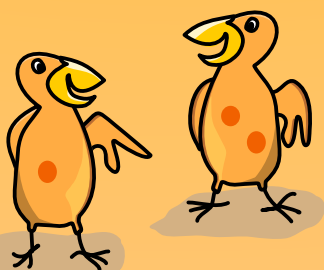




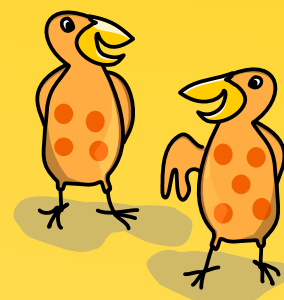
**INFORMATIK-BIBER SCHWEIZ**  
**CASTOR INFORMATIQUE SUISSE**  
**CASTORO INFORMATICO SVIZZERA**

## Aufgaben und Lösungen 2021

### Alle Stufen



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



Herausgeber:

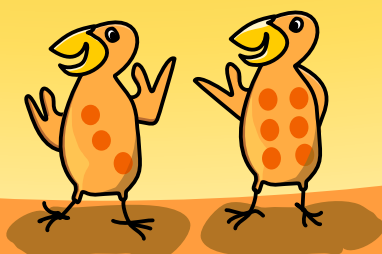
Susanne Datzko, Fabian Frei,  
Jean-Philippe Pellet



010100110101011001001001  
010000010010110101010011  
010100110100100101000101  
001011010101001101010011  
010010010100100100100001

# 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 2021

Masiar Babazadeh, Susanne Datzko, Fabian Frei, Martin Guggisberg, Gabriel Parriaux, Jean-Philippe Pellet

Projektleitung: Nora A. Escherle

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

Juraj Hromkovič, Michael Barot, Christian Datzko, Jens Gallenbacher, Dennis Komm, Regula Lacher, Peter Rossmann: ETH Zürich, Ausbildungs- und Beratungszentrum für Informatikunterricht  
Bernadette Spieler: Pädagogische Hochschule Zürich

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

Valentina Dagienė, Tomas Šiaulys, Vaidotas Kinčius: Bebras.org

Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend: Bundesweite Informatikwettbewerbe (BWINF), Deutschland

Wilfried Baumann, Liam Baumann, Anoki Eischer, Thomas Galler, Benjamin Hirsch, Martin Kandlhofer, Katharina Resch-Schobel: Österreichische Computer Gesellschaft

Gerald Futschek, Florentina Voboril: Technische Universität Wien

Zsuzsa Pluhár: ELTE Informatikai Kar, Ungarn

Michal Winzcer: Comenius University, Slowakei

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

Eljakim Schrijvers, Justina Dauksaite, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers, Alieke Stijf, Kyra Willekes: [cuttle.org](http://cuttle.org), Niederlande

Chris Roffey: UK Bebras Administrator, Vereinigtes Königreich

Für den Support während den Wettbewerbswochen danken wir:

Hanspeter Erni: Schulleitung Sekundarschule Rickenbach

Christoph Frei: Chragokyberneticks (Logo Informatik-Biber Schweiz)

Dr. Andrea Leu, Maggie Winter, Brigitte Manz-Brunner: Senarclens Leu + Partner AG

*Diese Broschüren sind dem Andenken an Martin Guggisberg gewidmet.*

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 2021 wurde vom Schweizerischen Verein für Informatik in der Ausbildung SVIA durchgeführt und von der Hasler Stiftung unterstützt.

## HASLERSTIFTUNG

Dieses Aufgabenheft wurde am 24. August 2022 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 2021 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. 114 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 im Rahmen des Förderprogramms FIT in IT 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 2021 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

In den Altersklassen 3 und 4 hatten 9 Aufgaben zu lösen, nämlich aus den drei Schwierigkeitsstufen leicht, mittel und schwer jeweils drei. 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, nämlich 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.

## **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 2021 . . . . .	i
Vorwort . . . . .	iii
Inhaltsverzeichnis . . . . .	v
1. Mikas Stempel . . . . .	1
2. Richtiges Trikot . . . . .	5
3. Brückenbau . . . . .	7
4. Lieblingsgeschenk . . . . .	9
5. Schlüsselanhänger . . . . .	11
6. Baum fällt! . . . . .	15
7. Schildkrötenpfad . . . . .	17
8. Wasser auf die Mühle . . . . .	21
9. Kugelspiel . . . . .	25
10. Sack mit Münzen . . . . .	27
11. Treffen sie sich? . . . . .	31
12. Dottis . . . . .	35
13. Erdbeerklau . . . . .	39
14. Tiere beobachten . . . . .	43
15. Geburtstagsrätsel . . . . .	47
16. Lieblingsgeschenk . . . . .	51
17. Rette den Baum! . . . . .	55
18. Bibliothek . . . . .	59
19. Fliesenmuster . . . . .	61
20. SOS aus den Bergen . . . . .	63
21. Schichte nach Dichte! . . . . .	67
22. Es presst! . . . . .	71
23. Theklas Netze . . . . .	75



