden wurde, so sind dies etwa 5000 Artikel². Das Prüfen dieser 5000 Artikel dauert $5000 \cdot 1 \text{ ms}$, also 5 Sekunden. Bis dein ganzer Warenkorb eingescannt ist, vergehen also im Durchschnitt 100 Sekunden!

Nun, wie könnten wir diese Wartezeit verkürzen? Eine naheliegende Lösung wäre, das System etwas zu modernisieren. 1 ms pro Vergleich einer Artikelnummer ist deutlich zu langsam. Wenn wir diese Zeit auf 0.1 ms verkürzen könnten, zum Beispiel durch Anschaffung schnellerer Rechner, so würde die Wartezeit an der Kasse um einen Faktor 10 verkürzt, würde also "nur" noch 10 Sekunden dauern. Haben wir damit aber den Algorithmus effizienter gemacht? Nicht wirklich, oder? Das Vorgehen ist immer noch das gleiche, die ausführende Partei (das Kassensystem) arbeitet nun einfach schneller. Natürlich wären die Kunden dieses Supermarktes mit dieser Lösung zufrieden. Eine gute Möglichkeit, die Effizienz des Suchalgorithmus zu messen, bietet dieser Stoppuhr-Ansatz aber leider in unserem Fall nicht: Eine Filiale mit einem schnelleren Kassensystem schneidet viel besser ab als eine, bei der noch das alte Kassensystem im Einsatz ist. Der Algorithmus dahinter ist aber der gleiche... Wir messen hier also nicht wirklich die Effizienz des Algorithmus (sonst würden wir in beiden Fällen ja das gleiche Resultat erwarten, da beide Filialen den selben Algorithmus einsetzen), sondern eher die Leistung des Kassensystems.

Aber vielleicht gibt es ja eine Möglichkeit, den Algorithmus selbst, also die Art und Weise, die Artikel gesucht werden, zu verbessern, ohne dass der Supermarkt das Kassensystem erneuern muss? Um diese Frage zu beantworten, wechseln wir vorerst das Szenario und werden versuchen, so eine Lösung zu erarbeiten. Wir werden diese dann später auf das Supermarkt-Beispiel anwenden und schauen, wie diese abschneidet.

²Ganz genau genommen sind es 5000.5 Artikel im Schnitt. Wenn wir annehmen, dass wir jeden Artikel einmal suchen, so müssen wir total $1 + 2 + 3 + \dots + 10000 = \frac{10000 \cdot 10001}{2}$ Artikel anschauen. Im Schnitt sind das dann $\frac{10000 \cdot 10001}{2 \cdot 10000} = \frac{10001}{2} = 5000.5$ Artikel.

Im allgemeinen Fall mit n Artikeln müssen wir im Schnitt also $\frac{n+1}{2}$ Artikel betrachten.

Kennst du das Spiel Hangman³, bei dem es darum geht, ein bestimmtes Wort zu erraten? Wir wollen eine Variante davon spielen, bei der wir statt einem Wort eine Zahl erraten müssen. Das Spiel funktioniert wie folgt:

ZAHLEN-HANGMAN

Alice merkt sich eine Zahl zwischen 1 und 1000. Bob hat 10 Versuche, die geheime Zahl von Alice zu erraten. In jedem Versuch darf er eine Zahl raten. Falls Bob richtig geraten hat, so gewinnt er. Ansonsten teilt Alice ihm mit, ob ihre geheime Zahl grösser oder kleiner ist als die eben von Bob geratene Zahl. Wenn Bob mehr als 10 Versuche benötigt, so gewinnt Alice.



AUFGABE 21: ZAHLEN-HANGMAN SPIELEN

- (a) Spiel mit einem Partner oder einer Partnerin Zahlen-Hangman. Einmal als Alice und einmal als Bob.
- (b) Welche Strategie verwendest du, wenn du als Bob die Zahl erraten musst?

Nun, ziemlich sicher kommt hier niemand auf die Idee, die gesuchte Zahl mittels linearer Suche zu finden:

Ist es die 1? – Nein, die gesuchte Zahl ist grösser.

