



**INFORMATIK-BIBER SCHWEIZ  
CASTOR INFORMATIQUE SUISSE  
CASTORO INFORMATICO SVIZZERA**

# Aufgaben und Lösungen 2023

## Schuljahre 9/10

<https://www.informatik-biber.ch/>

Herausgeber:

Susanne Datzko-Thut, Nora A. Escherle,  
Jean-Philippe Pellet

010100110101011001001001  
01000010010110101010011  
010100110100100101000101  
001011010101001101010011  
01001001010010010010001

# SV!A

[www.svia-ssie-ssii.ch](http://www.svia-ssie-ssii.ch)  
schweizerischerverein für informatik in d  
erausbildung // société suisse pour l'infor  
matique dans l'enseignement // società sviz  
zera per l'informatica nell'insegnamento







# Mitarbeit Informatik-Biber 2023

Masiar Babazadeh, Susanne Datzko-Thut, Jean-Philippe Pellet, Giovanni Serafini, Bernadette Spieler

Projektleitung: Nora A. Escherle

Herzlichen Dank für die Aufgabenentwicklung für den Schweizer-Wettbewerb an:

Juraj Hromkovič, Angélica Herrera Loyo, Regula Lacher und Manuel Wettstein: ETH Zürich,  
Ausbildungs- und Beratungszentrum für Informatikunterricht

Tobias Berner: Pädagogische Hochschule Zürich

Christian Datzko: Wirtschaftsgymnasium und Wirtschaftsmittelschule, Basel

Fabian Frei: CISPA - Helmholtz-Zentrum für Informationssicherheit

Sebastian Knüsli: Gymnasium Kirschgarten, Basel

Die Aufgabenauswahl wurde erstellt in Zusammenarbeit mit den Organisatoren von Bebras in  
Deutschland, Österreich, Ungarn und Litauen. Besonders danken wir:

Valentina Dagienė, Vaidotas Kinčius: Bebras.org, Litauen

Wolfgang Pohl, Jakob Schilke: Bundesweite Informatikwettbewerbe (BWINF), Deutschland

Hannes Endreß: Materna Information & Communications SE, Deutschland

Ulrich Kiesmüller: Simon-Marius-Gymnasium Gunzenhausen, Deutschland

Kirsten Schlüter: Bayerisches Staatsministerium für Unterricht und Kultus, Deutschland

Margareta Schlüter: Universität Tübingen, Deutschland

Jacqueline Staub: Universität Trier, Deutschland

Michael Weigend: WWU Münster, Deutschland

Wilfried Baumann, Liam Baumann, Josefine Hiebler: Österreichische Computer Gesellschaft, Österreich

Gerald Futschek: Technische Universität Wien, Österreich

Zsuzsa Pluhár: ELTE Informatikai Kar, Ungarn

Die Online-Version des Wettbewerbs wurde auf [cuttle.org](https://cuttle.org) realisiert. Für die gute Zusammenarbeit danken wir:

Eljakim Schrijvers, Justina Dauksaite, Arjan Huijsers, Dave Oostendorp, Alieke Stijf, Kyra Willekes:  
[cuttle.org](https://cuttle.org), Niederlande

Chris Roffey: UK Bebras Administrator, Vereinigtes Königreich

Für den Support während den Wettbewerbswochen danken wir:

Hanspeter Erni: Schulleitung Sekundarschule Rickenbach

Gabriel Thullen: Collège des Colombières, Versoix

Für die Organisation und Durchführung des Biberfinals an der ETH danken wir:

Dennis Komm, Hans-Joachim Bückenhauer, Jan Lichensteiger, Moritz Stocker: ETH Zürich,  
Ausbildungs- und Beratungszentrum für Informatikunterricht

Für die Korrektur der Finalaufgaben:

Fiona Binder, Joel Birrer, Marlene Bötschi, Danny Camenisch, Gianluca Danieletto, Alexander Frey,



Sven Grübel, Laure Guerrini, Charlotte Knierim, Richard Královič, Yanik Künzi, Kenli Lao, Sandro Marchon, Zoé Meier, Dario Nöpfer, Kai Zürcher

Für die Übersetzung der Finalaufgaben ins Französische:

Jan Schönbacher: Lycée-Collège de l'Abbaye de St-Maurice

Christoph Frei: Chragokyberneticks (Logo Informatik-Biber Schweiz)

Andrea Leu, Maggie Winter, Lena Frölich: Senarclens Leu + Partner AG

Ganz besonderen Dank gilt unseren grossen Förderern Juraj Hromkovič, Dennis Komm, Gabriel Parriaux und der Haslerstiftung. Ohne sie würde es diesen Wettbewerb nicht geben.

Die deutschsprachige Fassung der Aufgaben wurde ähnlich auch in Deutschland und Österreich verwendet.

Die französischsprachige Übersetzung wurde von Elsa Pellet und die italienischsprachige Übersetzung von Christian Giang erstellt.



**INFORMATIK-BIBER** SCHWEIZ  
**CASTOR INFORMATIQUE** SUISSE  
**CASTORO INFORMATICO** SVIZZERA

Der Informatik-Biber 2023 wurde vom Schweizerischen Verein für Informatik in der Ausbildung (SVIA) durchgeführt und massgeblich von der Hasler Stiftung unterstützt. Wettbewerbssponsoren sind das Amt für Wirtschaft und Arbeit des Kantons Zürich, der Kanton Bern, die Post sowie die UBS.

Dieses Aufgabenheft wurde am 10. Januar 2024 mit dem Textsatzsystem  $\text{\LaTeX}$  erstellt. Wir bedanken uns bei Christian Datzko für die Entwicklung und langjährige Pflege des Systems zum Generieren der 36 Versionen dieser Broschüre (nach Sprachen und Schulstufen). Das System wurde analog zum Vorgänger-System neu programmiert, welches ab 2014 gemeinsam mit Ivo Blöchliger entwickelt wurde. Jean-Philippe Pellet danken wir für die Entwicklung der **bebras** Toolchain, die seit 2020 für die automatisierte Konvertierung der Markdown- und YAML-Quelldokumente verwendet wird.

Hinweis: Alle Links wurden am 1. Dezember 2023 geprüft.



Die Aufgaben sind lizenziert unter einer Creative Commons Namensnennung – Nicht-kommerziell – Weitergabe unter gleichen Bedingungen 4.0 International Lizenz. Die Autoren sind auf S. 70 genannt.



# Vorwort

Der Wettbewerb «Informatik-Biber», der in verschiedenen Ländern der Welt schon seit mehreren Jahren bestens etabliert ist, will das Interesse von Kindern und Jugendlichen an der Informatik wecken. Der Wettbewerb wird in der Schweiz in Deutsch, Französisch und Italienisch vom Schweizerischen Verein für Informatik in der Ausbildung SVIA durchgeführt und von der Hasler Stiftung unterstützt.

Der Informatik-Biber ist der Schweizer Partner der Wettbewerbs-Initiative «Bebras International Contest on Informatics and Computer Fluency» (<https://www.bebas.org/>), die in Litauen ins Leben gerufen wurde.

Der Wettbewerb wurde 2010 zum ersten Mal in der Schweiz durchgeführt. 2012 wurde zum ersten Mal der «Kleine Biber» (Stufen 3 und 4) angeboten.

Der Informatik-Biber regt Schülerinnen und Schüler an, sich aktiv mit Themen der Informatik auseinander zu setzen. Er will Berührungsängste mit dem Schulfach Informatik abbauen und das Interesse an Fragenstellungen dieses Fachs wecken. Der Wettbewerb setzt keine Anwenderkenntnisse im Umgang mit dem Computer voraus – ausser dem «Surfen» im Internet, denn der Wettbewerb findet online am Computer statt. Für die Fragen ist strukturiertes und logisches Denken, aber auch Phantasie notwendig. Die Aufgaben sind bewusst für eine weiterführende Beschäftigung mit Informatik über den Wettbewerb hinaus angelegt.

Der Informatik-Biber 2023 wurde in fünf Altersgruppen durchgeführt:

- Stufen 3 und 4 («Kleiner Biber»)
- Stufen 5 und 6
- Stufen 7 und 8
- Stufen 9 und 10
- Stufen 11 bis 13

Jede Altersgruppe erhält Aufgaben in drei Schwierigkeitsstufen: leicht, mittel und schwierig. In den Altersgruppen 3 und 4 waren 9 Aufgaben zu lösen, mit je drei Aufgaben in jeder der drei Schwierigkeitsstufen. Für die Altersklassen 5 und 6 waren es je vier Aufgaben aus jeder Schwierigkeitsstufe, also 12 insgesamt. Für die restlichen Altersklassen waren es 15 Aufgaben, also fünf Aufgaben pro Schwierigkeitsstufe.

Für jede richtige Antwort wurden Punkte gutgeschrieben, für jede falsche Antwort wurden Punkte abgezogen. Wurde die Frage nicht beantwortet, blieb das Punktekonto unverändert. Je nach Schwierigkeitsgrad wurden unterschiedlich viele Punkte gutgeschrieben beziehungsweise abgezogen:

	leicht	mittel	schwer
richtige Antwort	6 Punkte	9 Punkte	12 Punkte
falsche Antwort	−2 Punkte	−3 Punkte	−4 Punkte



Dieses international angewandte System zur Punkteverteilung soll den Anreiz zum blossen Erraten der Lösung eliminieren.

Jede Teilnehmerin und jeder Teilnehmer hatte zu Beginn 45 Punkte («Kleiner Biber»: 27 Punkte, Stufen 5 und 6: 36 Punkte) auf dem Punktekonto.

Damit waren maximal 180 Punkte («Kleiner Biber»: 108 Punkte, Stufen 5 und 6: 144 Punkte) zu erreichen, das minimale Ergebnis betrug 0 Punkte.

Bei vielen Aufgaben wurden die Antwortalternativen am Bildschirm in zufälliger Reihenfolge angezeigt. Manche Aufgaben wurden in mehreren Altersgruppen gestellt. Diese Aufgaben hatten folglich in den verschiedenen Altersgruppen unterschiedliche Schwierigkeitsstufen.

Einige Aufgaben werden für bestimmte Altersgruppen als «Bonus» angegeben: sie haben keinen Einfluss auf die Berechnung der Gesamtpunktzahl. Diese Übungen dienen vielmehr dazu, bei mehreren TeilnehmerInnen mit identischer Punktzahl zu entscheiden, wer sich für eine mögliche nächste Runde qualifiziert.

## **Für weitere Informationen:**

SVIA-SSIE-SSII Schweizerischer Verein für Informatik in der Ausbildung

Informatik-Biber

Nora A. Escherle

<https://www.informatik-biber.ch/de/kontaktieren/>

<https://www.informatik-biber.ch/>



# Inhaltsverzeichnis

Mitarbeit Informatik-Biber 2023 . . . . .	i
Vorwort . . . . .	iii
Inhaltsverzeichnis . . . . .	v
1. Gemüsebeet . . . . .	1
2. Zug entladen . . . . .	5
3. Martinas Dorf . . . . .	7
4. Wärmer und Kälter . . . . .	11
5. Biber-Bausteine . . . . .	15
6. Brunnen . . . . .	19
7. Biberburg AG . . . . .	23
8. Ogham . . . . .	27
9. Wanderungen . . . . .	31
10. Go-Bots . . . . .	35
11. Emma erledigt . . . . .	39
12. Zerobots Mission . . . . .	43
13. Brücken bauen! . . . . .	47
14. Postfix-Notation . . . . .	51
15. Ziffernschloss . . . . .	55
16. Domino . . . . .	59
17. Anprobieren . . . . .	63
18. Konflikt-Detektor . . . . .	67
A. Aufgabenautoren . . . . .	70
B. Akademische Partner . . . . .	72
C. Sponsoring . . . . .	73
D. Weiterführende Angebote . . . . .	74





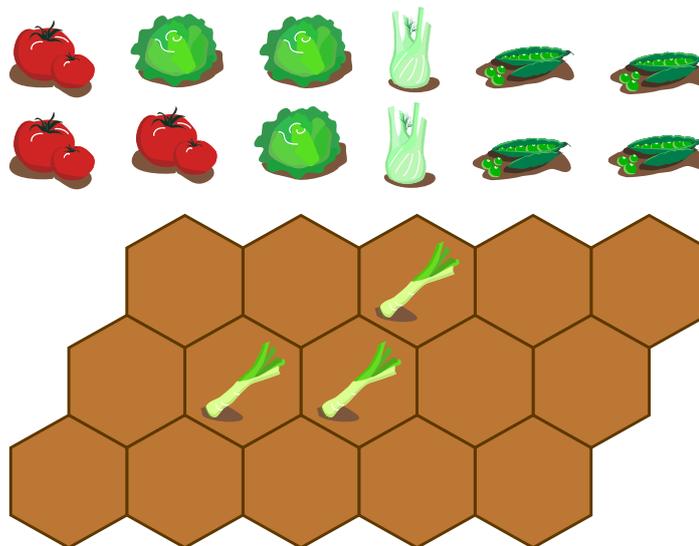
# 1. Gemüsebeet

Lisa legt ein Gemüsebeet an. Darauf will sie fünf verschiedene Gemüse pflanzen. Manche Gemüse vertragen sich gut miteinander ✓, andere nicht ⚡:



Lisa hat das Beet in sechseckige Bereiche aufgeteilt. In jeden Bereich will sie genau ein Gemüse pflanzen.

In drei Bereiche hat Lisa schon Lauch  gepflanzt.



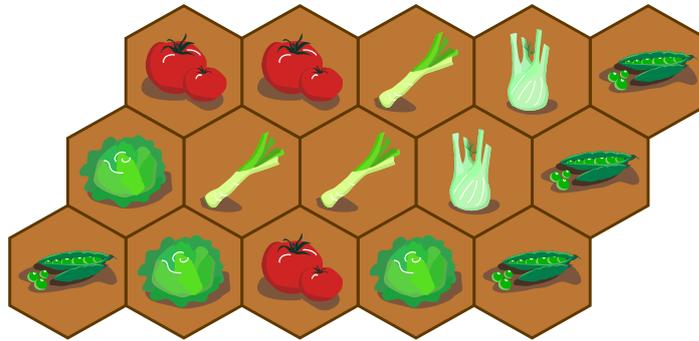
Beim Pflanzen beachtet Lisa folgende Regel: Gemüse, die sich nicht vertragen, dürfen nicht in Bereiche gepflanzt werden, die sich berühren.

*Bepflanze alle noch freien Bereiche und beachte Lisas Regel!*

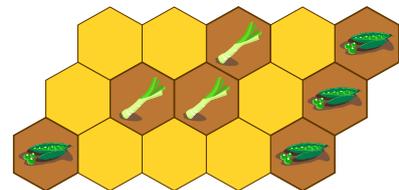


## Lösung

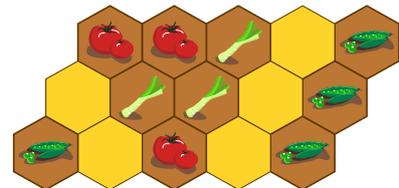
So ist es richtig:



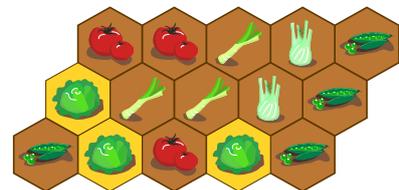
Weil Erbsen sich mit Lauch nicht vertragen, pflanzt Lisa keine Erbsen in die gelben Bereiche. Für die Erbsen bleiben nur die übrigen Bereiche.



Weil Tomaten sich mit Erbsen nicht vertragen, pflanzt Lisa keine Tomaten in die gelben Bereiche. In die übrigen Bereiche kann sie Tomaten pflanzen; Tomaten vertragen sich mit Lauch.



Weil Tomaten sich mit Fenchel nicht vertragen, pflanzt Lisa keinen Fenchel in die gelben Bereiche. Den Fenchel kann sie in die beiden Bereiche zwischen Lauch und Erbsen pflanzen. In die gelben Bereiche kann sie Salat pflanzen: Für Salat ist Lisa keine Unverträglichkeit bekannt.



## Dies ist Informatik!

Wer Gemüse so pflanzen will, dass die Ernte möglichst gross wird, muss viele *Bedingungen* beachten: Die einzelnen Sorten haben zum Beispiel unterschiedlichen Bedarf an Platz, Nährstoff und Licht. In dieser Biberaufgabe betrachten wir nur eine Art von Bedingung: die Verträglichkeit zwischen den Gemüsesorten.

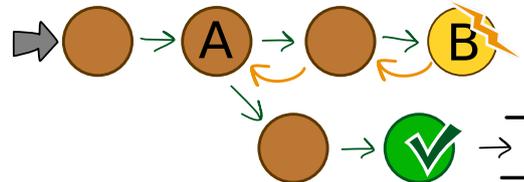
Um eine Bepflanzung von Lisas Beet zu finden, die alle Verträglichkeitsbedingungen beachtet, könnte man so vorgehen: Man probiert systematisch alle Kombinationen aus, die Gemüse auf dem Beet zu platzieren. Erst wenn das Beet voll ist, wird geprüft, ob diese Kombination alle Bedingungen erfüllt und eine Lösung für Lisas Problem ist. In der Informatik ist solch ein Ausprobieren aller Kombinationen als *Brute-Force-Methode* bekannt. Bei Problemen mit vielen Kombinationen und nur wenigen Lösungen kann ein Vorgehen nach dieser Methode sehr lange dauern.

Deshalb ist es meist besser, schrittweise vorzugehen und bei jedem Schritt alle Bedingungen zu berücksichtigen. Auf diese Weise haben wir die Lösung für Lisas Problem gefunden, und eine «falsche» Kombination bzw. Bepflanzung des Beets konnte gar nicht entstehen.



Zum Glück liess sich die Lösung auf direktem Weg finden: Es gab immer Bereiche, in die wir einige der noch übrigen Gemüse pflanzen konnten. Das gelingt im Allgemeinen nicht immer.

Wenn man versucht, die Lösung schrittweise zusammensetzen, kann es bei einem Schritt A mehrere Möglichkeiten geben, alle Bedingungen zu erfüllen.



Je nach Wahl kann es bei einem späteren Schritt B keine Möglichkeit mehr geben. Dann nimmt man die letzten Schritte solange zurück, bis man beim Schritt A mit den mehreren Möglichkeiten wieder angekommen ist. Dort wählt man eine andere Möglichkeit und versucht damit eine Lösung zu finden.

In der Informatik ist diese Rücknahme von Schritten als *Backtracking* bekannt.