24. Frucht auf Frucht . . . . .	79
25. Kletteräffchen Koko . . . . .	83
26. Verflixte Pulte . . . . .	87
27. Murrelband . . . . .	91
28. Biber-Arbeit . . . . .	95
29. Murrelzahlen . . . . .	97
30. Teamwork . . . . .	101
31. Zählen durch Nicken . . . . .	105
32. Beaver Sort . . . . .	109
33. Die Clans von Beavaria . . . . .	111
A. Aufgabenautoren . . . . .	114
B. Sponsoring: Wettbewerb 2021 . . . . .	116
C. Weiterführende Angebote . . . . .	119





# 1. Mikas Stempel

Mika hat vier Stempel mit verschiedenen Figuren. Jeden nimmt sie einmal in die Hand und stempelt damit zweimal. Es entsteht folgendes Bild:



*Welchen Stempel hat Mika zuerst verwendet?*

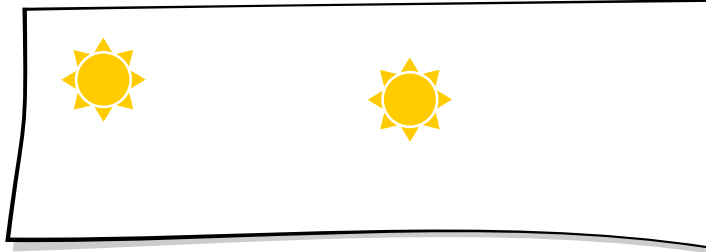




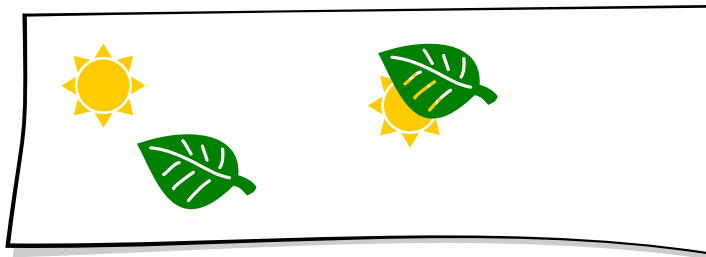
## Lösung

Die richtige Antwort ist C: 

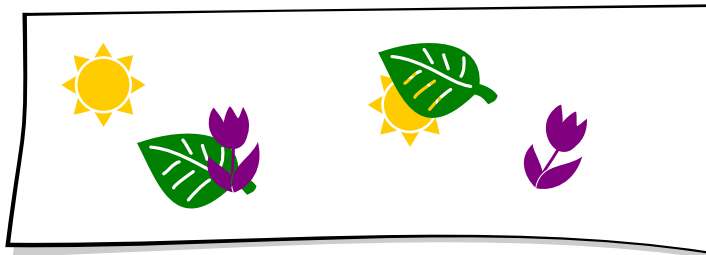
Die Reihenfolge, in der Mika die Stempel verwendet, erkennt man daran, welche Figuren andere Figuren überlagern oder von anderen Figuren überlagert werden. Die Sonne wird von allen anderen Figuren überlagert. Mika hat somit den Stempel mit der Sonne zuerst verwendet.



Das Blatt überlagert die Sonne, aber wird sowohl von der Blume als auch von dem Haus überlagert. Mika hat somit den Stempel mit dem Blatt als zweites verwendet:



Die Blume überlagert wiederum sowohl das Blatt also auch die Sonne.



Es ist jedoch nicht möglich, dass Mika den Stempel mit der Blume zuletzt verwendet hat: An zwei Stellen wird die Blume klar sichtbar von dem Haus überlagert. Somit hat Mika den Stempel mit der Blume als drittes verwendet und den Stempel mit dem Haus zuletzt.





## Dies ist Informatik!

Mikas Bild ist gewissermaßen ein Abbild oder eine *Modellierung* der Realität anhand von gestempelten Figuren auf vier verschiedenen Ebenen: eine Ebene mit Sonnen, eine Ebene mit Blättern, eine Ebene mit Blumen und eine Ebene mit Häusern. Die Überlagerung der Figuren verschiedener Ebenen ermöglicht es Mika, auf einer (zweidimensionalen) Fläche von Papier die Illusion von Tiefe und (dreidimensionaler) Räumlichkeit zu erzeugen.

Bei einer Modellierung werden üblicherweise nur Aspekte abgebildet, die notwendig sind, um eine bestimmte Aufgabe oder Funktion zu erfüllen. Die Realität wird also vereinfacht dargestellt. Modellierung ist ein wichtiges Prinzip in der Informatik.