Ist es die 2? – Nein, die gesuchte Zahl ist grösser.

Ist es die 3? – Nein, die gesuchte Zahl ist grösser.

...

Die Chance, die Zahl damit in nur 10 Versuchen zu erraten, ist relativ klein. Wenn Alice sich die 1000 aussucht, so müssten wir sogar 1000 mal raten!

Es ist doch viel zielführender, pro Raterunde so viele Zahlen wie möglich zu eliminieren. Dies können wir tun, indem wir immer von allen noch möglichen Zahlen immer diejenige raten, die sich (etwa) in der Mitte befindet. Damit können wir pro Versuch nämlich jeweils die Hälfte aller verbleibenden Zahlen ausschliessen und kommen viel schneller zum Ziel! Das folgende Beispiel zeigt, wie diese Strategie funktioniert.

³<https://de.wikipedia.org/wiki/Galgenmännchen>



BEISPIEL: ZAHLEN-HANGMAN GEWINNEN

Alice denkt an die Zahl 42. Bob würde mit der eben beschriebenen Strategie also wie folgt vorgehen:

Ist es die 500? – Nein, die gesuchte Zahl ist kleiner.

Ist es die 250? – Nein, die gesuchte Zahl ist kleiner.

Ist es die 125? – Nein, die gesuchte Zahl ist kleiner.

Ist es die 63? – Nein, die gesuchte Zahl ist kleiner.

Ist es die 32? – Nein, die gesuchte Zahl ist grösser.

Nach nur 5 Versuchen weiss Bob bereits, dass die gesuchte Zahl zwischen 32 und 63 liegen muss. Als nächstes würde er also die Zahl 48 raten, da sich diese in der Mitte zwischen 32 und 63 befindet:

Ist es die 48? – Nein, die gesuchte Zahl ist kleiner.

Aha, die gesuchte Zahl ist also irgendwo zwischen 32 und 48...

Ist es die 40? – Nein, die gesuchte Zahl ist grösser.

Ist es die 44? – Nein, die gesuchte Zahl ist kleiner.

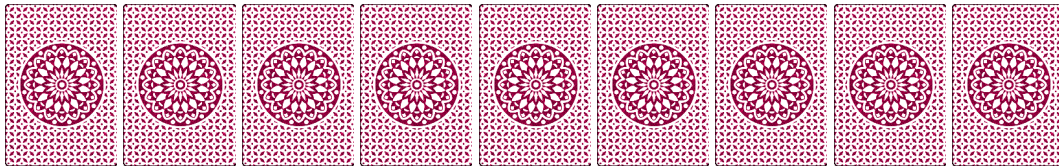
So, jetzt ist Bob der gesuchten Zahl schon sehr nahe: Es ist entweder die 41, die 42 oder die 43...

Ist es die 42? – Jawohl, du hast gewonnen!

Uff, das war knapp, aber Bob hat es mit 9 Versuchen geschafft, er hatte beim letzten Versuch etwas Glück. Doch auch wenn die 42 nicht korrekt gewesen wäre, so hätte er trotzdem durch das Feedback von Alice die richtige Zahl im letzten Versuch erraten. In der Tat ist es so, dass Bob mit dieser Strategie immer gewinnt, egal welche Zahl Alice sich aussucht!

Wenn wir die beiden Ansätze vergleichen wollen, so könnten wir hier beispielsweise die Anzahl Versuche zählen, die nötig sind. Im besten Fall sind beide Strategien gleich gut, nämlich genau dann, wenn wir im ersten Versuch korrekt raten. Bei der linearen Suche wäre das der Fall, wenn Alice sich die 1 merkt, bei der optimalen Strategie dann, wenn sie an die 500 denkt. Doch bei der optimalen Strategie müssen wir maximal, also im schlechtesten Fall, nur 10 mal raten, beim Ansatz der linearen Suche sind es hingegen 1000 Versuche. Wir können also sagen, dass die lineare Suche hier deutlich ineffizienter ist, ohne dabei irgend eine Zeitmessung vorzunehmen.