## Stichwörter und Webseiten

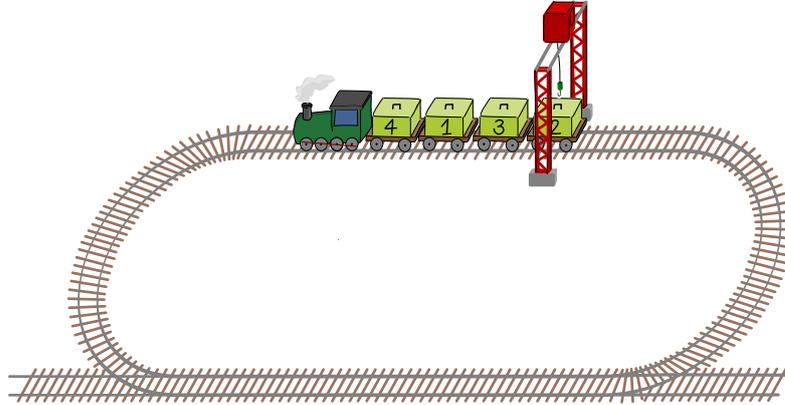
- Brute Force: <https://de.wikipedia.org/wiki/Brute-Force-Methode>
- Backtracking: <https://de.wikipedia.org/wiki/Backtracking>





## 2. Zug entladen

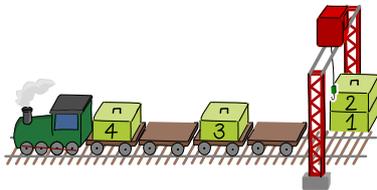
Ein Zug zieht Waggons mit nummerierten Kisten. Der Kran steht an einer festen Position und entlädt die Kisten. Um eine Kiste zu entladen, muss die Kiste direkt unter dem Kran positioniert werden.



Der Kran muss die Kisten von 1 an in der Reihenfolge ihrer Nummern entladen. Der Zug kann nur vorwärts fahren. Wenn er unter dem Kran hindurch gefahren ist, muss er eine Runde fahren, damit weitere Kisten entladen werden können.

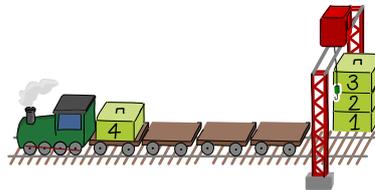
So entlädt der Kran die Kisten 1, 2, 3 und 4:

**Runde 1:**



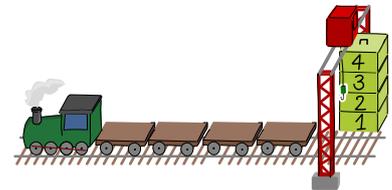
Er überspringt Kiste 4, entlädt Kiste 1, überspringt Kiste 3 und entlädt Kiste 2.

**Runde 2:**



Er überspringt Kiste 4 und entlädt Kiste 3.

**Runde 3:**



Er entlädt Kiste 4.

Der obige Zug muss also drei Runden fahren, bis alle Kisten in der richtigen Reihenfolge entladen sind.

Wie viele Runden werden für das Entladen des folgenden Zuges benötigt?



- |             |             |              |
|-------------|-------------|--------------|
| A) 1 Runde  | E) 5 Runden | I) 9 Runden  |
| B) 2 Runden | F) 6 Runden | J) 10 Runden |
| C) 3 Runden | G) 7 Runden |              |
| D) 4 Runden | H) 8 Runden |              |

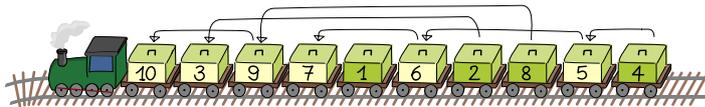


## Lösung

Die richtige Antwort ist 7 Runden.

Die vorgeschriebene Reihenfolge für das Entladen ist 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. In der ersten Runde entlädt der Kran die Kisten 1 und 2 zusammen. In der zweiten Runde entlädt der Kran 3 und 4 zusammen, dann 5, dann 6, dann 7 und 8 zusammen, dann 9 und schliesslich 10. Dies entspricht 7 Runden.

Alternativ kann man sich zunutze machen, dass jedes Mal, wenn für eine der Kistennummern in der Folge die nächste Kistennummer links davon steht, eine zusätzliche Entladerunde erforderlich ist.



Da z. B. die 3 links von der 2 steht, wird sie übersprungen, um die 2 zu entladen, so dass eine zusätzliche Runde erforderlich ist, um die 3 unter den Kran zu bringen. Beim gegebenen Zug gibt es sechs solcher Paare (2,3), (4,5), (5,6), (6,7), (8,9) und (9, 10), so dass 6 zusätzliche Runden benötigt werden, also insgesamt 7 Runden.

## Dies ist Informatik!

Wenn für eine beliebige Zahl in der Folge 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 die Kiste mit der nächstgrösseren Zahl weiter links auf dem Zug liegt, nennen wir dies eine *Inversion*, das bedeutet Umkehrung. Für jede solche Umkehrung ist eine zusätzliche Runde erforderlich. Wenn wir die Anzahl der Umkehrungen zählen, erhalten wir die Antwort.

Das Zählen von Inversionen in Bezug auf eine gewünschte Folge hat viele Anwendungen. Bei einigen *Sortieralgorithmen*, wie z. B. *Bubble-Sort*, gibt die Anzahl der Inversionen Aufschluss darüber, wie viele Vertauschungen erforderlich sind, um eine bestimmte Folge zu sortieren. Wenn zwei Kunden dieselbe Menge von Artikeln in eine Rangfolge bringen, sagt uns die Anzahl der Inversionen in ihren Ranglisten, wie sehr sich ihre Vorlieben gleichen. Dies wird von Online-Shops genutzt, um «ähnliche» Kunden zu identifizieren und ihnen Produktempfehlungen zu geben.

## Stichwörter und Webseiten

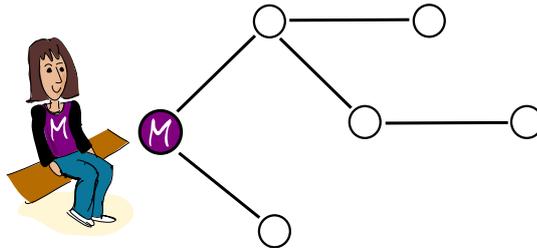
- Sortierverfahren: <https://de.wikipedia.org/wiki/Sortierverfahren>
- Bubble Sort: <https://de.wikipedia.org/wiki/Bubblesort>
- Einfache Sortierverfahren:  
<https://hpi.de/friedrich/teaching/units/einfache-sortierverfahren.html>
- Inversion: [https://www.ziemke-koeln.de/unterricht/informatik/gk12/sortieren/suchen\\_sortieren.htm#Inversion](https://www.ziemke-koeln.de/unterricht/informatik/gk12/sortieren/suchen_sortieren.htm#Inversion)



### 3. Martinas Dorf

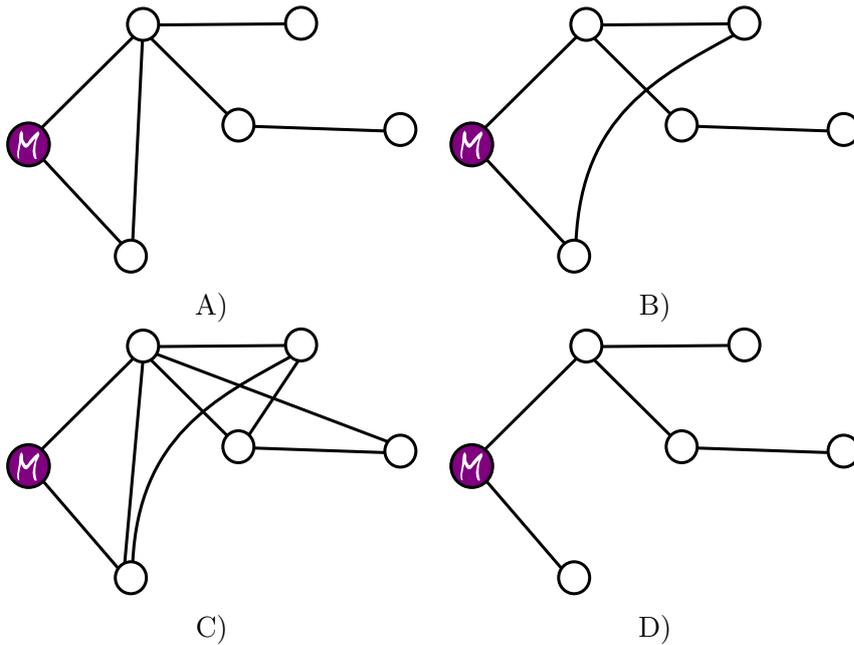
In Martinas Dorf gibt es sechs Häuser. Ausserdem gibt es Wege, über die man von einem Haus zum nächsten gehen kann. Für alle diese Wege benötigt Martina die gleiche Zeit.

Martina hat eine besondere Karte des Dorfs gezeichnet. Sie hat darin Wege eingezeichnet, über die sie am schnellsten zu den anderen Häusern gehen kann.



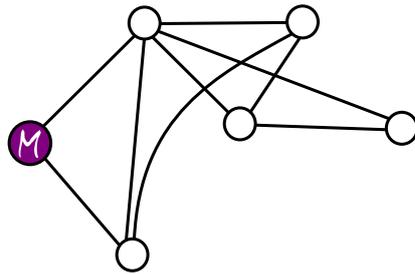
Natürlich gibt es auch eine richtige Karte des Dorfs, mit allen Wegen.

Welche dieser Zeichnungen kann nicht die richtige Karte sein?





## Lösung



Antwort C ist richtig:

Martinas besondere Karte zeigt, dass sie zu dem Haus ganz rechts am schnellsten über drei Wege gehen kann. Wenn C die richtige Karte des Dorfes wäre, dann könnte Martina schneller zu diesem Haus gehen, nämlich über zwei Wege. Also kann C nicht die richtige Karte des Dorfes sein.

Bei den Karten A, B und D gibt es keine Möglichkeit, schneller zu einem der anderen Häuser zu gehen als über die Wege auf Martina besonderer Karte. Diese Karten könnten also richtige Karten des Dorfs sein.

## Dies ist Informatik!

Martina ist Informatikerin. Sie hat ihre Karte als *Graph* gezeichnet. Graphen bestehen aus *Knoten* (hier die Häuser), die durch *Kanten* (hier die Wege) verbunden sein können. Sie sind in vielen Bereichen der Informatik geeignet, die Realität zu modellieren – auch hier in dieser Biberaufgabe.

Martina weiss, dass es für Graphen eine ganze Reihe von Algorithmen gibt, beispielsweise die sogenannte Breitensuche, um Aufgaben zu lösen wie «Wie kommt man am schnellsten zu einem anderen Haus?». Vielleicht hat sie ihre besondere Dorfkarte mit Hilfe einer Breitensuche in einem grösseren Graph, der richtigen Karte des Dorfes mit allen Wegen, erstellt.

In der Graphentheorie, die sich mit Graphen und Graph-Algorithmen beschäftigt, entspricht Martinas Karte einem Teilgraph der Gesamtkarte des Dorfes. Martinas Teilgraph hat zwei Besonderheiten:

- Alle Knoten sind direkt (über eine Kante) oder indirekt (über mehreren Kanten) miteinander verbunden.
- Egal welche zwei Knoten man zufällig auswählt, es gibt immer nur genau einen Weg zwischen den beiden.

Ein Graph mit diesen Besonderheiten wird in der Informatik als *Baum* bezeichnet. Martinas Haus stellt die *Wurzel* des Baumes dar. Von der Wurzel aus kann Martina alle anderen Knoten (die anderen Häuser im Dorf) auf einem eindeutigen Weg erreichen. Martinas Teilgraph ist also ein Baum; ausserdem enthält er alle Knoten des gesamten Graphen (der Gesamtkarte des Dorfes) – aber möglicherweise nicht alle Kanten. Ein Teilgraph mit diesen Eigenschaften wird als *Spannbaum* des gesamten Graphen bezeichnet.

In der Informatik gibt es viele Anwendungen für Graph-Algorithmen, vor allem im Zusammenhang mit Netzwerken (Verkehrsnetze, Telekommunikationsnetze ...), etwa bei der Berechnung von Routen



in Navigationssystemen. Spannbäume können beim Aufbau kostengünstiger Netze eingesetzt werden und hilfreich bei der Lösung besonders schwieriger Probleme sein.

## Stichwörter und Webseiten

- Graphentheorie: [https://de.wikipedia.org/wiki/Graph\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Graph_(Graphentheorie))
- Baum: [https://de.wikipedia.org/wiki/Baum\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Baum_(Graphentheorie))
- Breitensuche: <https://de.wikipedia.org/wiki/Breitensuche>
- Spannbaum: <https://de.wikipedia.org/wiki/Spannbaum>

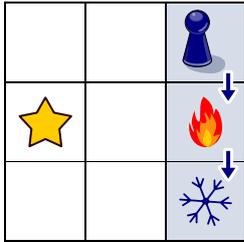




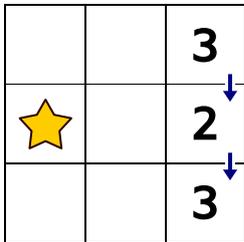
## 4. Wärmer und Kälter

Nina und Daniel spielen Schatzsuche. Auf einem Spielbrett mit quadratischen Feldern wählt Nina im Kopf ein Feld aus. Dort ist der Schatz versteckt.

Daniel wählt ein Startfeld aus. Von dort geht er schrittweise mit seiner Spielfigur  um je ein Feld weiter: nach links, rechts, oben oder unten.



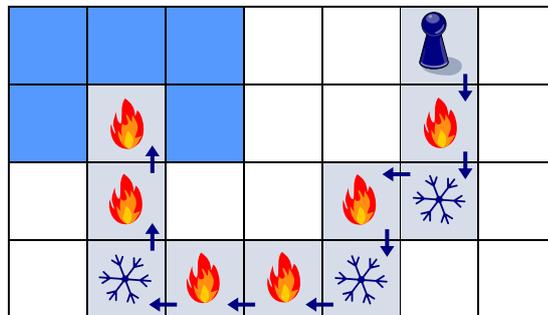
Beim ersten Versuch nehmen sie ein kleines Spielbrett. Nina versteckt den Schatz auf dem Feld mit dem Stern . Daniel startet rechts oben und macht zwei Schritte entlang der Pfeile. Nach jedem Schritt sagt Nina, ob Daniel nun näher  am Schatz oder weiter weg  vom Schatz ist als vor dem Schritt.



Das Bild rechts zeigt Daniels Entfernungen vom Schatz. Die Entfernung vom Schatz ist die kleinste Anzahl Schritte, mit denen Daniel aktuell zum Schatz gehen könnte.

Nun nehmen sie ein grösseres Spielbrett. Nina versteckt den Schatz auf einem der blau markierten Felder. Das Bild zeigt wieder Daniels Schritte und was Nina nach jedem Schritt sagt.

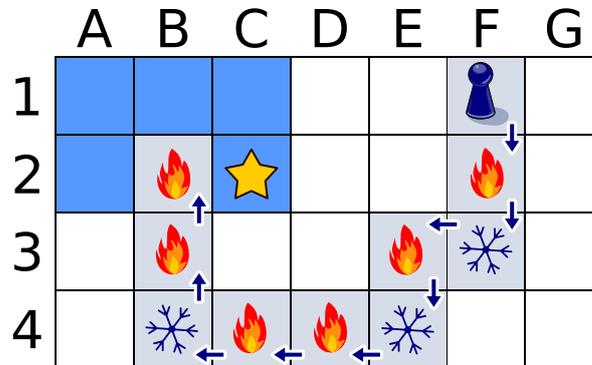
*Wo ist der Schatz versteckt?*





## Losung

So ist es richtig:



Wir verfolgen Daniels Weg und Ninas Ruckmeldungen. Daniel startet in Zeile 1 des Spielbretts. Nach dem ersten Schritt ist er in Zeile 2 und naher am Schatz als in Zeile 1. Nach dem nachsten Schritt ist er in Zeile 3 und wieder weiter weg vom Schatz. Da er in der gleichen Spalte geblieben ist, muss der Schatz auf einem Feld in Zeile 2 sein. Denn egal, in welcher Spalte der Schatz versteckt ist: Den kurzesten Weg zum Schatz von einer anderen Spalte aus hat man, wenn man in der gleichen Zeile ist.

Aber in welcher Spalte ist der Schatz versteckt? Auf seinem weiteren Weg kommt Daniel in Zeile 4 mit einigen Schritten nach links dem Schatz zunachst naher; insbesondere ist er in Spalte 3 naher am Schatz als in Spalte 4. Aber nach dem letzten Schritt in der Zeile ist Daniel in Spalte 2 weiter weg vom Schatz als in Spalte 3. Der Schatz muss also auf einem Feld in Spalte 3 sein. Denn was wir oben fur Spalten gesagt haben, gilt auch fur Zeilen: Den kurzesten Weg zum Schatz von einer anderen Zeile aus hat man, wenn man in der gleichen Spalte ist.

## Dies ist Informatik!

Daniel geht (mit seiner Spielfigur) uber das Spielbrett. Von jedem Feld, auf dem er gerade steht, misst Nina die Entfernung zum Feld mit dem Schatz und nutzt dies fur ihre Ruckmeldung. Ublicherweise wird die Entfernung zwischen zwei Punkten als Lange der geradlinigen Verbindung zwischen den Punkten gemessen (*euklidische Distanz*). Doch die zwei Felder sind genau genommen keine Punkte. Deshalb bemisst Nina die Entfernung zwischen zwei Feldern in der Anzahl von Schritten, die Daniel fur einen kurzesten Weg vom einen Feld zum anderen gehen musste. Dieses *Ma* kann man generell bei Rastern anwenden und ist in der Informatik als *Manhattan-Distanz* bekannt – abgeleitet vom gerasterten Straenplan des New Yorker Stadtteils Manhattan. Daniel geht (mit seiner Spielfigur) uber das Spielbrett. Von jedem Feld, auf dem er gerade steht, misst Nina die Entfernung zum Feld mit dem Schatz und nutzt dies fur ihre Ruckmeldung. Ublicherweise wird die Entfernung zwischen zwei Punkten als Lange der geradlinigen Verbindung zwischen den Punkten gemessen (*euklidische Distanz*). Doch die zwei Felder sind genau genommen keine Punkte. Deshalb bemisst Nina die Entfernung zwischen zwei Feldern in der Anzahl von Schritten, die Daniel fur einen kurzesten Weg vom einen Feld zum anderen gehen musste. Dieses *Mass* kann man generell bei Rastern anwenden



und ist in der Informatik als *Manhattan-Distanz* bekannt – abgeleitet vom gerasterten Strassenplan des New Yorker Stadtteils Manhattan.

Informatikerinnen und Informatiker wählen die Art der Distanzberechnung zwischen zwei Objekten in Abhängigkeit von der Frage, die sie lösen möchten. Wenn man zum Beispiel die Entfernung zwischen zwei gleich langen Wörtern in einer natürlichen Sprache messen möchte, kann man die Anzahl Stellen, an welchen sich die Wörter unterscheiden, zählen; das ist dann der *Hamming-Abstand* oder die *Hamming-Distanz*. Sind die Wörter unterschiedlich lang, kann man die *Editierdistanz* verwenden. Entfernungen bzw. Distanzen spielen in der Informatik häufig eine Rolle, wenn es darum geht, optimale Lösungen eines Problems zu finden. Egal ob die Lösung eines Problems am schnellsten, am kürzesten oder am günstigsten sein soll: Man muss oft nicht den Algorithmus ändern, sondern nur das Entfernungsmass: Dauer, Länge oder Kosten.

## Stichwörter und Webseiten

- Manhattan-Distanz: <https://de.wikipedia.org/wiki/Manhattan-Metrik>
- Hamming-Abstand: <https://de.wikipedia.org/wiki/Hamming-Abstand>
- Editierdistanz: <https://de.wikipedia.org/wiki/Levenshtein-Distanz>

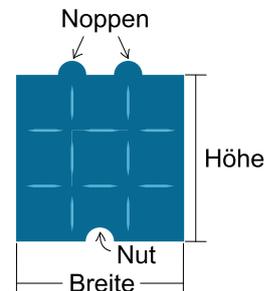




## 5. Biber-Bausteine

Die Biber-Bausteine unterscheiden sich in vier Eigenschaften:

1. Breite: schmal, mittel, breit
2. Höhe: klein, mittel, gross
3. Anzahl der Noppen oben: null, eins, zwei
4. Anzahl der Nuten unten: null, eins, zwei



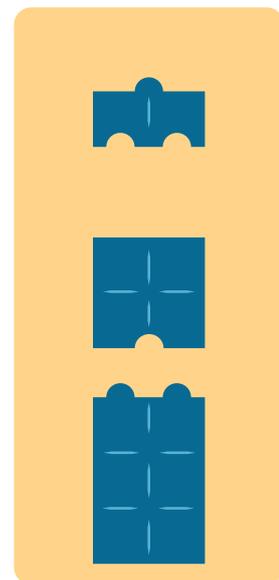
Otto teilt die Bausteine in Dreier-Gruppen ein. Er macht das so, dass für jede Gruppe gilt: Die drei Steine haben für jede der vier Eigenschaften ...

- ... entweder alle den gleichen Wert ...
- ... oder drei verschiedene Werte.

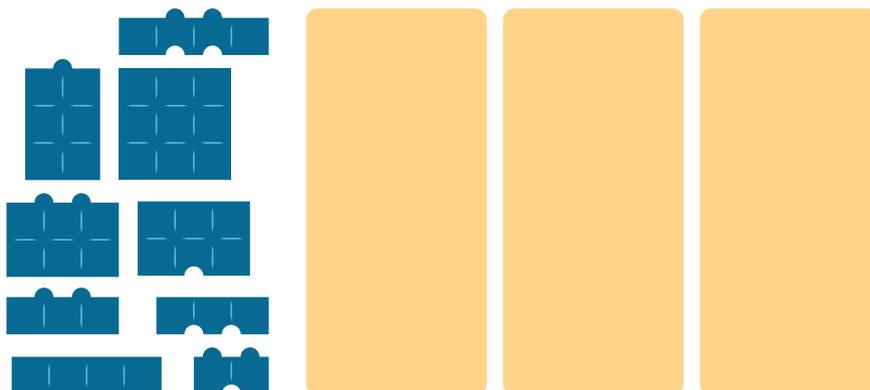
Rechts ist eine von Ottos Gruppen.

Denn diese drei Steine haben alle

- die gleiche Breite,
- unterschiedliche Höhen,
- unterschiedlich viele Noppen und
- unterschiedlich viele Nuten.



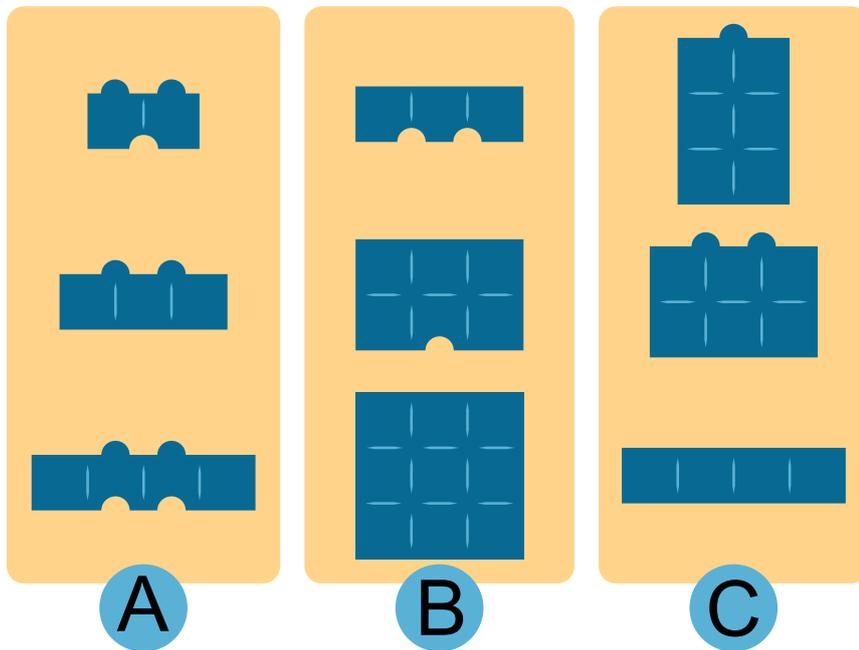
Teile diese Bausteine in Dreier-Gruppen ein, so wie Otto es machen würde.





## Lösung

So ist es richtig:



Die Steine sind so in Gruppen eingeteilt, wie Otto es machen würde. Die Tabelle zeigt für die drei Gruppen, bei welchen Eigenschaften die Werte alle unterschiedlich bzw. alle gleich sind.

Eigenschaft	Gruppe A	Gruppe B	Gruppe C
Breite	unterschiedlich	gleich	unterschiedlich
Höhe	gleich	unterschiedlich	unterschiedlich
Noppen	gleich	gleich	unterschiedlich
Nuten	unterschiedlich	unterschiedlich	gleich

Aber ist das die einzige Möglichkeit, die Steine so einzuteilen, wie Otto es machen würde?

Man kann überlegen: Wenn für eine Eigenschaft die Werte in allen Gruppen unterschiedlich sein sollen, müssen die verschiedenen Werte in allen Steinen genau so oft vorkommen, wie es Gruppen gibt. Ist das nicht der Fall, muss es mindestens eine Gruppe geben, in der die Werte für diese Eigenschaft alle gleich sind.

Ein genauer Blick auf alle Steine zeigt, dass bei der Breite die Werte schmal und breit jeweils nur zweimal vorkommen. Es muss also eine Gruppe geben, in der alle Steine die Breite mittel haben.

Von den fünf Steinen mit mittlerer Breite gibt es keinen mit nur einer Noppe; deshalb kann keine Gruppe mit unterschiedlich vielen Noppen gebildet werden. Es gibt aber drei Steine mit null Noppen – und sie haben alle unterschiedliche Höhen und unterschiedlich viele Nuten. Damit ist Gruppe B die einzig mögliche Gruppe von Steinen mit mittlerer Breite.

In den anderen beiden Gruppen müssen die Breiten alle unterschiedlich sein.



In den verbleibenden sechs Steinen kommen bei der Höhe die Werte gross und mittel nur noch einmal vor. Es muss also eine Gruppe geben, in der alle Steine die Höhe klein haben. Gruppe A ist die einzig mögliche Dreier-Gruppe von Steinen mit kleiner Höhe, die Ottos Vorstellungen entspricht. Damit bleiben die drei Steine in Gruppe C übrig. Sie bilden ebenfalls eine Dreier-Gruppe, wie Otto sie auch bilden würde.

## Dies ist Informatik!

In dieser Biberaufgabe werden die Biber-Bausteine mit Hilfe von vier *Eigenschaften* (oder *Attributen*) beschrieben.

Um die Bausteine wie Otto in Dreier-Gruppen aufteilen zu können, muss man für jeden Stein die Werte der Eigenschaften kennen.

Dazu genügt Menschen ein Blick auf jeden Baustein. Ein Computerprogramm, das die Dreier-Gruppen zusammenstellen soll, kann in der Regel nicht sehen und braucht eine Beschreibung in einer *Datenstruktur*.

Zum Beispiel kann man die Steine in einer *Datenbank* als Zeilen einer Tabelle beschreiben. Die Spalten der Tabelle entsprechen den Eigenschaften, und in jeder Zeile (auch *Datensatz* genannt) stehen die Werte eines Bausteins in den passenden Spalten:

Stein-Nr	Breite	Höhe	Noppen	Nuten
1	schmal	hoch	1	0
2	mittel	mittel	2	0
...	...	...	...	...

Der Entwurf von Datenbank-Tabellen gehört zu den üblichen Tätigkeiten für Informatikerinnen und Informatiker.

Sie müssen dabei gründlich vorgehen und überlegen, welche Eigenschaften von Objekten für die Verarbeitung durch ein Computerprogramm wichtig sind. Nachträgliche Änderungen sind nicht so einfach, insbesondere wenn schon Daten vieler Objekte gespeichert sind.

## Stichwörter und Webseiten

- Datenstruktur: <https://de.wikipedia.org/wiki/Datenstruktur>
- Datenbanken: <https://de.wikipedia.org/wiki/Datenbanktabelle>



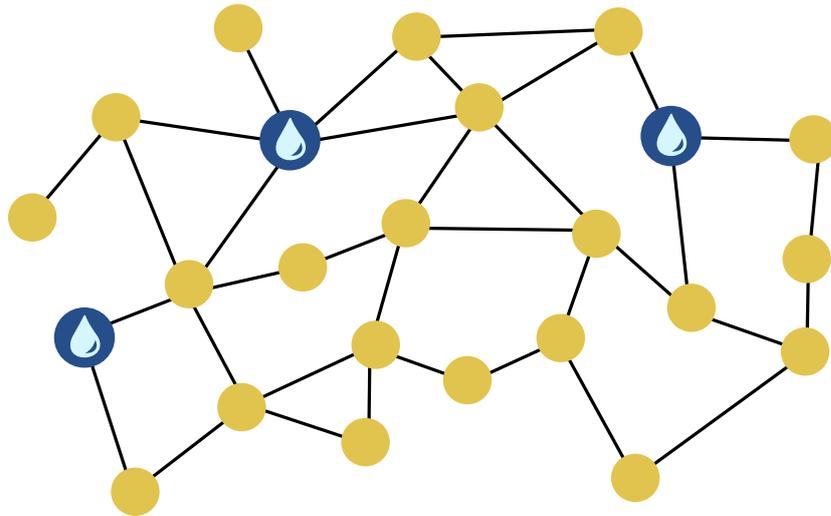


## 6. Brunnen

Der Sommer in der Stadt ist heiss. Die Bürgermeisterin lässt deshalb Brunnen mit Trinkwasser aufstellen.

Die Brunnen sollen so stehen, dass man von jeder Strassenecke aus höchstens zwei Strassenabschnitte gehen muss, um einen Brunnen zu erreichen. Dann ist die Bürgermeisterin zufrieden.

Hier ist ein Stadtplan. Die Linien sind Strassenabschnitte, und die Punkte sind Strassenecken. An drei Ecken stehen bereits Brunnen .

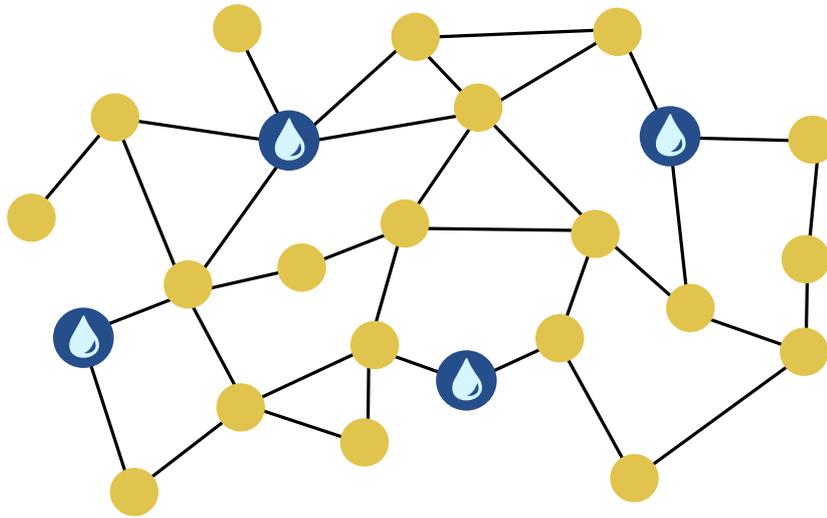


*Stelle einen weiteren Brunnen so auf, dass die Bürgermeisterin zufrieden ist.*



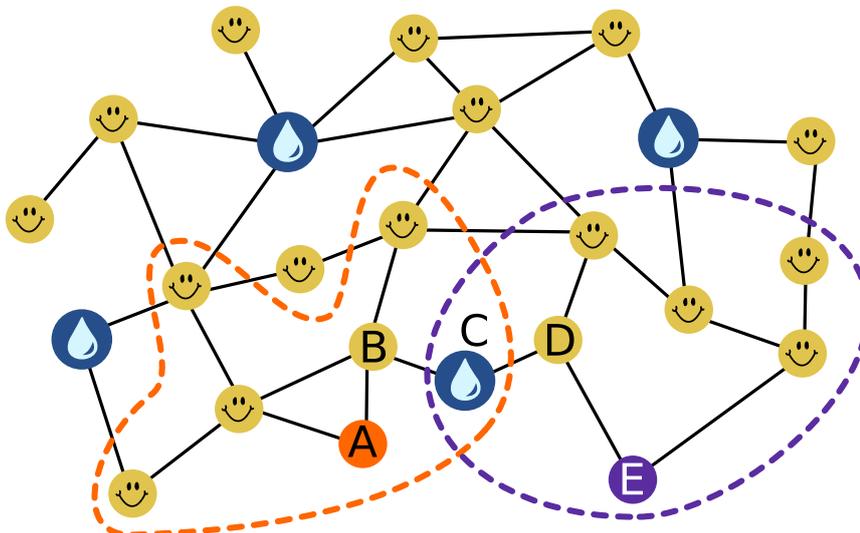
## Lösung

So ist es richtig:



Wenn ein weiterer Brunnen unten in der Mitte aufgestellt wird, muss man von jeder Strassenecke aus höchstens zwei Strassenabschnitte gehen, um einen Brunnen zu erreichen. Dann ist die Bürgermeisterin zufrieden.

Wie können wir herausfinden, an welcher Strassenecke ein weiterer Brunnen aufgestellt werden soll? Im Stadtplan markieren wir alle Strassenecken mit einem 😊, die höchstens zwei Strassenabschnitte von einem der Brunnen entfernt sind, die bereits aufgestellt sind. In Bezug auf diese Ecken kann die Bürgermeisterin bereits zufrieden sein.



Für die fünf übrigen Strassenecken A, B, C, D und E stellen wir einen weiteren Brunnen bei C auf. Damit muss man auch von diesen Ecken höchstens zwei Strassenabschnitte zum nächsten Brunnen gehen.



Die Ecke C ist die einzige Stelle für einen neuen Brunnen, die das ermöglicht: Wenn wir für die Ecken A und E jeweils alle anderen Ecken betrachten, die über zwei Strassenabschnitte erreichbar sind (im Bild mit gestrichelten Linien umrandet), ist die Strassenecke C die einzige, die diese Bedingung für A und E erfüllt.

## Dies ist Informatik!

Der Stadtplan kann als *Graph* modelliert werden. Das ist ein für die Informatik wichtiges Werkzeug, um Beziehungen zwischen Objekten zu modellieren und Fragen in Bezug auf diese Beziehungen zu beantworten. Hier kann man die Strassenecken als Objekte und damit *Knoten* des Graphen auffassen. Die Beziehung zwischen zwei Objekten wird im Graph durch *Kanten* modelliert, die man als Verbindungslinien darstellt. Hier bedeutet eine Kante zwischen zwei Strassenecken, dass sie durch einen Strassenabschnitt verbunden sind. Diese Beziehung kann man Nachbarschaft nennen. Kanten können aber auch andere Beziehungen modellieren, wie z.B. Freundschaft.

In dieser Biberaufgabe soll eine Teilmenge der Knoten gefunden werden (zum Aufstellen der Brunnen), so dass jeder Knoten ausserhalb dieser Teilmenge über einen Weg mit einem «Brunnen-Knoten» verbunden ist, der höchstens zwei Kanten lang ist. In der Fachsprache der Informatik würde dies als Suche nach einem «distance-2 dominating set» bezeichnet. Im allgemeinen (für alle Weglängen  $k \geq 1$ ) gehört diese Suche nach einer möglichst kleinen solchen Teilmenge zu den schwierigsten Problemen der Informatik.

Solche «minimum distance  $k$ -dominating sets» spielen in der letzten Zeit eine grössere Rolle, insbesondere im Bereich des *Social Computing* (auf Deutsch auch *Sozioinformatik*): Zur automatischen Verarbeitung von Daten über soziale Netzwerke (etwa um die Verbreitung von Fake News zu erkennen) werden die Fan- oder Follower-Beziehungen zwischen den Nutzern als Graph modelliert. Diese Graphen können so gross sein, dass nur eine (möglichst kleine) repräsentative Auswahl von Nutzern betrachtet werden kann - zum Beispiel ein «minimum distance 3-dominating set». Da die wirklich kleinste Auswahl nicht effizient berechnet werden kann, entwickelt die Informatik Verfahren, die in kurzer Zeit möglichst kleine, aber nicht garantiert kleinste Auswahlen berechnen.

## Stichwörter und Webseiten

- Minimum Distance  $k$ -Dominating Sets:  
<https://computationalsocialnetworks.springeropen.com/articles/10.1186/s40649-020-00078-5>
- Sozioinformatik: <https://de.wikipedia.org/wiki/Sozioinformatik>





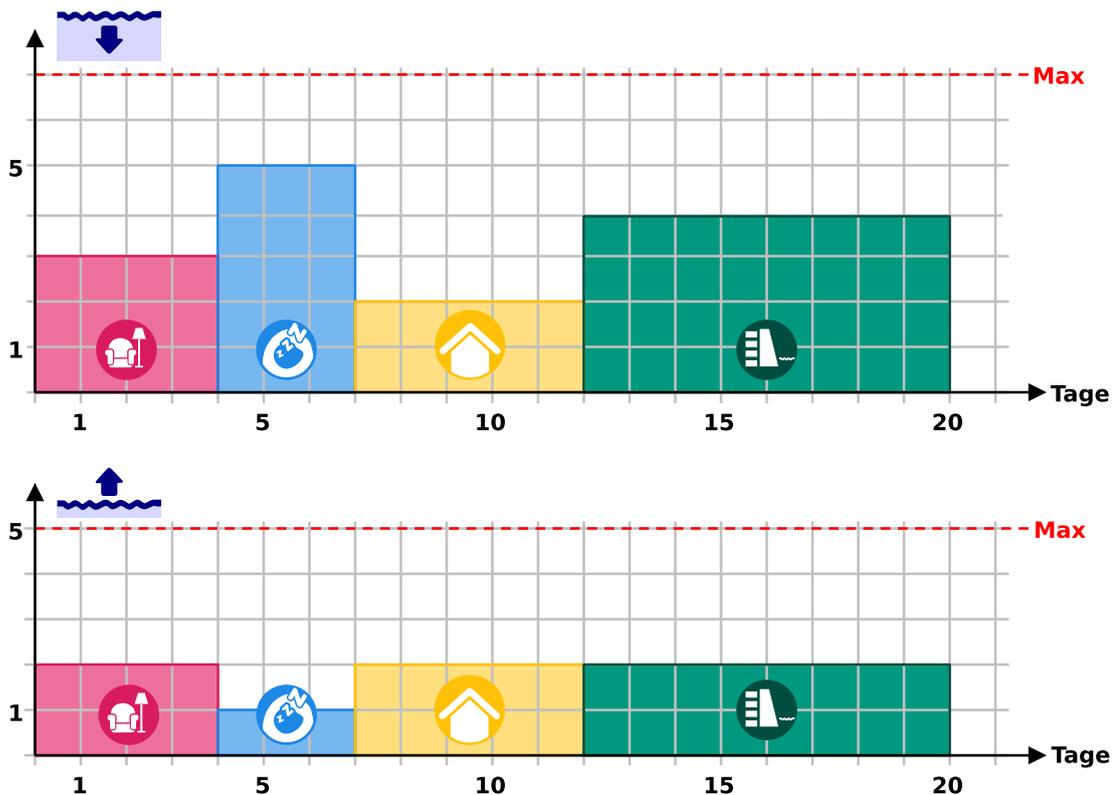
## 7. Biberburg AG

Eine Biberburg besteht aus 4 Teilen, die alle teilweise unter und teilweise über Wasser liegen. Beim Bau einer Biberburg ist jeder beteiligte Arbeiter entweder nur unter Wasser oder nur über Wasser tätig. Bei jedem Teil wird gleichzeitig über und unter Wasser gearbeitet. Die Tabelle zeigt für jedes Teil, wie lange die Biberbau AG braucht und wie viele Arbeiter unter und über Wasser dafür benötigt werden.