Bevor wir diese Strategie auf unser Supermarkt-Beispiel anwenden, holen wir nochmals die Spielkarten hervor. Dieses Mal liegen 9 Spielkarten der Farbe Herz verdeckt auf dem Tisch:



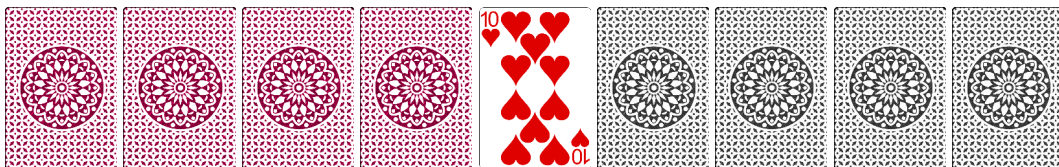
Die Regeln sind gleich wie zuvor: du kannst immer jeweils eine davon umdrehen. Allerdings weist du, dass die Karten aufsteigend **sortiert** sind. Das bedeutet, dass der Wert einer Karte immer kleiner oder gleich gross ist wie die Werte der Karten rechts von ihr.



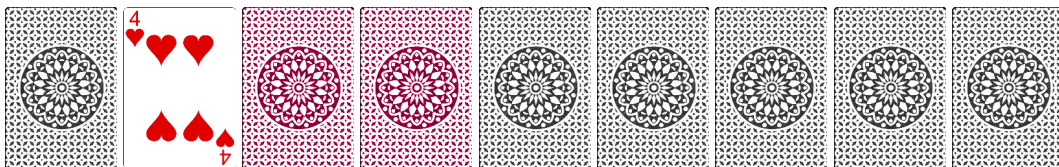
AUFGABE 22: KARTEN SCHNELLER SUCHEN

Welche Strategie würdest du hier anwenden, um eine Karte, beispielsweise die Herz 7, zu suchen?

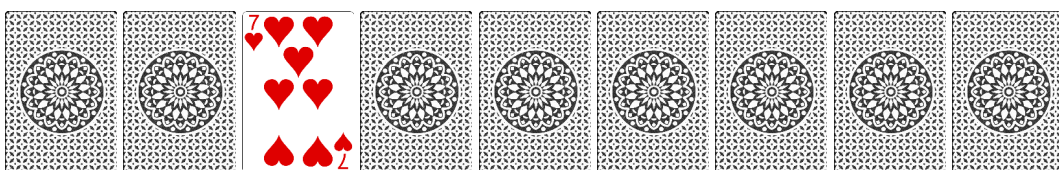
Auf den ersten Blick ist vielleicht nicht ganz klar, wie wir den Hangman-Ansatz hier anwenden können. Schliesslich wissen wir dort ja gar nicht, was wir suchen. Aber das Prinzip, welches uns im Zahlen-Hangman immer gewinnen lässt, basiert ja darauf, dass wir pro Zug immer möglichst viele Möglichkeiten ausschliessen wollen. Das funktioniert auch in diesem Kartenszenario! Als erstes decken wir die mittlere Karte auf:



Was können wir daraus schliessen? Richtig, die gesuchte Karte, die Herz 7, muss also links von der eben aufgedeckten Karte liegen! Die ersten vier Karten müssen wir also nicht durchsuchen, dort wird unsere Zielkarte definitiv nicht zu finden sein! Auf die verbleibenden vier Karten wenden wir nun wieder die gleiche Strategie an, und decken die mittlere Karte auf. Hier haben wir nun zwei Möglichkeiten: die 2. oder die 3. Karte. Wir entscheiden uns für erstere:



Nun wissen wir also, dass es entweder die 3. oder die 4. Karte sein muss. Wir tippen auf wieder auf erstere und haben Glück:



Diese Vorgehensweise nennt man **binäre Suche**. Sie ist die effizienteste Art, eine **sortierte** Liste von Elementen zu durchsuchen⁴.



AUFGABE 23: ERFOLGLOSE BINÄRE SUCHE

Nehmen wir an, die Karte Herz 7 wäre nicht unter den obigen Karten gewesen. Wie hättest du erkannt, dass die Karte nicht dabei ist?