Teile	Wohnraum	Schlafhöhle	Dach	Damm
Baudauer	4 Tage	3 Tage	5 Tage	8 Tage
	3 Biber	5 Biber	2 Biber	4 Biber
	2 Biber	1 Biber	2 Biber	2 Biber

Das Dach kann erst gebaut werden, wenn die Schlafhöhle fertig ist! Bei allen anderen Teilen ist die Reihenfolge egal.

Für den Bau einer neuen Burg stehen höchstens 7 Unterwasser-Arbeiter und 5 Überwasser-Arbeiter zur Verfügung. Sie können auch gleichzeitig verschiedene Teile bauen. Hier ist ein Arbeitsplan, mit dem die Biberburg in 20 Tagen fertig wird.



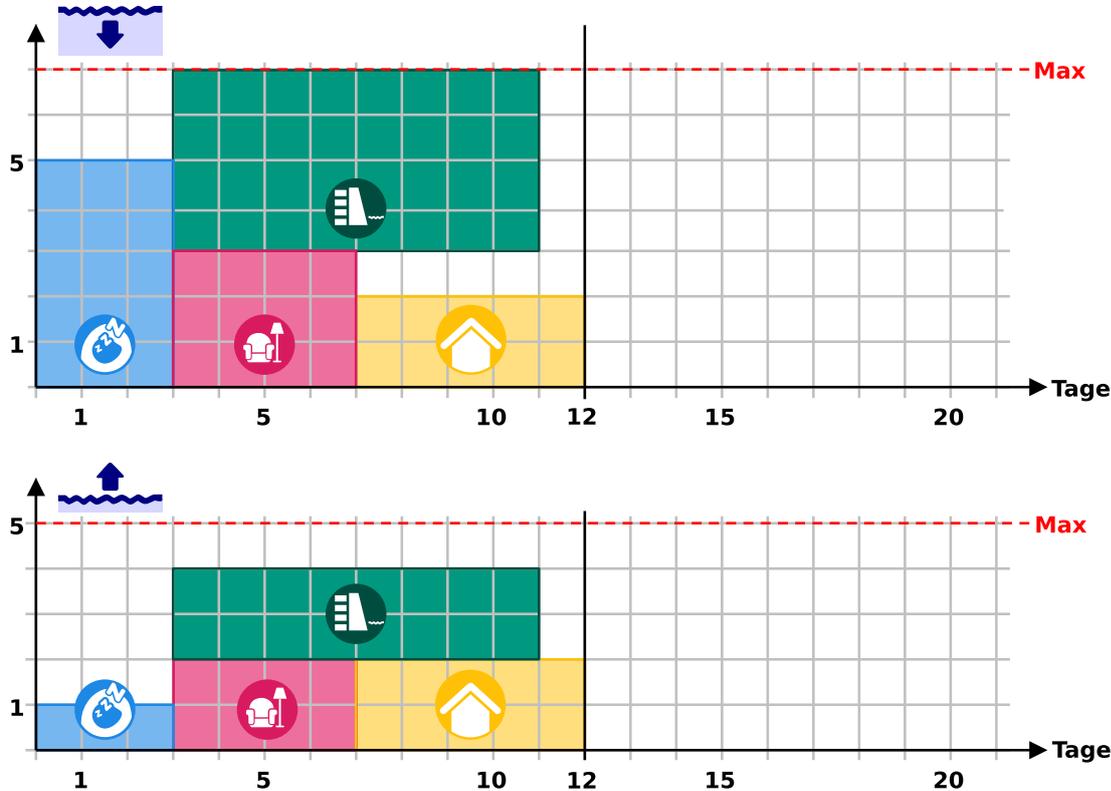
Überlege dir einen Plan, mit dem die Biberburg nach möglichst wenigen Tagen fertig wird. Wie viele Tage sind das?



## Lösung

12 Tage ist die richtige Antwort.

Dies ist ein Plan, mit dem die Biberburg in 12 Tagen fertig wird:



Einen solchen Plan mit der kürzesten Bauzeit kann man in zwei Schritten bestimmen:

1. Zuerst muss die Schlafhöhle vor dem Dach eingeplant werden. Da die Schlafhöhle 5 Unterwasser-Arbeiter benötigt, der Damm 3 und der Wohnraum 4, kann die Schlafhöhle – bei der Beschränkung auf 7 Unterwasser-Arbeiter – auch nicht gleichzeitig mit Damm oder Wohnraum gebaut werden. Die Schlafhöhle muss also zuerst gebaut werden und alle drei anderen Teile danach.
2. Damm und Wohnraum können gleichzeitig nach der Schlafhöhle gebaut werden, oder eines der beiden Teile gleichzeitig mit dem Dach. Es ist aber nicht möglich, alle drei Teile gleichzeitig zu bauen, weil sie zusammen  $3 + 4 + 2 = 9$  Unterwasser-Arbeiter benötigen – mehr als zur Verfügung stehen. Die kürzeste Bauzeit kann erzielt werden, wenn die beiden Teile mit den kürzeren Bauzeiten (Dach und Wohnraum) hintereinander und der Damm gleichzeitig zu diesen gebaut werden.

## Dies ist Informatik!

Einen optimalen, möglichst zügigen Ablauf eines Projekts zu planen, ist eine schwierige Aufgabe, bei der einige Bedingungen zu berücksichtigen sind. Zwischen Teilaufgaben eines Projekts bestehen oft zeitliche Abhängigkeiten; z.B. kann es Teilaufgaben geben die erst nach Beendigung einer anderen



Teilaufgabe begonnen werden können – wie hier bei Dach und Schlafhöhle. Ausserdem braucht jede Teilaufgabe bestimmte Ressourcen wie Arbeitskraft, Zeit und Geräte. Bei der Erstellung von Projektplänen hilft es, wenn man den Plan gut darstellen kann. Die in dieser Biberaufgabe gezeigten Diagramme sind eine Art von Gantt-Diagrammen, die von Henry Gantt (1861–1919) zwischen 1910 und 1915 entwickelt wurden; ähnliche Darstellungen wurden unabhängig von Gantt zur gleichen Zeit auch in Deutschland verwendet. Sie zeigen die Nutzung von Ressourcen (in diesem Fall die beiden Arten von Arbeitskräften) im Zeitverlauf.

Den optimalen Plan für die Biberburg kann man sich im Kopf überlegen und dabei alle erlaubten Möglichkeiten ausprobieren. Bei grösseren Projekten würde das zu lange dauern und zu unübersichtlich werden. Hier können Computerprogramme helfen, und deshalb ist die Erstellung von Zeitplänen (engl. *Scheduling*) ein wichtiges Thema der Informatik. Wie häufig bei schwierigen Problemen wurden Verfahren entwickelt, die statt eines garantiert optimalen Plans einen Plan mit etwas grösserem, aber immer noch sehr gutem Zeitbedarf erstellen. Scheduling wird auch bei der Steuerung von Computern selbst angewandt, deren Prozesse um Ressourcen (Rechenleistung, Speicherzugriff, Zugriff auf externe Geräte wie Speichergeräte, Drucker oder Netzwerkschnittstellen) konkurrieren.

## Stichwörter und Webseiten

- Schedule: <https://de.wikipedia.org/wiki/Scheduling>
- Gantt-Diagramm: <https://de.wikipedia.org/wiki/Gantt-Diagramm>
- Software für Projektmanagement:  
<https://de.wikipedia.org/wiki/Projektmanagement-Software>



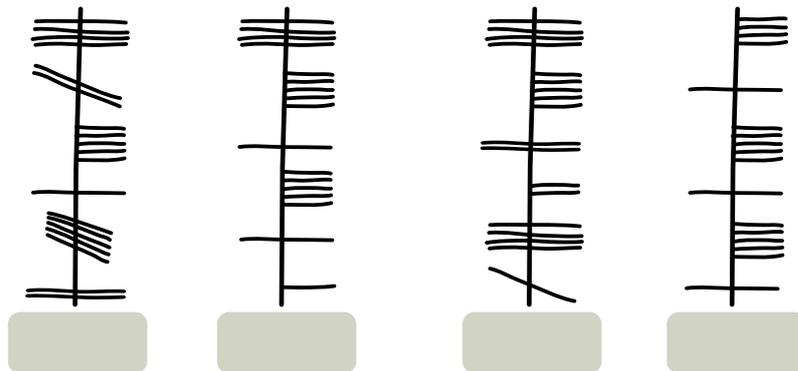


## 8. Ogham

Sue kennt das alte irische Alphabet Ogham. Jeder Buchstabe besteht aus einem oder mehreren Strichen, die entlang einer langen Linie angeordnet sind. Zwei aufeinander folgende Buchstaben werden durch einen Zwischenraum getrennt.

Sue benutzt Ogham als Code. Sie kodiert vier Wörter – ihre liebsten Fruchtsorten: ANANAS, BANANE, MELONE und ORANGE.

*Welches Wort passt zu welchem Ogham-Code?*



ANANAS

BANANE

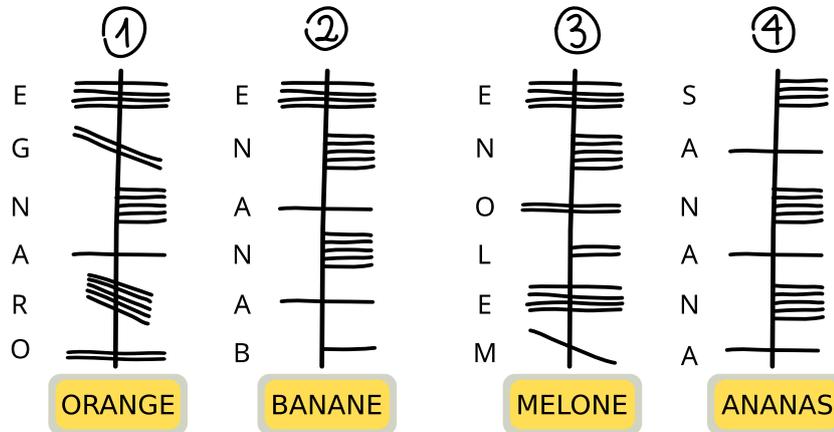
MELONE

ORANGE



## Lösung

So ist es richtig:



Es gibt verschiedene Möglichkeiten, die richtige Zuordnung zu bestimmen. Auf jeden Fall aber muss herausgefunden werden, in welcher Richtung die Buchstaben entlang der Linie geschrieben werden. Dabei hilft das besonders markante Wort ANANAS. Darin kommt der Buchstabe A dreimal vor, mit jeweils einem anderen Buchstaben dazwischen.

Nur im Ogham-Code 4 kommt ein Buchstabe dreimal vor, und auch dort ist jeweils ein Buchstabe dazwischen. Code 4 ist also der einzige, zu dem das Wort ANANAS passt. So erkennt man, dass man in Ogham Wörter von unten nach oben schreibt und dass der dreifach vorkommende Buchstabe A in Ogham als horizontaler Strich durch die Linie geschrieben wird.

Dieser Ogham-Buchstabe A kommt nur im Code 2 zweimal vor. Auch wegen der aus ANANAS bekannten Kodierung von N (fünf horizontale Striche rechts von der Linie) und der Anordnung der weiteren Buchstaben passt nur BANANE zu diesem Code. ORANGE passt nur zum Code 1; unter anderem, weil man dort den Ogham-Buchstaben A genau einmal findet. Nun ist nur noch Code 3 übrig; er muss also das Ogham-Wort für MELONE sein und enthält die von den übrigen Wörtern bekannten Ogham-Buchstaben E und N an den passenden Stellen.

## Dies ist Informatik!

In dieser Biberaufgabe muss ein unbekannter Text entschlüsselt bzw. dechiffriert werden. Das ist hier nicht sehr schwierig, weil der entschlüsselte *Klartext* bekannt ist. Ausserdem ist der unbekannte Text auf gleiche Weise in Buchstaben und Wörter eingeteilt wie der bekannte Text. Beim Dechiffrieren eines geheimen Textes bzw. eines Textes in unbekannter Schrift, dessen Klartext nicht bekannt ist, hilft es in diesem Fall oft, sich Gedanken über die Häufigkeit von Buchstaben und Wörtern zu machen und auf dieser Grundlage zu versuchen, sie im Text zu finden. Auf diese Weise sind einige antike Alphabete und Schriften entschlüsselt worden. Schwierig wird es aber, wenn die Schriftzeichen im unbekanntem Text nicht so einfach den Buchstaben und Wörtern der bekannten Sprache zuzuordnen sind wie im Fall von Ogham. Dann hilft oft nur der Abgleich mit bekannten Texten oder Schriften, wie in dieser Biberaufgabe. Zum Beispiel wurden die ägyptischen Hieroglyphen jahrhundertlang



nicht entschlüsselt, bis durch einen Zufall ein Stein mit Hieroglyphen und zwei bekannten Schriften gefunden wurde, der Stein von Rosetta. Auf dem Stein fand sich dreimal der gleiche Text. Der war zwar in verschiedenen Sprachen geschrieben, enthielt aber immer dieselben Namen. So konnten wesentliche Elemente der Hieroglyphen entschlüsselt werden. Das gilt aber nicht für alle Schriften: Noch immer sind die etwa 650 Schriftzeichen der Maya-Kultur nicht vollständig entschlüsselt, genau so wenig wie die Schriften Linearschrift A und Linearschrift B aus der Mittelmeerregion.

Auch in der Informatik werden Schriftzeichen und Texte entschlüsselt – nachdem sie vorher zur abhörsicheren Datenübertragung verschlüsselt wurden. Dazu werden aber ganz andere Verfahren verwendet als bei der Kodierung von Wörtern in anderen Schriften. Solche einfachen Kodierungen sind insbesondere mit Hilfe von Computern zu leicht zu entschlüsseln, meist mit Hilfe der oben schon genannten Überlegungen zur Häufigkeit von Buchstaben und Wörtern.

## Stichwörter und Webseiten

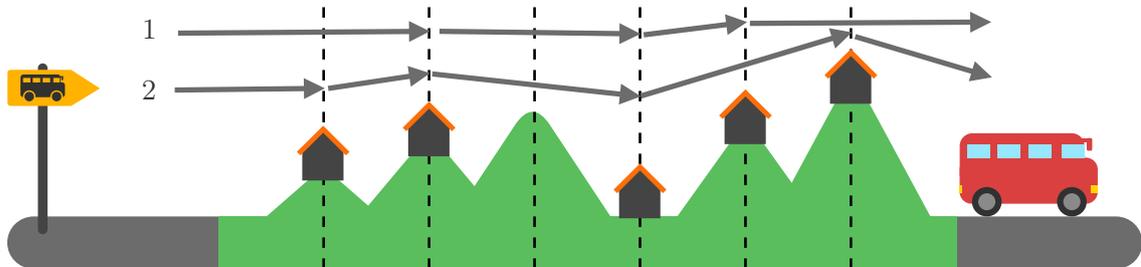
- Kryptographie: <https://de.wikipedia.org/wiki/Kryptographie>
- Kryptoanalyse: <https://de.wikipedia.org/wiki/Kryptoanalyse>
- Ogham: <https://de.wikipedia.org/wiki/Ogham>





## 9. Wanderungen

Mia mag Wanderurlaube, bei denen sie jede Nacht an einem anderen Ort übernachtet. Für ihren nächsten Urlaub hat Mia eine Karte der Region. Die Karte zeigt Mias Startpunkt 🚌, ihr Ziel 🚐 und alle Orte, an denen sie übernachten kann 🏠.



Mia hat die Region mit gestrichelten Linien in Abschnitte eingeteilt. Sie kann immer nur einen oder zwei Abschnitte an einem Tag wandern. Zwei verschiedene Wanderungen, die sie machen kann, hat sie bereits in die Karte eingetragen:

- Wanderung 1 hat drei Übernachtungsorte
- Wanderung 2 hat vier Übernachtungsorte

Mia kann aber noch andere Wanderungen machen.

Wie viele verschiedene Wanderungen kann Mia insgesamt machen? Zähle die Wanderungen 1 und 2 mit.

- A) 2 Wanderungen
- B) 3 Wanderungen
- C) 4 Wanderungen
- D) 5 Wanderungen
- E) 6 Wanderungen
- F) 7 Wanderungen
- G) 8 Wanderungen



## Lösung

Die richtige Antwort ist E) 6 Wanderungen.



Zuerst stellen wir fest, dass Mia in **B** und **C** übernachten muss, weil die Entfernung zwischen diesen beiden Orten die grösste Entfernung (2), die sie an einem einzigen Tag zurücklegen kann. Für den Weg von **B** nach **C** hat Mia also nur eine Möglichkeit.

Nun können wir die Möglichkeiten für die anderen Teilstücke ihres Weges ermitteln: Vom Startpunkt () bis **B** kann Mia entweder in einem Stück durchwandern oder zwischendurch in **A** übernachten; das sind zwei Möglichkeiten (wie in den Wanderungen 1 und 2). Von **C** zum Ziel ( $\cong$  ) muss Mia drei Abschnitte wandern, und sie kann nach jedem Abschnitt übernachten. Deshalb kann sie den gesamten Weg in alle drei Kombinationen von 1 und 2 Abschnitten aufteilen:

- $C \rightarrow D \rightarrow E \rightarrow \cong$  ;
- $C \rightarrow E \rightarrow \cong$  ;
- $C \rightarrow D \rightarrow \cong$  .

Die Gesamtzahl aller Wanderungen, die Mia machen kann, ist also  $2 \times 1 \times 3 = 6$ .

## Dies ist Informatik!

Manchmal kann die Zahl aller Möglichkeiten, eine gegebene Aufgabe zu erledigen, sehr gross sein. Zum Beispiel gibt es etwa 14 Millionen Möglichkeiten, 6 verschiedene Zahlen aus den Zahlen 1 bis 49 auszuwählen. Und es gibt etwa eine halbe Milliarde Möglichkeiten, die Zahlen von 1 bis 12 in unterschiedlicher Folge aufzuschreiben. Dafür braucht dann auch ein Computer ein wenig Zeit.

Wie gut, dass es in dieser Biberaufgabe nach dem dritten Abschnitt keinen Übernachtungsort gibt und das Zählen aller Wanderungen, die Mia machen kann, in drei Teile aufgeteilt werden kann. Das Zählproblem wird sozusagen in drei kleinere Zählprobleme zerlegt. In der Informatik wird die Technik der *Problemzerlegung* (engl.: *decomposition*) beim Entwurf von Algorithmen häufig verwendet. Dieses Lösungsprinzip ist auch als *Divide and Conquer* (auf Deutsch auch «Teile und herrsche») bekannt.

Nach diesem Prinzip funktionieren zum Beispiel einige wichtige Sortieralgorithmen. Auch die *dynamische Programmierung*, eine Methode zur algorithmischen Lösung von von Optimierungsproblemen (beschrieben 1957 von Richard Bellman), basiert auf diesem Prinzip: Wenn man erkennt, dass die optimalen Lösungen eines Problems sich aus den optimalen Lösungen von Teilproblemen zusammensetzen, kann man dies nutzen, um sozusagen «klein anzufangen»: Zunächst werden die Lösungen für die kleinsten Teilprobleme direkt berechnet und anschliessend zu Lösungen für die nächstgrösseren



Teilprobleme zusammengesetzt. Dies wird wiederholt, bis die optimale Lösung des vollständigen Problems gefunden ist. Da gefundene Teil-Lösungen häufig zu Lösungen vieler grösserer Teile beitragen, werden sie gespeichert, um wiederholte gleiche Berechnungen einzusparen. Auch beim Zählen von Möglichkeiten kann dynamische Programmierung sehr hilfreich sein.

## Stichwörter und Webseiten

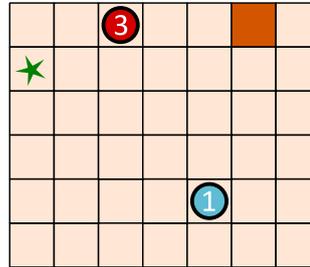
- Problemzerlegung, Decomposition
- Divide and Conquer / Teile und herrsche:  
<https://de.wikipedia.org/wiki/Teile-und-herrsche-Verfahren>
- dynamische Programmierung:  
[https://de.wikipedia.org/wiki/Dynamische\\_Programmierung](https://de.wikipedia.org/wiki/Dynamische_Programmierung)





# 10. Go-Bots

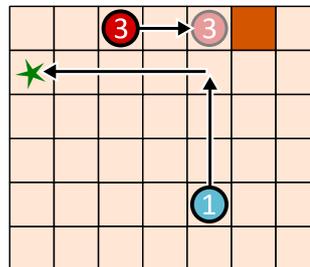
Die Go-Bots sind sehr einfache Roboter. Sie fahren über ein Spielbrett mit Feldern.



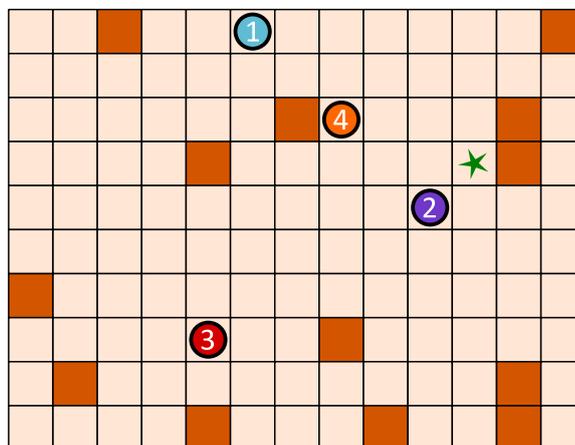
Um sie zu steuern, wählt man zunächst einen der Go-Bots aus. Den schickt man dann mit einem Pfeil-Befehl in eine Richtung: hoch , runter , links und rechts . Der Go-Bot fährt dann stur geradeaus, bis er direkt vor einem Hindernis oder einem anderen Roboter ankommt. Dort bleibt er stehen, bis er einen neuen Befehl bekommt.

Mit einer geschickten Folge von Befehlen sollst du dafür sorgen, dass Go-Bot **1** das Ziel erreicht, also genau dort stehen bleibt.

Mit dieser Befehlsfolge erreicht Go-Bot **1** das Ziel .



*Erstelle eine Befehlsfolge mit vier Pfeilen, mit der Go-Bot **1** das Ziel erreicht!*

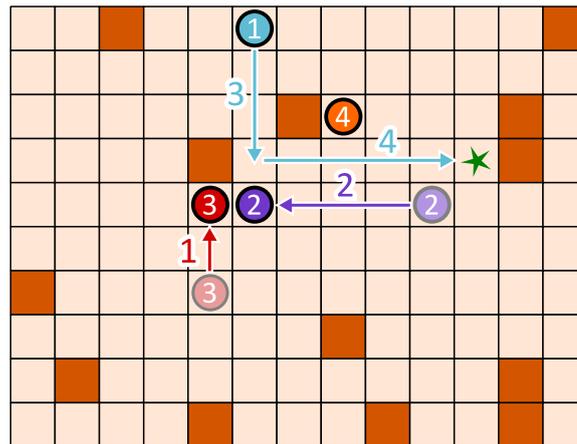




## Lösung

So ist es richtig:

Damit Go-Bot durch eine Befehlsfolge mit vier Pfeilen das Ziel erreichen kann, müssen drei Go-Bots kooperieren. Zuerst geht nach oben, bis er vor einem Hindernis stehen bleibt. Damit wird er selbst zum Hindernis für auf seinem Weg nach links. Wenn man nun nach unten schickt, geht er bis und kann von dort aus nach rechts gehen, wo er vor dem Hindernis stehen bleibt - auf dem Ziel.



Wie findet man die richtige Befehlsfolge? Man kann hinten anfangen und sich überlegen, was die letzte Bewegung von Go-Bot zum Ziel sein muss. Es gibt nur zwei Möglichkeiten:

- a) Er kommt von links, wie in unserer Lösung.
- b) Er kommt von oben. In diesem Fall müsste Go-Bot mit drei Befehlen nach oben rechts bewegt werden, um für als Hindernis zu dienen. Wir benötigen dann  $3 + 2 = 5$  Pfeil-Befehle. Gesucht ist aber eine Folge mit vier Befehlen. Also muss Möglichkeit a) korrekt sein, von Go-Bot kommt von links zum Ziel. Dann geht die vorletzte Bewegung des Go-Bots von oben nach unten. Damit er an der richtigen Stelle stehen bleibt, müssen zuvor und wie im Bild bewegt werden.

## Dies ist Informatik!

In dieser Biberaufgabe haben mehrere Roboter zusammen gearbeitet, um gemeinsam ein Ziel zu erreichen. Dabei hatten sie unterschiedliche Aufgaben. Der blaue Roboter musste zum Ziel kommen, und die anderen dienten als Hindernisse.

Aufgabenverteilung ist ein wichtiger Aspekt der Robotik. Zum Beispiel arbeiten in einem automatisierten Warenlager unterschiedliche Roboter zusammen, um Waren einzulagern, wiederzufinden und zu transportieren. Dabei werden alle Aktivitäten so koordiniert, dass möglichst wenig nutzlose Ruhezeiten entstehen, alle Transportwege möglichst kurz sind, wenig Energie verbraucht wird und so insgesamt das Warenlager möglichst effizient arbeitet.

Ein besonderes Gebiet der Robotik sind Schwarmroboter. Das sind – wie die Go-Bots – einfache Maschinen, die in einer grossen Gruppe gemeinsam eine Aufgabe lösen. In der Landwirtschaft können



inzwischen Schwärme von Robotern die Aussaat von Mais erledigen, die Entwicklung der Pflanzen und die Bodenbeschaffenheit beobachten und schliesslich sogar das Getreide ernten. Jeder Schwarmroboter ist klein und einfach konstruiert, aber der Schwarm als Ganzes kann Grosses leisten. Dieses Prinzip gilt auch für Multiagentensysteme: Das sind einfache Softwareeinheiten, die gemeinsam komplexe Probleme lösen können. Die Aufgabe der Informatik ist, Algorithmen für eine optimale Koordination und Kooperation von Gesamtsystemen mit mehreren Akteuren – ob Hardware oder Software – zu entwickeln.

## Stichwörter und Webseiten

- Multi-Robotersysteme: <https://www.hsu-hh.de/rt/forschung/multirobotersysteme>
- Robotik: <https://www.infineon.com/cms/de/discoveries/grundlagen-robotics/>
- Schwarmroboter
- Industrieroboter: <https://de.wikipedia.org/wiki/industrieroboter>
- Multiagentensystem: <https://de.wikipedia.org/wiki/Multiagentensystem>





# 11. Emma erledigt

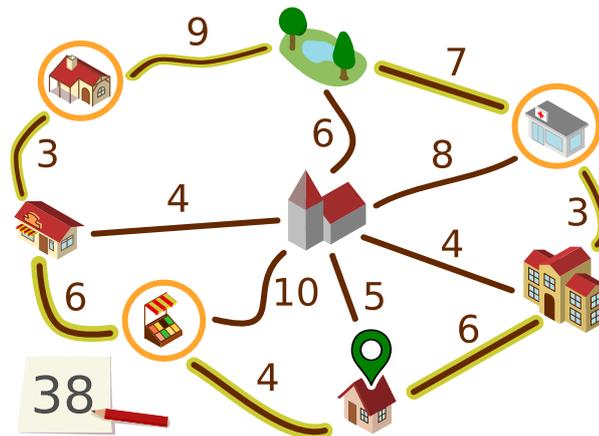
Emma ist zu Hause . Sie soll drei Aufgaben erledigen und zurückkommen:

- beim Kiosk ein Päckchen abholen,
- auf dem Markt Obst kaufen und
- in der Apotheke ein Medikament besorgen.

Emma weiss nicht, wie lange sie in jedem Geschäft brauchen wird. Aber zumindest ihr Weg soll so kurz wie möglich sein.

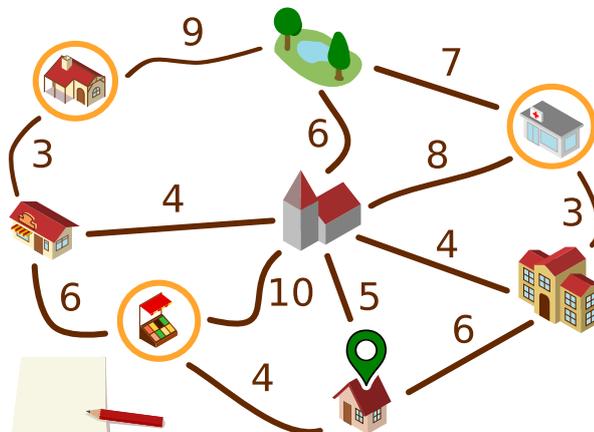
Auf einem Plan hat Emma eingetragen, wie viele Minuten sie für die Strecken zwischen einzelnen Orten ihrer Stadt benötigt. Ausserdem hat sie im Plan markiert, welche Strecke sie auf ihrem Weg geht.

Für diesen Weg benötigt Emma insgesamt  $6 + 3 + 7 + 9 + 3 + 6 + 4 = 38$  Minuten.



Emma überlegt, ob es noch schneller geht. Vielleicht hilft es, manche Strecken hin und zurück zu gehen?

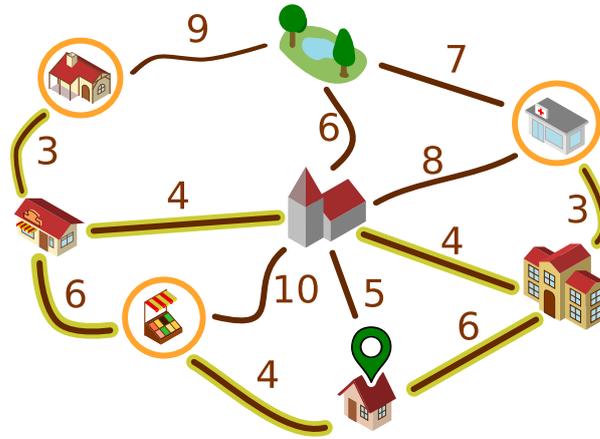
Bestimme den kürzesten Weg, den Emma gehen kann, um ihre drei Aufgaben zu erledigen.





## Lösung

So ist es richtig:



Emma kann so entlang der ausgewählten Strecken gehen (oder in die Gegenrichtung):



Für diesen Weg braucht sie  $6 + 3 + 3 + 4 + 4 + 3 + 3 + 6 + 4 = 36$  Minuten.



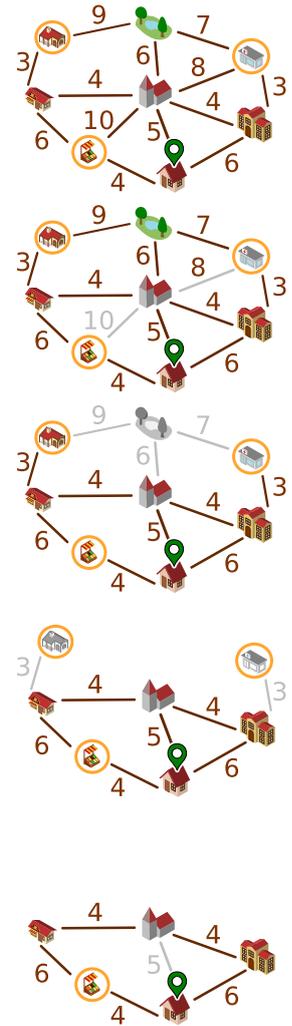
Nun wollen wir begründen, warum es keinen noch kürzeren Weg geben kann. Dazu benutzen wir eine vereinfachte Darstellung des Plans.

Die grau gezeichneten Strecken können wir ignorieren. Es gibt kürzere Wege zwischen den durch die Strecken verbundenen Orte, nämlich über andere Orte.

Auch den Park können wir ignorieren. Emma muss nicht zum Park. Zudem gibt es für jeden Weg, der über den Park geht, eine kürzere Alternative.

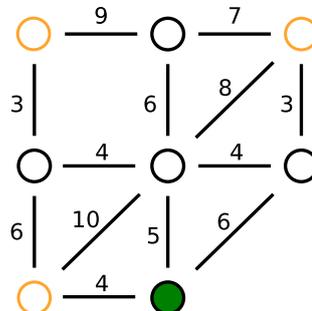
Emma muss zur Apotheke  und zum Kiosk  gehen. Dorthin kommt sie jeweils nur von der Bäckerei  bzw. der Schule . Sie muss jeweils die Strecke zwischen diesen Orten hin- und her gehen. Das dauert jeweils  $3 + 3 = 6$ , insgesamt also 12 Minuten. Das merken wir uns und fassen nun die beiden Orte oben mit denen darunter zu einem zusammen.

Nun bleibt nur noch der Plan rechts übrig. Start und Ende des Weges ist hier . Diese drei Orte (, , ) müssen besucht werden. Der kürzeste Weg, der das erfüllt, geht über alle fünf Orte und entlang aller Strecken ausser der grauen und dauert  $4 + 6 + 4 + 4 + 6 = 24$  Minuten. Mit den 12 Minuten von oben macht das 36 Minuten. Die vorherigen Überlegungen zeigen, dass es keinen kürzeren Weg geben kann.



## Dies ist Informatik!

Für die Begründung der richtigen Antwort wurde eine vereinfachte Darstellung des Plans benutzt. Es wäre möglich gewesen, den Plan noch deutlicher abstrakter darzustellen:



Diese Darstellung enthält alle für Emmas Weg wichtigen Informationen, nämlich

- Objekte: die Orte, wobei die für den Weg wichtigen Orte markiert sind;



- und Beziehungen zwischen den Objekten: die Strecken zwischen den Orten, für die jeweils eine Länge angegeben ist.

Ein wichtiges Werkzeug zur Modellierung von Beziehungen zwischen Objekten sind *Graphen*. Graphen bestehen aus Knoten (für die Objekte) und Kanten (Paare von Objekten, für die Beziehungen). Emmas Plan lässt sich als *gewichteter Graph* modellieren, bei denen die einzelnen Beziehungen mit Zahlenwerten (den *Gewichten*) versehen werden.

Die Informatik interessiert sich für Fragen, die in Bezug auf Graphen gestellt werden können, und für Algorithmen, mit denen man die Fragen beantworten kann. Eine für gewichtete Graphen bedeutsame Frage lautet: Was ist der kürzeste (oder schnellste) Weg zwischen zwei Knoten? Die «Graphen-Frage» in dieser Biberaufgabe ist ähnlich: Was ist der kürzeste Rundweg von einem Knoten aus, bei dem eine Menge anderer Knoten besucht werden? Die Informatik kennt viele Algorithmen, die kürzeste Wege in Graphen effizient bestimmen können. Solche Algorithmen werden zum Beispiel in Software zur Routenplanung implementiert.

## Stichwörter und Webseiten

- Weg (in Graphen): [https://de.wikipedia.org/wiki/Weg\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Weg_(Graphentheorie))
- Kürzeste Wege: Abenteuer Informatik, Kapitel 1  
(<http://abenteuer-informatik.de/dasbuch.html>)



## 12. Zerobots Mission

Zerobot hat einen austauschbaren Treibstofftank. Zerobot bewegt sich damit in einem Raster: nach oben, unten, rechts und links. Bei jeder Bewegung von einem Rasterfeld zum nächsten sinkt der Füllstand des Tanks um 1.

Auf einigen Feldern sind Austausch tanks; die Zahl darauf zeigt den Füllstand an. Wenn Zerobot ein solches Feld erreicht, tauscht er seinen Tank, egal wie voll der ist: Er nimmt den Austausch tank auf, setzt seinen bisherigen Tank auf dem Feld ab und fährt weiter.

Zerobots aktuelle Position und der Füllstand seines Tanks werden im Bild so angezeigt: 



Alarm: Die Tanks sind fehlerhaft und könnten explodieren!

Das ist Zerobots Mission: Er soll so zur Basisstation  fahren, dass am Ende alle Tanks leer sind (Füllstand 0).

*Wie muss sich Zerobot bewegen, um seine Mission zu erfüllen?*

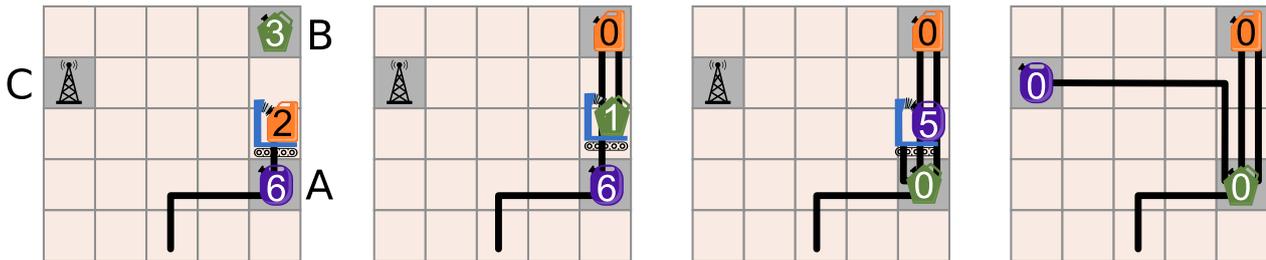


## Lösung

So ist es richtig:



Zerobot kann mit 15 Bewegungen so zur Basisstation fahren, dass am Ende alle Tanks Füllstand 0 haben:



Um die richtige Antwort leichter erklären zu können, bezeichnen wir die Felder mit den Austausch tanks und der Basisstation mit den Buchstaben A, B und C:

Zerobot fährt 3 Felder bis A und tauscht (Füllstand 6) gegen (Füllstand 3) aus. Dann fährt er 3 Felder bis B und tauscht (Füllstand 0) gegen (Füllstand 3) aus. Damit fährt er wieder zu A und tauscht (Füllstand 0) gegen (Füllstand 6) aus. Damit fährt er 6 Felder bis zur Basisstation C. hat dann den Füllstand 0. Mission erfüllt!