Im obigen Beispiel mussten wir 3 Karten aufdecken, bis wir die gesuchte Karte gefunden haben. Mit dem Ansatz der linearen Suche wären ebenfalls 3 gewesen. Das klingt jetzt auf den ersten Blick nicht nach einer gewaltigen Verbesserung...Aber wie sähe die Situation aus, wenn wir nicht nur 9, sondern 100, 1000 oder 10000 Karten durchsuchen müssten?



★ AUFGABE 24: BINÄRE SUCHE ANALYSIEREN

Wie viele Karten musst du mit der binären Suche maximal aufdecken, wenn du...

- (a) 1 Karte vor dir hast?
- (b) 2 Karten vor dir hast?
- (c) 3 Karten vor dir hast?
- (d) 7 Karten vor dir hast?
- (e) 8 Karten vor dir hast?
- (f) 15 Karten vor dir hast?
- (g) 31 Karten vor dir hast?

Tipp: Betrachte auch den Fall, dass die Karte, die du suchst, gar nicht unter den verdeckten Karten zu finden ist.



★★ AUFGABE 25: UND BEI n KARTEN?

Wie viele Versuche benötigst du bei n Karten?

⁴Zumindest dann, wenn dabei nur Vergleiche zwischen Elementen erlaubt sind

Graphen und Graphenalgorithmen

In diesem Kapitel befassen wir uns mit Graphen und ersten, einfachen Graphenalgorithmen. Das Themengebiet der Graphentheorie gehört wie die Algorithmik selbst zur theoretischen Informatik. Wir werden uns aber bemühen, dieses durchaus spannende Thema so intuitiv wie möglich anzugehen und möglichst viel daraus mitzunehmen, obwohl wir nicht allzu tief darin abtauchen können. Wir werden lernen, geeignete Probleme bzw. deren Ausgangslage als Graphen zu modellieren und mit Hilfe von Graphenalgorithmen zu lösen. Eine ideale Art, unser Abstraktionsvermögen, also das Erkennen von wichtigen bzw. das Weglassen von unnötigen Elementen einer Problemstellung, zu trainieren und auszuweiten.

Wir beginnen unsere Reise durch das Land der Graphen mit einer Aufgabe, anhand derer wir den Begriff eines Graphen selbst sowie die damit verbundene Abstraktion einführen.



AUFGABE 49: ENDLICH URLAUB!

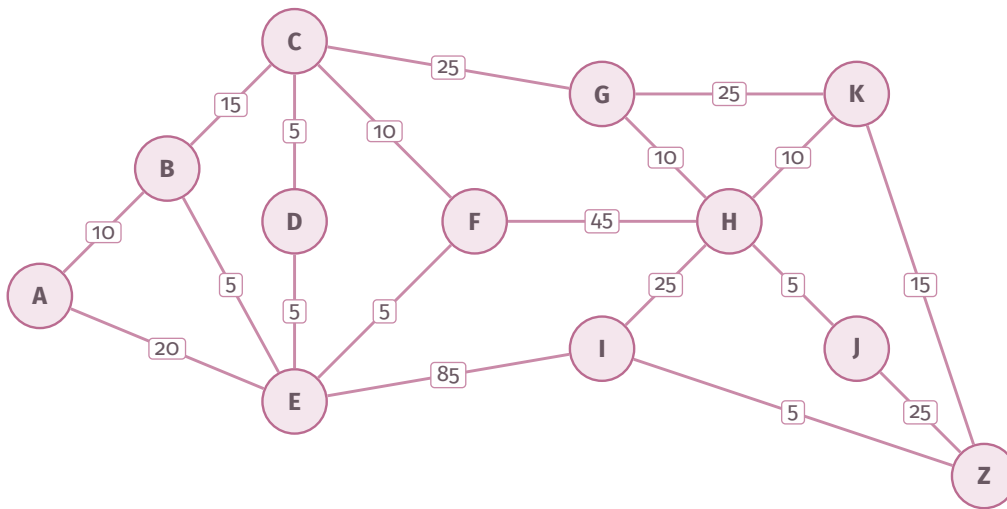
Nach einem anstrengenden Semester freust du dich auf die anstehenden Ferien: Urlaub in Zettwil! Fragt sich nur noch, wie du am besten dorthin kommst. Von deinem Zuhause in Ahbach gibt es leider keine direkte Verbindung, und der online-Fahrplan des regionalen S-Bahn-Systems ist momentan mal wieder überlastet und nicht erreichbar...