Ist dies die einzige richtige Lösung? Zerobot muss exakt 15 Bewegungen machen: 15 Bewegungen sind mindestens nötig, um den gesamten verfügbaren Treibstoff von  $9 + 3 + 3 = 15$  Einheiten zu verbrauchen, und für mehr Bewegungen reicht der Treibstoff nicht. Um alle Tanks zu leeren, muss er beide Felder mit Austausch tanks besuchen, und A sogar zweimal. Wenn der Zerobot zuerst das Feld B besuchen würde, bräuchte er 17 Bewegungen, um die Basisstation zu erreichen, was nicht möglich ist. Somit ist die gezeigte Reihenfolge der Tanks die einzige richtige Antwort.

## Dies ist Informatik!

In dieser Biberaufgabe werden einige grundsätzliche Probleme der autonomen Mobilität angesprochen: Jeder autonome mobile Roboter (wie z.B. ein selbstfahrendes Auto) muss beachten, wie viel Energie in Form von Treibstoff oder Batterieladung zur Verfügung steht, wenn er seine Aktivitäten plant. Auf der einen Seite muss er sicherstellen, dass er rechtzeitig eine Ladestation oder Tankstelle erreicht, bevor sein Energievorrat verbraucht ist. Auf der anderen Seite gibt es Rahmenbedingungen zu beachten. In der Aufgabe ist eine Rahmenbedingung, dass am Ende der Energievorrat komplett verbraucht sein musste. In der Wirklichkeit hat man es vor allem mit anderen Rahmenbedingungen zu tun, wie z.B. die Position und Verfügbarkeit von Ladestationen. Die Software zur Steuerung mobiler Roboter enthält Komponenten, die für die Sicherstellung ausreichender Energie durch Nachladen sorgen (*intelligentes Batterieladungsmanagement*).



Darüber hinaus werden Computerprogramme auch zur Planung und Verwaltung effizienter Netze von Ladestationen verwendet. Informatikerinnen und Informatiker forschen an Lösungen zum charging station placement problem: Ladestationen für mobile Roboter müssen so platziert werden, dass ein Roboter mit einem gewissen Mindestladestand eine der verfügbaren Ladestationen erreichen kann. Für die Kommunikation zwischen Ladestationen und selbstfahrenden Autos wurden Protokolle entwickelt wie z.B. das OCPP (Open Charge Point Protocol).

## Stichwörter und Webseiten

- Intelligentes Batterieladungsmanagement:  
[https://www.researchgate.net/publication/364734487\\_Intelligent\\_Battery\\_Recharge\\_Management\\_for\\_Mobile\\_Robots](https://www.researchgate.net/publication/364734487_Intelligent_Battery_Recharge_Management_for_Mobile_Robots)
- Open Charge Point Protocol: <https://de.wikipedia.org/wiki/OCPP>





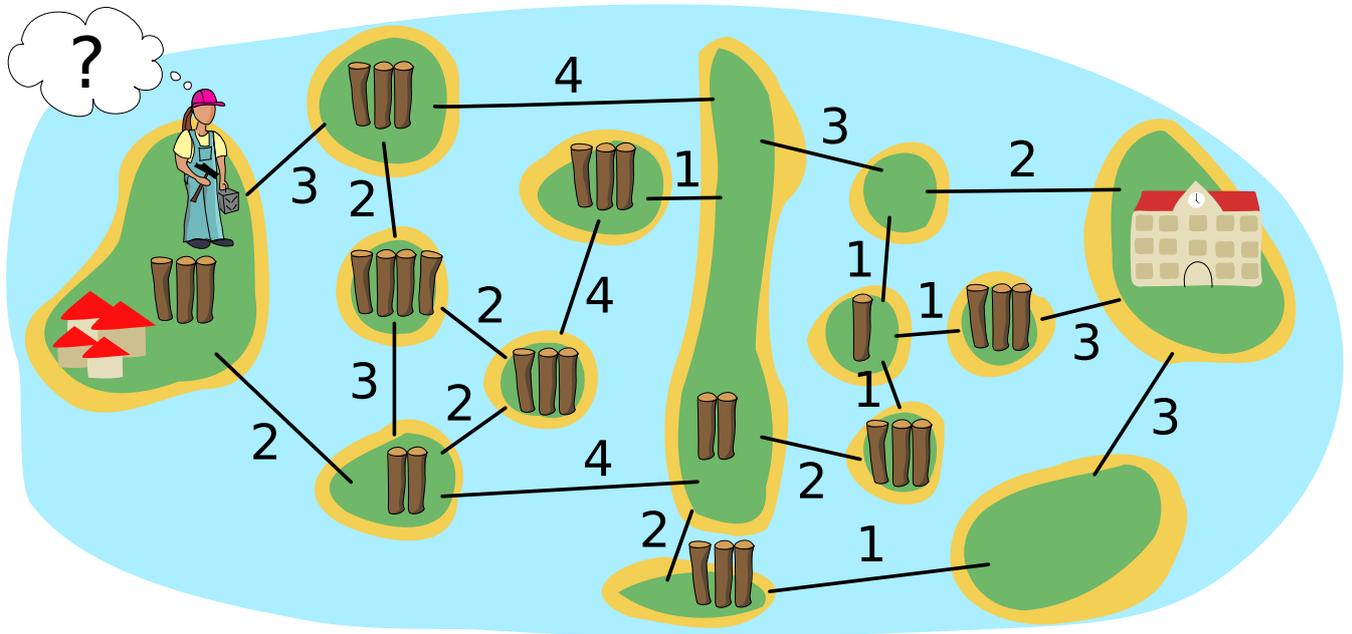
# 13. Brücken bauen!

Auf der Insel ganz links sind Kinder eingezogen. Bianca soll Brücken bauen, über die die Kinder zur Schule auf der Insel ganz rechts gehen können.

Die Insel-Karte zeigt, wie viele Baumstämme es auf jeder Insel gibt. Diese Baumstämme kann Bianca nehmen, um an den Linien Brücken zu bauen. Die Zahl an einer Linie sagt, wie viele Baumstämme dort für eine Brücke benutzt werden. Sobald es zwischen zwei Inseln eine Brücke gibt, kann Bianca darüber gehen und Stämme, die sie noch hat, mitnehmen. Natürlich kann sie jeden Baumstamm nur für eine Brücke benutzen.

Bianca fängt auf er Insel links an. Ihr Ziel ist, möglichst wenige Baumstämme zu benutzen.

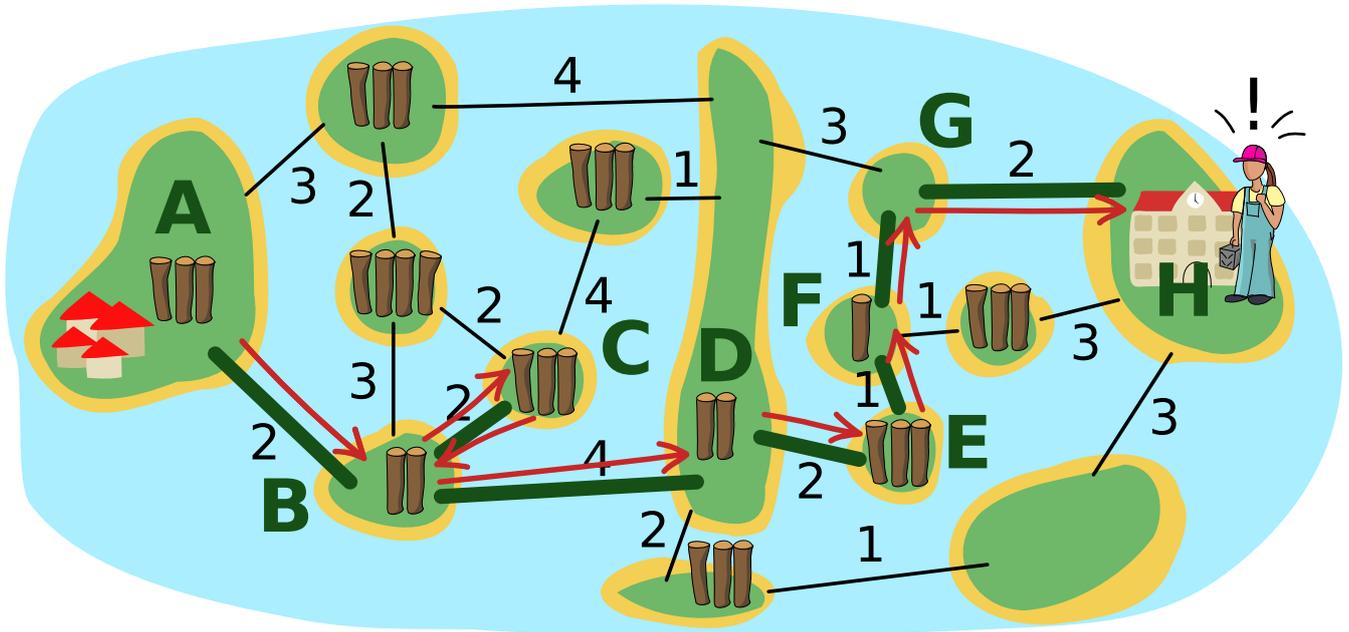
*An welchen Linien soll Bianca Brücken bauen, damit sie ihr Ziel erreicht?*





## Lösung

So ist es richtig:



Die grünen Linien zeigen, wo Bianca Brücken gebaut hat. Die roten Pfeile zeigen, wie Bianca über die Brücken gegangen ist:

- Auf der Insel A nimmt sie die drei Baumstämme und benutzt zwei davon für die erste Brücke. Mit dem verbleibenden Baumstamm geht sie über die Brücke und hat auf der Insel B  $3 - 2 + 2 = 3$  Baumstämme. Das sind nicht genug, um eine Brücke zur Insel D zu bauen.
- Deshalb baut sie mit 2 Stämmen eine Brücke zur Insel C. Sie geht über die Brücke, nimmt die 3 Stämme von der Insel C und geht zurück. Nun hat sie  $3 - 2 + 3 = 4$  Stämme.
- Damit baut sie eine Brücke zur Insel D, geht über die Brücke und hat dann die 2 Stämme von Insel D.
- Damit baut sie eine Brücke zur Insel E und kann dort 3 Stämme nehmen. Sie baut weitere Brücken zu den Inseln F und G. Auf der Insel E hat sie also 3 Stämme, auf der Insel F  $3 - 1 + 1 = 3$  Stämme und auf der Insel G noch 2 Stämme.
- Die reichen genau, um eine Brücke zur Insel H mit der Schule zu bauen.

Insgesamt konnte Bianca also Brücken für einen Weg von Insel A zu Insel H bauen und hat dafür 14 Baumstämme benutzt. Aber geht es auch mit weniger Stämmen? Dazu müssen alle möglichen Wege untersucht werden. Weil die alle über die lange Insel D führen, lässt sich das Problem in zwei Teile zerlegen: Von Insel A zu Insel D, und von Insel D zu Insel H:

- Für die Brücken von Insel A bis Insel D hat Bianca 8 Stämme benutzt und kam ohne Stamm auf Insel D an. Wir notieren ihren Weg so:  $2 - [2, 2] - 4$  (von der Insel A über die Linie mit der 2 zur Insel B, dann zwischen B und C hin und zurück über die 2, dann über die 4 zu Insel D). Ein Weg mit weniger Stämmen wäre  $3 - 4$ , kann aber nur mit Umweg gebaut werden



$(3 - [2, 2] - 4)$ , verbraucht also 9 Stämme, wobei Bianca auf Insel D mit einem Stamm im Vorrat ankommt. Alle anderen Wege von Insel A bis D verbrauchen 9 Stämme oder mehr.

- Für die Brücken von Insel D bis H hat Bianca 6 Stämme benutzt. Den direkten Weg  $3 - 2$  kann sie nicht bauen, auch nicht mit einem Stamm im Vorrat. Alle anderen Wege von Insel D zu Insel H verbrauchen 6 Stämme oder mehr.

Es ist also nicht möglich, mit weniger als 14 Stämmen Brücken zu bauen, über die die Kinder von der Dorf-Insel A zur Schul-Insel H gehen können. Mit den von ihr gebauten Brücken hat Bianca also ihr Ziel erreicht.

## Dies ist Informatik!

Die Insel-Karte mit den durch Linien angezeigten «Brücken-Bauplätzen» kann als *Graph* modelliert werden: Das ist eine mathematische Struktur, die Objekte (auch Knoten genannt) paarweise miteinander in Relation setzt (die Paare nennt man auch Kanten). In einem Graphen man die Inseln als Knoten und die Linien als Kanten modellieren. Dabei haben die Kanten *Gewichte*, nämlich die Anzahl der für den Brückenbau entlang einer Linie benutzten Baumstämme, aber auch die Knoten (die Anzahl der Stämme auf einer Insel) – das ist eher ungewöhnlich. Für Graphen, bei denen nur die Kanten gewichtet sind, kennt die Informatik mehrere effiziente Algorithmen, die einen kürzesten Weg (über Kanten mit minimaler Summe der Gewichte) zwischen zwei Knoten berechnen können.

Das Problem, das Bianca in dieser Biberaufgabe optimal lösen möchte, ist komplizierter: Sie möchte zwar auch einen kürzesten Weg gehen, hat aber eine Randbedingung: Die Summe der Knotengewichte auf ihrem bisherigen Weg (die Stämme, die sie nehmen konnte) abzüglich der Summe der Kantengewichte auf ihrem Weg (die Stämme, die sie für den Brückenbau benutzt hat) muss grösser sein als das Gewicht der Kante, die sie als nächste gehen bzw. wo sie eine Brücke bauen möchte. Um den optimalen Weg zu finden, müssen hier eventuell alle Möglichkeiten ausprobiert werden. Die Zerlegung des Problems in zwei Teile hilft, die Anzahl der Möglichkeiten zu reduzieren. Und wegen der Randbedingung kann man viele Möglichkeiten ausschliessen, bevor man sie komplett probiert hat. In der Informatik ist ein solches Vorgehen (Probieren und Ausschliessen) als *Backtracking* bekannt (siehe auch die Biberaufgabe «Gemüsebeet»).

## Stichwörter und Webseiten

- Graph: [https://de.wikipedia.org/wiki/Graph\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Graph_(Graphentheorie))
- gewichteter Graph: [https://de.wikipedia.org/wiki/Kantengewichteter\\_Graph](https://de.wikipedia.org/wiki/Kantengewichteter_Graph)





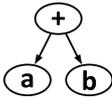
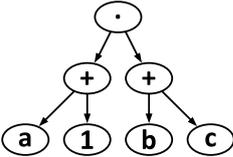
# 14. Postfix-Notation

Ein mathematischer Ausdruck besteht aus ...

- ... einem *Operator*: +, -, · oder :
- ... und den *Operanden*: Zahlen wie 1, 2, ..., Buchstaben wie a, b, ... oder wieder Ausdrücke wie (1 + 2).

Die Struktur eines mathematischen Ausdrucks kann man als *Strukturbaum* darstellen. Dieses Diagramm aus Operatoren und Operanden wird so gezeichnet: Ein Kringel mit dem Operator wird durch Pfeile mit den Strukturbäumen der Operanden verbunden. Das sind im einfachsten Fall Kringel mit einer Zahl oder einem Buchstaben.

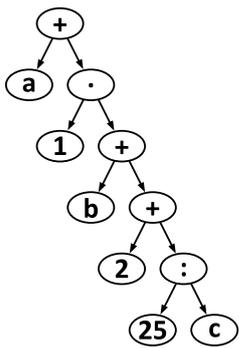
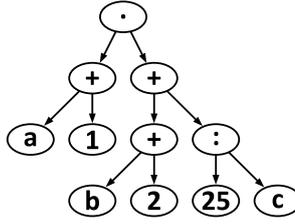
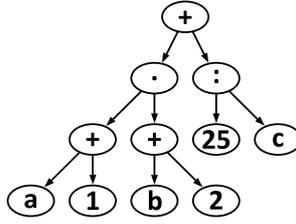
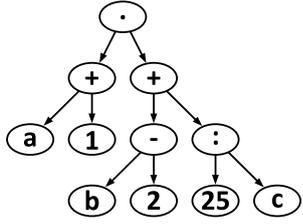
Aus einem Strukturbaum wiederum kann man die *Postfix-Notation* eines mathematischen Ausdrucks ablesen. In dieser Notation werden für jeden Ausdruck zunächst die Operanden und dahinter der Operator geschrieben.

<b>Mathematischer Ausdruck:</b>	$a + b$	$(a + 1) \cdot (b + c)$
<b>Strukturbaum:</b>		
<b>Postfix-Notation:</b>	$a b +$	$a 1 + b c + \cdot$

Hier ist die Postfix-Notation eines anderen Ausdrucks:

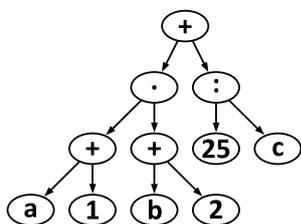
$$a 1 + b 2 + \cdot 25 c : +$$

Welchen Strukturbaum hat dieser Ausdruck?

			
A)	B)	C)	D)



## Lösung



Antwort C ist richtig:

Wie in der Aufgabenstellung beschrieben ist, findet sich der zentrale Operator eines mathematischen Ausdrucks im Strukturbaum ganz oben (er ist dessen *Wurzel*) und in der Postfix-Notation ganz hinten. Möchte man den Strukturbaum eines Ausdrucks in Postfix-Notation finden oder erstellen, muss man das in der Postfix-Notation letzte Zeichen oben im Baum suchen, in diesem Fall das +. Nur bei den Bäumen der Antworten A und C findet sich ein + in der Wurzel.

Der Operator + hat zwei Operanden, einer links und einer rechts. In der Postfix-Notation sieht man direkt (am vorletzten Zeichen), dass der rechte Operand des Ausdrucks wiederum ein Ausdruck ist, der den Operator : hat. Im Strukturbaum muss also rechts unter der Wurzel ein : zu sehen sein. Das ist nur im Baum aus Antwort C der Fall. Also muss das die richtige Antwort sein.

Das kann man auch zeigen, indem man den Strukturbaum aus Antwort C vollständig in Postfix-Notation umwandelt:

- Die untersten drei «kleinsten» Teilbäume, die jeweils aus 3 Elementen bestehen, werden zu  $a\ 1\ +$ ,  $b\ 2\ +$  und  $25\ c\ :$ .
- Die beiden linken dieser drei «kleinsten» Teilbäume werden zum linken Operanden des oberen +, so dass die Umwandlung des linken Teilbaums nun  $a\ 1\ +\ b\ 2\ +\ \cdot$  lautet. Der dritte der «kleinsten» Teilbäume ist bereits der rechte Operand.
- Damit lautet die Postfix-Notation des Strukturbaums aus Antwort C insgesamt so:  $a\ 1\ +\ b\ 2\ +\ \cdot\ 25\ c\ : +$ . Das ist genau der in der Aufgabenstellung vorgegebene Ausdruck.

## Dies ist Informatik!

Die *Postfix-Notation*, auch *umgekehrte polnische Notation* (engl. *reverse Polish notation RPN*) genannt, wird oftmals verwendet, um mathematische oder andere Ausdrücke (z. B. in Programmiersprachen) missverständnisfrei und kompakt zu formulieren. Würde man den durch den Strukturbaum aus Antwort C gegebenen Ausdruck in normaler Notation schreiben (also mit Operatoren zwischen den Operanden, deshalb auch *Infix-Notation* genannt), müsste man Klammern setzen  $(a + 1) \cdot (b + 2) + 25 : c$ , die man für die Postfix-Notation nicht braucht. Die Postfix-Notation wurde von Jan Łukasiewicz (1878–1956) zunächst als Präfix-Notation eingeführt, mit dem Operator vor den Operanden. So schreibt man u.a. die Anwendung von Funktionen auf: in der Mathematik  $f(x, y)$ ,



beim Programmieren `funktionsname(argument1, argument2, argument3)`. Im Computer wird sie unter anderem beim *Parsen* von Ausdrücken einer Programmiersprache verwendet.

In der jüngeren Vergangenheit haben viele Menschen die Postfix-Notation vor allem bei der Nutzung der ersten wissenschaftlichen Taschenrechner kennengelernt: Dort konnte man damit schnell und zuverlässig und vor allem ohne Klammern komplexe mathematische Ausdrücke eingeben und berechnen lassen. Noch heute gibt es eine eingeschworene Gemeinschaft, die programmierbare Taschenrechner (wie z. B. den HP 35s) mit Postfix-Notation nutzen.

## Stichwörter und Webseiten

- Umgekehrte polnische Notation UPN:  
[https://de.wikipedia.org/wiki/Umgekehrte\\_polnische\\_Notation](https://de.wikipedia.org/wiki/Umgekehrte_polnische_Notation)
- Syntaxbaum: <https://de.wikipedia.org/wiki/Syntaxbaum>
- Jan Łukasiewicz: [https://de.wikipedia.org/wiki/Jan\\_Łukasiewicz](https://de.wikipedia.org/wiki/Jan_Łukasiewicz)
- HP 35s: [https://en.wikipedia.org/wiki/HP\\_35s](https://en.wikipedia.org/wiki/HP_35s)





## 15. Ziffernschloss

Bob hat ein Ziffernschloss an seiner Haustür. Um es zu öffnen, muss man einen Zifferncode eingeben. Alle Ziffern im Code müssen verschieden sein. Aktuell hat der Code fünf Stellen und lautet so:

Bob hat sich den Code notiert, aber ein wenig verschleiert:  $n \gg c$  bedeutet, dass links von Ziffer  $c$  im Code genau  $n$  Ziffern stehen, die grösser sind als  $c$ . Zum Beispiel notiert Bob mit

$1 \gg 3$

dass links von Ziffer 3 genau eine Ziffer steht (nämlich die 4), die grösser ist als 3. Den aktuellen Zifferncode hat er sich insgesamt so notiert:

$0 \gg 0 ; 3 \gg 1 ; 0 \gg 2 ; 1 \gg 3 ; 0 \gg 4$

Ein Code aus nur fünf Ziffern ist Bob zu unsicher. Deshalb überlegt er sich einen neuen Code, aus den Ziffern 0 bis 7. Den neuen Code notiert er sich so:

$3 \gg 0 ; 2 \gg 1 ; 4 \gg 2 ; 4 \gg 3 ; 1 \gg 4 ; 1 \gg 5 ; 1 \gg 6 ; 0 \gg 7$

Wie lautet der neue Code?



## Lösung

So ist es richtig:

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

Um den Code zu bestimmen, schauen wir uns Bobs Notation genauer an, nach und nach für die Ziffern 0 bis 7.

- $3 \gg 0$ : Links von 0 stehen genau 3 Ziffern, die grösser sind als 0. Die Ziffer 0 muss also an der vierten Stelle des Codes stehen.
- $2 \gg 1$ : Links von 1 stehen genau 2 Ziffern, die grösser sind als 1. Die Ziffer 1 muss also an der dritten Stelle des Codes stehen.
- $4 \gg 2$ : Links von 2 stehen genau 4 Ziffern, die grösser sind als 2. Da die kleineren Ziffern 1 und 0 bereits an dritter und vierter Stelle stehen, müssen die 4 grösseren Ziffern an erster, zweiter, fünfter und sechster Stelle stehen. Die Ziffer 2 muss also an der siebten Stelle des Codes stehen.
- $4 \gg 3$ : Links von 3 stehen genau 4 Ziffern, die grösser sind als 3. Die Ziffer 3 muss also an der achten und letzten Stelle des Codes stehen.
- $1 \gg 4$ : Links von 4 steht genau 1 Ziffer, die grösser ist als 4. Die Ziffer 4 muss also an der zweiten der verbleibenden Stellen stehen; das ist die zweite Stelle des Codes.
- $1 \gg 5$ : Links von 5 steht genau 1 Ziffer, die grösser ist als 5. Die Ziffer 5 muss also an der zweiten der nun noch verbleibenden Stellen stehen; das ist die fünfte Stelle des Codes.
- $1 \gg 6$ : Links von 6 steht genau 1 Ziffer, die grösser ist als 6. Die Ziffer 6 muss also an der zweiten der nun noch verbleibenden Stellen stehen; das ist die sechste Stelle des Codes.
- $0 \gg 7$ : Es gibt keine Ziffer, die grösser ist als 7. Die Ziffer 7 muss an der letzten freien Stelle, also an der ersten Stelle des Codes stehen.

## Dies ist Informatik!

Bob beschreibt in seiner Notation, wie sich der Code zu einer sortierten Folge der verwendeten Ziffern bzw. Zahlen verhält.

Schauen wir uns den fünfstelligen Code noch einmal an: 0 2 4 3 1. Er entsteht, indem man die sortierten Zahlen 0 1 2 3 4 nimmt und deren Positionen verändert. Das Ergebnis nennt man auch *Permutation* (der Zahlen 0 bis 4). In einer Permutation können die Zahlen bzgl. ihrer Sortierung «verdreht» sein. Zum Beispiel steht im Code die 4 vor der 3, während in der sortierten Folge die 3 vor der 4 steht (denn  $3 < 4$ ). Also steht die 3 «falsch» bzgl. der Sortierung. Das bezeichnet man in der Kombinatorik, einem Teilgebiet der Mathematik, als *Inversion* oder auch *Fehlstand*.

Bobs Code ist also eine Permutation, und seine Notation gibt für jede Zahl an, wie oft sie darin «invertiert» ist: Die 0 steht richtig, die 1 ist Teil von 3 Inversionen ( $3 \gg 1$ : drei grössere Zahlen stehen vor der 1), die 2 steht richtig, die 3 ist einmal invertiert, die 4 steht richtig.

Die Folge dieser Inversionszahlen heisst *Inversionssequenz*. Die 0 steht richtig, die 1 ist Teil von



3 Inversionen (3 >> 1: drei größere Zahlen stehen vor der 1), die 2 steht richtig, die 3 ist einmal invertiert, die 4 steht richtig. Die Folge dieser Inversionszahlen heißt *Inversionssequenz*. (Die Summe der Inversionszahlen beschreibt übrigens den Grad der Unsortiertheit einer Permutation; vergleiche dazu auch die Biberaufgabe «Zug entladen».)

Wir haben nun drei Folgen – den Code (bzw. die Permutation), die sortierte Folge und die Inversionssequenz – und fassen sie in dieser Tabelle zusammen:

<b>Code / Permutation</b>	0	2	4	3	1
<b>Sortierte Folge</b>	0	1	2	3	4
<b>Inversionssequenz</b>	0	3	0	1	0

Die Beschreibung der Lösung hat gezeigt, dass es einen effizienten Algorithmus gibt, der aus der Inversionssequenz die zugehörige Permutation berechnet. Es genügt, die Inversionssequenz einmal durchzugehen. Die Informatik beschäftigt sich häufig mit kombinatorischen Problemen und kennt viele Algorithmen zur Lösung solcher Probleme. Sie können bei der automatischen Lösung von Rätseln verwendet werden (wie etwa Sudokus), aber auch bei vielen «ernsthaften» Problemen. Meist sind sie deutlich komplizierter als der Algorithmus zur Lösung dieser Biberaufgabe.

## Stichwörter und Webseiten

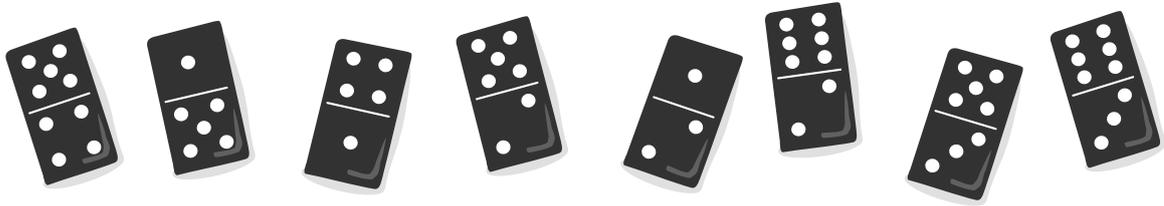
- Permutation: <https://de.wikipedia.org/wiki/Permutation>
- Inversion bzw. Fehlstand: <https://de.wikipedia.org/wiki/Fehlstand>





## 16. Domino

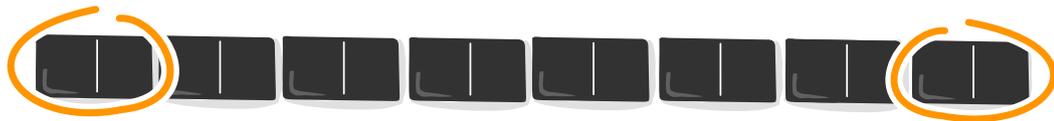
Jeder Dominostein hat zwei Felder. Auf jedem Feld sind 1 bis 6 Punkte. Du hast diese acht Steine:



Alle acht Steine sollst du so in eine Reihe legen, dass auf den angrenzenden Feldern zweier benachbarter Steine immer gleich viele Punkte sind.



Du kannst mehrere solcher Reihen legen. Es gibt aber Steine, die du auf keinen Fall an den Anfang oder das Ende deiner Reihe legen kannst.



Welche Steine sind das?



## Lösung

Drei der acht Steine können nicht an den Anfang oder das Ende der Reihe gelegt werden:



Um die Aufgabe zu lösen, untersuchen wir die Augenzahlen (die Punkte auf Dominosteinen nennt man auch *Augen*, wie bei einem Würfel) der 16 Felder der Dominosteine. Wir halten fest, wie häufig die einzelnen Augenzahlen vorkommen, und ob die Häufigkeit eine gerade oder eine ungerade Zahl ist:

Augenzahl	Häufigkeit	Gerade/Ungerade
	3	ungerade
	3	ungerade
	2	gerade
	2	gerade
	4	gerade
	2	gerade

Felder mit Augenzahlen, die mit gerader Häufigkeit vorkommen, müssen paarweise mitten in der Reihe liegen oder gleichzeitig an Anfang und Ende. Felder mit Augenzahlen, die mit ungerader Häufigkeit vorkommen, können aber nicht alle mitten in der Reihe liegen: Es kann nämlich nicht für jedes Feld mit dieser Augenzahl ein passendes angrenzendes Feld gefunden werden; das geht nur bei gerader Häufigkeit. Hier siehst du eine Reihe, in die ein Stein mit der dreimal vorkommenden Augenzahl 1 nicht mehr mitten hinein passt.



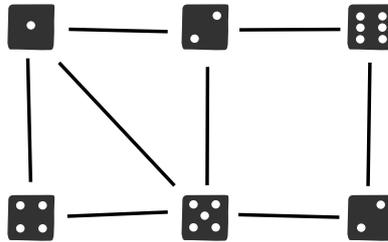
Da es auf den acht Steinen dieser Biberaufgabe Felder mit ungerader Häufigkeit gibt, müssen Steine mit solchen Feldern außen liegen. Steine, die zwei Felder mit gerader Häufigkeit haben, können also nicht an den Anfang oder das Ende der Reihe gelegt werden. Das sind folgende Steine:



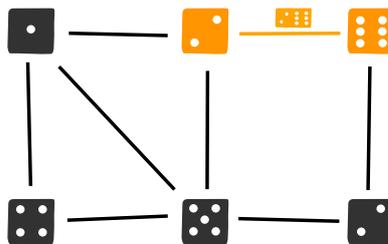


## Dies ist Informatik!

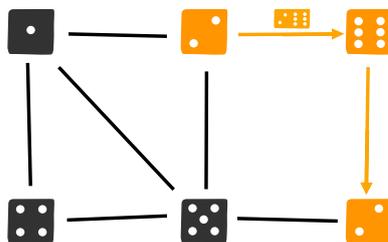
Es gibt mehrere Möglichkeiten, die acht Dominosteine dieser Biberaufgabe in eine korrekte Reihe zu legen. Um sich hier eine bessere Übersicht zu verschaffen, verwenden Informatiker sogenannte *Graphen*:



Im Graphen oben sind Quadrate (sogenannte *Knoten*) ersichtlich, die die sechs Augenzahlen der Dominosteine aufzeigen. Die acht Linien (sogenannte *Kanten*), stellen die acht Dominosteine dar; jede Linie verbindet zwei Felder. Der Dominostein 2–6 wird beispielsweise durch die folgende Kante dargestellt:



Um die Aufgabe zu lösen, müssen alle acht Dominosteine passend aneinander gereiht werden. Dabei ist nach dem Legen des ersten Dominosteins bereits klar, mit welcher Augenzahl der zweite Stein beginnen muss, denn angrenzende Felder zweier Steine sollen ja immer die gleiche Augenzahl haben. Im Graph erkennt man dies dadurch, dass Dominosteine genau dann nebeneinander gelegt werden dürfen, wenn deren Kanten sich bei demselben Knoten treffen. Die Steine 2–6 und 6–3 können beispielsweise aneinander gelegt werden, da sie beide die Augenzahl 6 enthalten:



Das Aneinanderreihen der Dominosteine lässt sich als *Weg* (eine Abfolge von Kanten) durch den Graphen verstehen. Dieser Weg soll sämtliche Kanten *genau einmal* besuchen, um sicherzustellen, dass die acht Dominosteine einerseits alle verwendet werden, andererseits aber auch nicht mehrmals zum Einsatz kommen. Ein Weg, der *jede Kante genau einmal* besucht, wird *Eulerweg* genannt. Der Name geht auf Leonhard Euler, einen Schweizer Mathematiker und den Erfinder der Graphentheorie, zurück. Euler konnte zeigen, dass in einem zusammenhängenden Graphen ein Eulerweg genau dann



existiert, wenn maximal zwei Knoten eine ungerade Anzahl von diesem Knoten ausgehender Kanten haben.

## Stichwörter und Webseiten

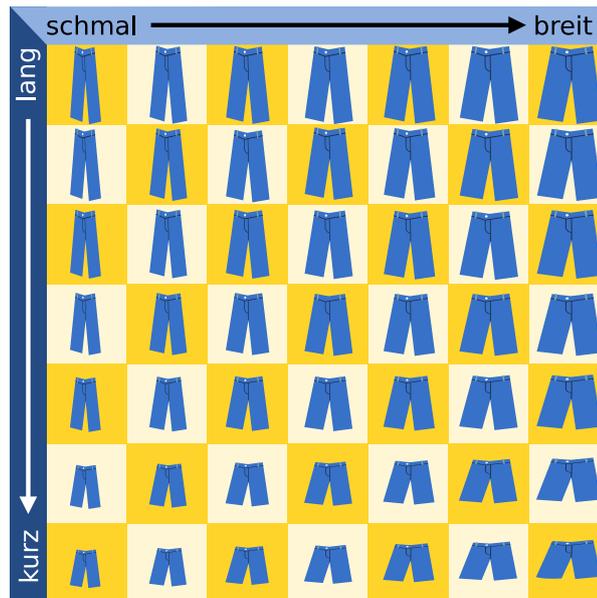
- Graph, Knoten, Kanten: [https://de.wikipedia.org/wiki/Graph\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Graph_(Graphentheorie))
- Eulerweg: <https://de.wikipedia.org/wiki/Eulerkreisproblem>



## 17. Anprobieren

Christian braucht neue Hosen. Im Geschäft gibt es seine Lieblings-Hose in sieben Längen und sieben Breiten. Hosen in allen 49 Grössen sind im Regal, nach Länge und Breite sortiert.

Weil Christian seine richtige Grösse nicht weiss, muss er sie durch Anprobieren herausfinden. Bei jeder Anprobe merkt Christian, ob die Hose passt oder ob er eine kürzere, längere, schmalere oder breitere Hose braucht. Damit eine Hose passt, müssen Länge und Breite stimmen.



Der Verkäufer stöhnt: Bei 49 Grössen die richtige zu finden – das kann dauern.

Doch Christian ist eine Methode eingefallen, die richtige Grösse in jedem Fall nach möglichst wenigen Anproben zu wissen.

*Wie viele Anproben braucht er mit dieser Methode höchstens, bis er die richtige Grösse weiss?*



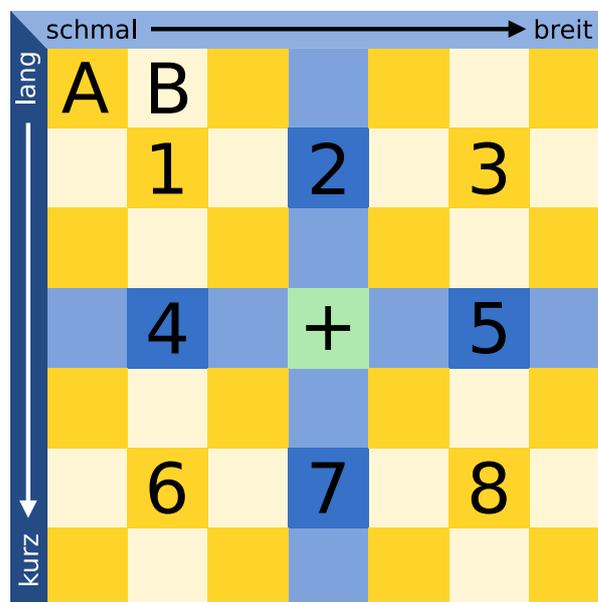
## Lösung

2 ist die richtige Antwort.

Natürlich kann Christian Glück haben und direkt bei der ersten Anprobe die Hose in der richtigen Grösse erwischen. Aber auf Glück kann er sich nicht verlassen und geht nach dieser Methode vor:

Zuerst probiert er die Hose in der Mitte an (Position + im Bild). Bei der Anprobe prüft er Länge und Breite der Hose.

- Wenn Länge und Breite stimmen, hat der die Hose mit der richtigen Grösse gefunden.
- Wenn die Hose zu kurz und zu breit ist, ist die passende Hose in Bereich 1.
- Wenn die Hose zu kurz ist aber die richtige Breite hat, ist die passende Hose in Bereich 2.
- Wenn die Hose zu kurz und zu schmal ist, ist die passende Hose in Bereich 3.
- Dies kann man für die Bereiche 4 bis 8 fortführen.