Nach einer kurzen Recherche konntest du folgende Reisezeiten (in Minuten) zwischen den einzelnen Stationen ermitteln:

VON \ NACH	A	B	C	D	E	F	G	H	I	J	K	Z
A		10			20							
B	10		15		5							
C		15		5		10	25					
D			5		5							
E	20	5		5		5			85			
F			10		5			45				
G			25					10			25	
H						45	10		25	5	10	
I					85			25				5
J								5				20
K							25	10				15
Z									5	20	15	

Wie kommst du am schnellsten von Ahbach (A) nach Zettwil (Z)?

Nun, diese Darstellung ist, wie du wohl gemerkt hast, nicht sehr übersichtlich. Viel einfacher ist es doch, wenn wir die Verbindung in einer Art S-Bahn-Netzwerk darstellen:



AUFGABE 50: KÜRZESTER WEG

Betrachte die oben dargestellten Verbindungen.

Wie kommst du am schnellsten von **A** nach **Z**? Wie viele Minuten dauert diese Reise?

Wenn du auf 85 Minuten gekommen bist: Gratuliere, du hast soeben den kürzesten Weg in einem ungerichteten, gewichteten Graphen gefunden!

Wie bitte? Ungerichtet? Gewichtet? Graph? Kürzester Weg?

Keine Angst, wir werden all diese Begriffe in diesem Kapitel kennenlernen.

Grundbegriffe

Als erstes wollen wir definieren, was ein Graph eigentlich ist:

GRAPH



Unter einem **Graph** versteht man in der Graphentheorie nichts anderes als eine nicht-leere Menge von **Knoten** (engl. *vertices*) und eine Menge von **Kanten** (engl. *edges*). Formal schreibt man deshalb meist $G = (V, E)$.

Mit Hilfe eines Graphen lassen sich auf abstrakte Weise Objekte (als Knoten) und deren Beziehungen zueinander (als Kanten) modellieren.



BEISPIEL: GRAPHEN UND IHRE DARSTELLUNG

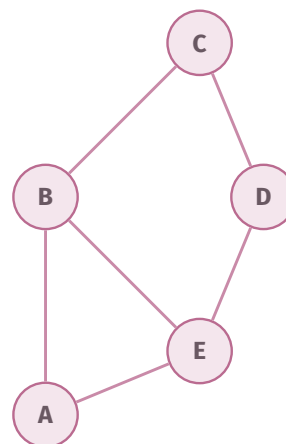
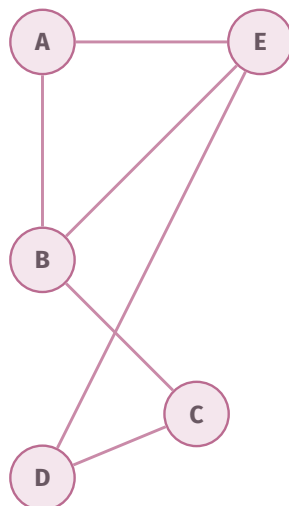
Wie wir die Graphen genau aufzeichnen, spielt keine Rolle. Solange die Knoten und Kanten übereinstimmen, so betrachten wir zwei Graphen als gleich.

Die folgenden zwei Graphen sind beispielsweise gleich: Beide bestehen aus den Knoten

$$V = \{A, B, C, D, E\}$$

und den Kanten

$$E = \{(A, B), (A, E), (B, C), (B, E), (C, D), (D, E)\}$$



ZUSAMMENHÄNGENDE GRAPHEN



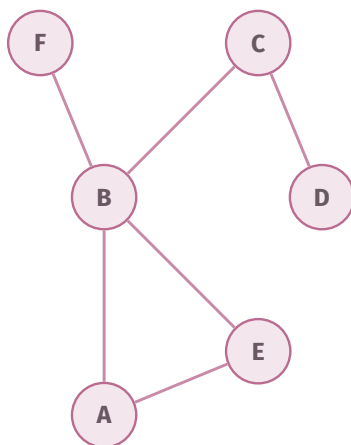
Ein Graph heisst **zusammenhängend**, wenn alle Knoten direkt oder indirekt über andere Knoten miteinander durch Kanten verbunden sind.