Nehmen wir an, die Hose mit der richtigen Grösse ist in Bereich 1. Christian wählt für die zweite Anprobe die Hose in der Mitte von Bereich 1.

Nun gibt es wieder mehrere Möglichkeiten:

- Wenn die Hose passt, hat er die richtige Grösse gefunden.
- Wenn die Hose immer noch zu kurz und zu breit ist, weiss Christian, dass die Hose an Position A die richtige Grösse hat.
- Wenn die Hose zu kurz ist, aber die passende Breite hat, weiss Christian, dass die Hose an Position B die richtige Grösse hat.
- Dies kann man für die anderen Positionen rund um die Mitte von Bereich 1 fortführen.



Weil in jedem nummerierten Bereich das mittlere Regalfach in jeder Richtung nur ein Nachbarfach hat, sind keine weiteren Anproben notwendig. Christian braucht also in jedem Fall höchstens zwei Anproben, um die richtige Grösse zu wissen.

## Dies ist Informatik!

Die Methode, die Christian bei der Anprobe anwendet, heisst in der Informatik binäre Suche. Der Begriff *binär* kommt vom lateinischen Wort *bis* (zweimal). Bei der binären Suche nach einem Objekt in einer Folge sortierter Objekte wird deren mittleres Objekt mit dem gesuchten verglichen. Wenn das mittlere Objekt nicht schon das gesuchte ist, weiss man immerhin, in welcher Hälfte der Folge sich das gesuchte Objekt befindet und durchsucht diese Hälfte wieder binär. In jedem Schritt wird die Folge also in zwei Teile geteilt – deshalb «binär». Auf diese Weise kommt man sehr schnell beim gesuchten Objekt an. Bei 1.000 Objekten werden etwa 10 Suchschritte benötigt, bei 1.000.000 Objekten etwa 20. Allgemein kann man sagen: Bei  $n$  Objekten werden etwa  $\log(n)$  Schritte benötigt; die Funktion  $\log$  ist der «Zweier-Logarithmus» oder der Logarithmus zur Basis 2. Weil die binäre Suche so schnell ist, wird sie in Computerprogrammen häufig für die Suche in sortierten Daten verwendet.

In dieser Biberaufgabe ist der Suchraum, nämlich die Hosen im Regal, in zwei Dimensionen (Länge und Breite) sortiert. Deshalb kann Christian die binäre Suche gleich auf beide Dimensionen anwenden. Dann teilt sich die Suchmenge in einem Schritt nicht in 2, sondern gleich in 8 Teile auf - falls Christian nicht direkt die richtige Grösse erwischt hat.

## Stichwörter und Webseiten

- Binäre Suche: [https://de.wikipedia.org/wiki/Binäre\\_Suche](https://de.wikipedia.org/wiki/Binäre_Suche)
- Suchverfahren: <https://de.wikipedia.org/wiki/Suchverfahren>





# 18. Konflikt-Detektor

Anna und Ben wollen einen «Konflikt-Detektor» bauen, der anzeigt, ob sie eine unterschiedliche Meinung haben.

Sie verwenden Einheiten, die in zwei Zuständen sein können: Ja und Nein. Zwei Einheiten können mit einem Kabel verbunden werden.

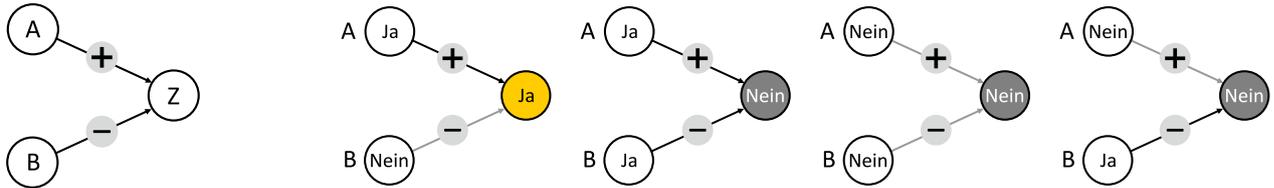
Wenn eine Einheit im

- Zustand Ja ist, sendet sie über alle ausgehenden Kabel ein Signal.
- Zustand Nein ist, sendet sie kein Signal.

Die Kabel werden so eingestellt, dass sie ein Signal als positives (+) oder negatives (-) Signal an die rechts angeschlossene Einheit übermitteln.

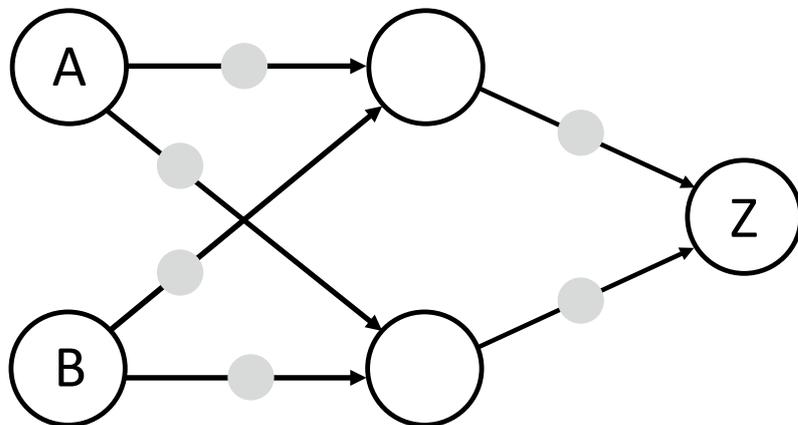
Eine angeschlossene Einheit geht in den Zustand Ja, wenn sie mehr positive als negative Signale empfängt, und sonst in den Zustand Nein. Als Eingabe setzt Anna den Zustand der Einheit A und Ben den Zustand der Einheit B.

Zuerst bauen Anna und Ben diese Maschine: Sie bemerken, dass die Einheit Z nur dann Ja ist, wenn A Ja und B Nein ist. Das ist nicht das, was sie wollen.



Dann bauen Anna und Ben eine grössere Maschine (unten im Bild) und sind sicher, dass sie der Konflikt-Detektor sein kann: Z soll nur dann Ja sein, wenn A und B in unterschiedlichen Zuständen sind (Ja und Nein bzw. Nein und Ja). Ansonsten soll Z im Zustand Nein sein. Jetzt müssen nur noch die Kabel richtig eingestellt werden.

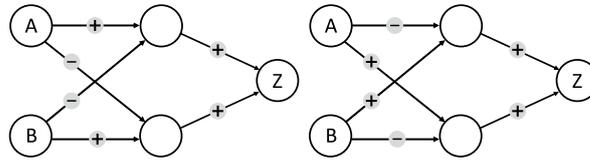
Stelle für jedes Kabel ein, ob es ein Signal positiv (+) oder negativ (-) übermittelt, damit der Konflikt-Detektor korrekt arbeitet.



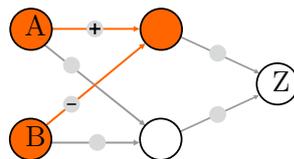


### Lösung

Diese beiden Antworten sind richtig:

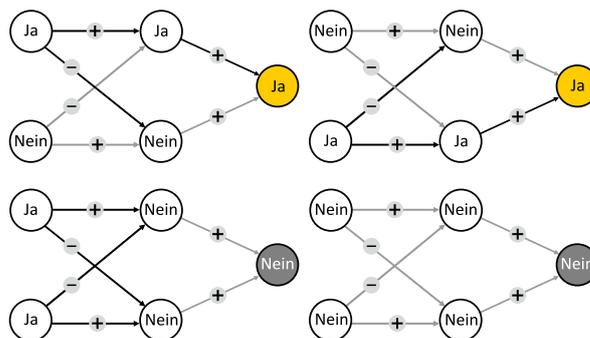


Im Konflikt-Detektor muss die Ausgabe-Einheit genau bei zwei unterschiedlichen Eingaben (A=Ja und B=Nein sowie A=Nein und B=Ja) auf Ja sein. Z kann nur Ja sein, wenn über die zwei eingehenden Kabel mehr positive als negative Signale ankommen. Mindestens eines der Kabel muss also ein positives Signal (+) übermitteln. Nehmen wir einmal an, nur das obere Kabel, das zu Z führt, wird auf + gestellt. Dann muss die Einheit oben Mitte beide gewünschten Eingabekombinationen erkennen können, also in beiden Fällen Ja sein. Zusammen mit den Eingabeeinheiten A und B bildet diese Einheit aber genau so eine Maschine, wie Anna und Ben sie zu Beginn gebaut haben. Sie kann nur in genau einem der gewünschten Fälle Ja sein, und zwar, wenn eines ihrer Kabel auf + und das andere auf - gestellt wird:



Es wird also für jeden der gewünschten Eingabefälle eine eigene Einheit in der Mitte benötigt, eine für A=Ja und B=Nein, die andere für A=Nein und B=Ja. Die Kabel zur ersten Einheit müssen auf + (Kabel von A) und - (B) gestellt werden, die Kabel zur anderen Einheit auf - (A) und + (B). Welche Einheit in der Mitte welchen Fall übernimmt, ist egal; deshalb gibt es bei den Kabeln von A und B zur Mitte zwei Möglichkeiten. Wenn nun jede Einheit in der Mitte in genau einem gewünschten Fall Ja ist, müssen beide Kabel von der Mitte zu Z auf + gestellt werden; nur dann ist Z=Ja in genau beiden gewünschten Fällen.

Für die erste richtige Antwort zeigt das Bild unten die Funktion des Konflikt-Detektors. Man sieht: Die obere Einheit in der Mitte erkennt den Fall A=Ja und B=Nein, die untere den Fall A=Nein und B=Ja. Die jeweilige Einheit sendet ein positives Signal zu Z, und Z ist Ja. Für die anderen Eingaben (A=Ja und B=Ja sowie A=Nein und B=Nein) sind beide mittleren Einheiten Nein, Z empfängt kein positives Signal und ist damit auch Nein.





## Dies ist Informatik!

Der Konflikt-Detektor verarbeitet zwei Eingabewerte (Ja oder Nein) und liefert die Ausgabe Ja genau dann, wenn die beiden Eingabewerte unterschiedlich sind. Diese logische Funktion nennt man Exklusiv-Oder (XOR, Kontravalenz). Die erste in dieser Biberaufgabe beschriebene Maschine von Anna und Ben (zwei Schalter und eine Ausgabe-Einheit) ist eine vereinfachte Version eines *Perzeptrons*, das Frank Rosenblatt im Jahr 1957 beschrieben hat. Die Ausgabe-Einheit bildet eine Nervenzelle (Neuron) nach, die Eingabesignale verarbeiten kann und ein Ausgabesignal erzeugt. Mit einem Perzeptron kann man zwar die logischen Operationen Und und Oder implementieren, nicht aber das Exklusiv-Oder. Dazu benötigt man eine weitere Schicht von Schalteinheiten wie in der Lösung dieser Aufgabe. Erst in den 1980er Jahren hat man das erkannt (z.B. Rumelhart, Hinton & Williams 1986) und war dann (später) in der Lage, künstliche neuronale Netze zu programmieren, die ähnlich wie das menschliche Gehirn arbeiten und z. B. Kamerabilder auswerten und Objekte erkennen können. Die Informatik hat Methoden entwickelt, wie grosse neuronale Netze mit vielen Schichten und Einheiten ihre Berechnungen effizient durchführen können. Solche Netze bilden die Grundlage vieler aktueller KI-Systeme.

## Stichwörter und Webseiten

- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536 :  
<http://www.cs.toronto.edu/~hinton/absps/naturebp.pdf>
- Perzeptron, Überblick: <https://de.wikipedia.org/wiki/Perzeptron>
- Tutorial zur Programmierung eines Perzeptrons:  
[https://neuromant.de/2018/11/25/Tutorial\\_Das-Perzeptron/](https://neuromant.de/2018/11/25/Tutorial_Das-Perzeptron/)
- Exklusive-Oder (Kontravalenz): <https://de.wikipedia.org/wiki/Kontravalenz>



## A. Aufgabenautoren

 Eslam AbdElAal	 Alisher Ikramov
 Akram Ahmed	 Thomas Ioannou
 Somayah Albaradei	 Filiz Kalelioğlu
 Laila Alharthi	 Merel Kämper
 Khairul Anwar	 Gohar Khachatryan
 James Atlas	 Jihye Kim
 Leonardo Barichello	 Mhairi King
 Wilfried Baumann	 Jia-Ling Koh
 Tim Bell	 Sophie Koh
 Leonardo Cavalcante	 Víctor Koleszar
 Diego César	 Taina Lehtimäki
 Zaheer Chothia	 Marielle Léonard
 Eimear Colreavy	 Carlos Luna
 Raluca Constantinescu	 Michael Weigend
 Kris Coolsaet	 Yong Mao
 Christian Datzko	 Madhavan Mukund
 Justina Dauksaite	 Natalia Natalia
 Nora A. Escherle	 Tom Naughton
 Georgios Fesakis	 Zsuzsa Pluhár
 Gerald Futschek	 Wolfgang Pohl
 Emily Gates	 Estela Ramić
 Adam Grodeck	 Omar Colon Reyes
 Juan Gutiérrez	 Chris Roffey
 Tracy Henderson	 Karima Sayeh
 Josefine Hiebler	 Kirsten Schlüter
 Mathias Hiron	 Margareta Schlüter



 Eljakim Schrijvers

 Giovanni Serafini

 Vipul Shah

 Rostyslav Shpakovych

 Jacqueline Staub

 Alieke Stijf

 Marianne Thut

 Monika Tomcsányiová

 Ahto Truu

 Laura Ungureanu

 Jiří Vaníček

 Michael Weigend

 Manuel Wettstein

 Kyra Willekes



## B. Akademische Partner

**ABZ**

AUSBILDUNGS- UND BERATUNGSZENTRUM  
FÜR INFORMATIKUNTERRICHT

<http://www.abz.inf.ethz.ch/>

Ausbildungs- und Beratungszentrum für Informatikunterricht  
der ETH Zürich.

**hep/** haute  
école  
pédagogique  
vaud

<http://www.hepl.ch/>

Haute école pédagogique du canton de Vaud

Scuola universitaria professionale  
della Svizzera italiana

<http://www.supsi.ch/home/supsi.html>

La Scuola universitaria professionale della Svizzera italiana  
(SUPSI)

**SUPSI**



## C. Sponsoring

### HASLERSTIFTUNG

<http://www.haslerstiftung.ch/>

Stiftungszweck der Hasler Stiftung ist die Förderung der Informations- und Kommunikationstechnologie (IKT) zum Wohl und Nutzen des Denk- und Arbeitsplatzes Schweiz. Die Stiftung will aktiv dazu beitragen, dass die Schweiz in Wissenschaft und Technologie auch in Zukunft eine führende Stellung innehat.



Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich



<http://www.ubs.com/>

Wealth Management IT and UBS Switzerland IT



<http://www.verkehrshaus.ch/>



i-factory (Verkehrshaus Luzern)

Die i-factory bietet ein anschauliches und interaktives Erproben von vier Grundtechniken der Informatik und ermöglicht damit einen Erstkontakt mit Informatik als Kulturtechnik. Im optischen Zentrum der i-factory stehen Anwendungsbeispiele zur Informatik aus dem Alltag und insbesondere aus der Verkehrswelt in Form von authentischen Bildern, Filmbeiträgen und Computer-Animationen. Diese Beispiele schlagen die Brücke zwischen der spielerischen Auseinandersetzung in der i-factory und der realen Welt.



<http://senarclens.com/>

Senarclens Leu & Partner



## D. Weiterführende Angebote



IT Feuer: <https://it-feuer.ch/>

In der Schweiz engagieren sich zahlreiche Organisationen für die Nachwuchsförderung in Informatik. Die Initiative «IT-Feuer» möchte diese vorhandenen Kräfte bündeln und einen Beitrag leisten, das Thema in der Öffentlichkeit schweizweit bekannter zu machen. Das IT-Feuer präsentiert eine grosse Palette an Angeboten für Lehrpersonen sowie Schüler\*innen und Schulklassen.

### Das Lehrmittel zum Informatik-Biber

#### Module

Verkehr – Optimieren

Musik – Komprimieren

Geheime Botschaften – Verschlüsseln

Internet – Routing

Apps

Auszeichnungssprachen

<http://informatik-biber.ch/einleitung/>

Das Lehrmittel zum Biber-Wettbewerb ist ein vom SVIA, dem schweizerischen Verein für Informatik in der Ausbildung, initiiertes Projekt und hat die Förderung der Informatik in der Sekundarstufe I zum Ziel.

Das Lehrmittel bringt Jugendlichen auf niederschwellige Weise Konzepte der Informatik näher und zeigt dadurch auf, dass die Informatikbranche vielseitige und spannende Berufsperspektiven bietet.

Lehrpersonen der Sekundarstufe I und weiteren interessierten Lehrkräften steht das Lehrmittel als Ressource zur Vor- und Nachbereitung des Wettbewerbs kostenlos zur Verfügung.

Die sechs Unterrichtseinheiten des Lehrmittels wurden seit Juni 2012 von der LerNetz AG in Zusammenarbeit mit dem Fachdidaktiker und Dozenten Dr. Martin Guggisberg der PH FHNW entwickelt. Das Angebot wurde zweisprachig (Deutsch und Französisch) entwickelt.



CoetryLab: <https://www.coetry-lab.org/>

Das Team des CoetryLab möchte Kindern und Jugendlichen den Zugang zum Programmieren und zu Medien ermöglichen. Das CoetryLab soll die Anlaufstelle ausserschulischen Experimentierens und Gestaltens sein und allen die Coding-Welt eröffnen. Eigene Ideen können kreativ umgesetzt und im Team oder alleine Webseiten, Apps, Games und vieles mehr entwickelt werden.



Roteco: <https://www.roteco.ch/de/>

Das ROTECO Projekt bildet eine Community für und mit Lehrpersonen, welche Schülerinnen und Schüler auf die digitale Gesellschaft vorbereiten möchten. Lehrpersonen können auf dieser Plattform Erfahrungen austauschen, erhalten Informationen zu den neusten Kursen und Workshops und finden Aktivitäten, welche sich direkt in den Unterricht integrieren lassen.



I learn it: <http://ilearnit.ch/>

In thematischen Modulen können Kinder und Jugendliche auf dieser Website einen Aspekt der Informatik auf deutsch und französisch selbständig entdecken und damit experimentieren. Derzeit sind sechs Module verfügbar.

010100110101011001001001  
010000010010110101010011  
010100110100100101000101  
001011010101001101010011  
010010010100100100100001

**SV!A**

[www.svia-ssie-ssii.ch](http://www.svia-ssie-ssii.ch)  
schweizerischer vereinfürinformatikind  
erausbildung//société suisse pour l'infor  
matique dans l'enseignement//società sviz  
zera per l'informaticanell'insegnamento

Werden Sie SVIA Mitglied – <http://svia-ssie-ssii.ch/svia/mitgliedschaft> und unterstützen Sie damit den Informatik-Biber.

Ordentliches Mitglied des SVIA kann werden, wer an einer schweizerischen Primarschule, Sekundarschule, Mittelschule, Berufsschule, Hochschule oder in der übrigen beruflichen Aus- und Weiterbildung unterrichtet.

Als Kollektivmitglieder können Schulen, Vereine oder andere Organisationen aufgenommen werden.