Einen nicht-zusammenhängenden Graphen kann man in jeweils in sich **zusammenhängende Komponenten** unterteilen.

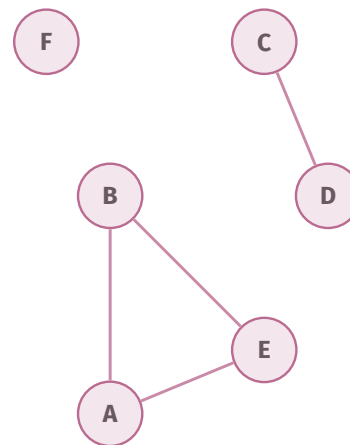


BEISPIEL: ZUSAMMENHANG

Den Begriff *zusammenhängend* können wir ganz intuitiv wie folgt auffassen: In einem zusammenhängenden Graphen können wir von jedem Knoten aus zu jedem anderen Knoten 'wandern', wenn wir den Kanten entlang laufen.



Der linke Graph ist zusammenhängend: Wir können von jedem Knoten aus zu jedem anderen gelangen.

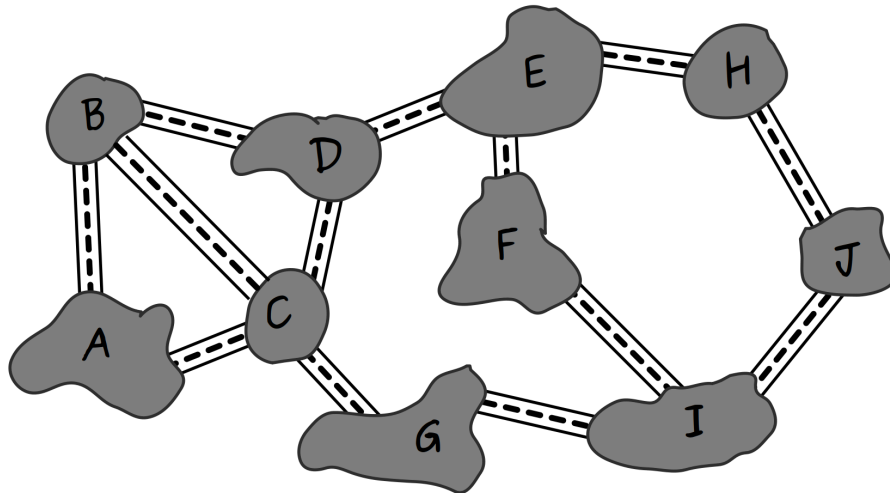


Der rechte Graph hingegen ist nicht zusammenhängend: Er besteht aus 3 Komponenten, die jeweils in sich zusammenhängend sind.



AUFGABE 51: INSELN UND BRÜCKEN

Das Inselkönigreich Tavula besteht aus 10 kleinen Inseln, zwischen denen Brücken gebaut wurden:



Eines Tages wütet ein grosser Sturm, der viele Brücken zerstört. Der folgende Schadensreport gibt an, welche Brücken vom Sturm betroffen und nicht mehr passierbar sind:

- zwischen A und C
- zwischen B und D
- zwischen C und D
- zwischen C und G
- zwischen F und I
- zwischen H und J
- zwischen I und J

- (a) Stelle die Ausgangslage nach dem Sturm als Graph dar. Was sind dabei die Knoten, was die Kanten?
- (b) Wie viele Brücken müssen mindestens wieder aufgebaut werden, damit man von jeder Insel zu jeder anderen Insel gelangen kann? Begründe deine Antwort mit Begriffen aus der Graphentheorie.