So erstellen Informatiker:innen beispielsweise manche Computerprogramme mit dem Zweck, Teile der realen Welt möglichst schnell und präzise zu untersuchen und Erkenntnisse über diese zu gewinnen. Um solch ein Programm zu erstellen, müssen sich Informatiker:innen immer erst ein Gedankenmodell überlegen, welches die für sie wesentlichen Elemente der realen Welt enthält. Dieses Gedankenmodell ermöglicht es, ein reales Modell in Form einer Skizze oder eines Computerprogramms zu erstellen. Mit Hilfe dieses Computerprogramms, also einer Modellierung eines Teils der realen Welt, können nun Erkenntnisse über die reale Welt erlangt werden.

## Stichwörter und Webseiten

- Modellierung: [https://files.ifi.uzh.ch/rerg/amadeus/teaching/courses/infII\\_-ss04/kapitel\\_01.pdf](https://files.ifi.uzh.ch/rerg/amadeus/teaching/courses/infII_-ss04/kapitel_01.pdf)
- Ebenen, Bildbearbeitung: <https://de.wikipedia.org/wiki/Ebenentechnik>





## 2. Richtiges Trikot

Anne packt ihre Tasche, um an den Match zu gehen. Heute muss sie das Trikot mit hellen Ärmeln und schwarzem Kragen, aber ohne Streifen mitbringen.



Welches Trikot packt sie in ihre Tasche?





## Lösung



Antwort B ist richtig.



und



Das Trikot A und D passen heute nicht, weil sie schwarze Ärmel haben, und schwarz ist nicht hell.



Das Trikot C hat Streifen, deswegen passt es nicht für den heutigen Match.

Das Trikot B ist perfekt für heute: Es hat helle Ärmel und einen schwarzen Kragen und keine Streifen.

## Dies ist Informatik!

In dieser Biberaufgabe solltest du aus einer Menge von Dingen das Ding herausuchen, das bestimmte *Bedingungen* erfüllt oder eben nicht erfüllt.

Mehrere *Teilbedingungen*, wie zum Beispiel die Farbe der Ärmel und das Stoffmuster wurden hier definiert und zu einer *Gesamtbedingung* verknüpft. Informatiker verwenden für diese Verknüpfungen *logische Operatoren*.

Wenn alle Teilbedingungen gleichzeitig erfüllt sein müssen, verwendet man den Operator *UND*: «Die Farbe der Ärmel ist hell.» *UND* «die Farbe des Kragens ist schwarz.» Wenn es reicht, wenn mindestens eine von mehreren Teilbedingungen erfüllt ist, verwendet man den Operator *ODER*. Wenn eine Teilbedingung nicht erfüllt sein darf, kann man den Operator *NICHT* verwenden, zum Beispiel: *NICHT*(Das Trikot hat Streifen.)

Für das Suchen in Datenbanken kann man mit Hilfe von Abfragesprachen sehr komplexe Bedingungen formulieren.

Die Bedingungen selbst müssen dafür klar definiert sein. Die Bedingung, dass die Ärmel hell sein müssen, kann zum Beispiel unklar sein. In einem solchen Fall schreiben Informatiker ein Programm oder eine Funktion, welche überprüft, ob eine Farbe hell ist oder nicht. Hierfür benötigen die Informatiker eine klare Definition, wann genau eine Farbe «hell» ist, sonst ist es nicht möglich, ein funktionierendes Programm zu schreiben.

## Stichwörter und Webseiten

- Boolesche Algebra: [https://de.wikipedia.org/wiki/Boolesche\\_Algebra](https://de.wikipedia.org/wiki/Boolesche_Algebra)
- Logische Verknüpfungen: [https://de.wikipedia.org/wiki/Logische\\_Verknüpfung](https://de.wikipedia.org/wiki/Logische_Verknüpfung)



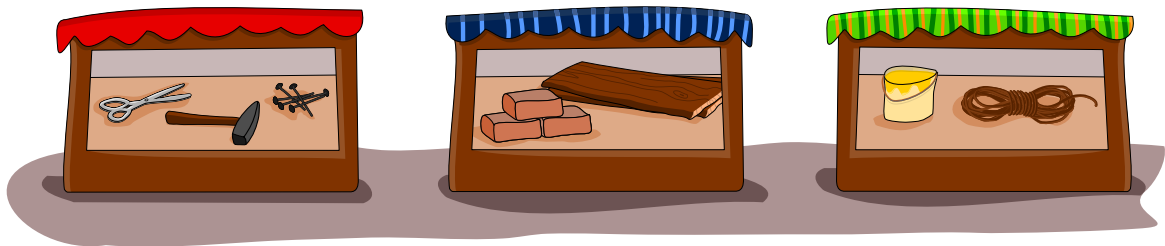
### 3. Brückenbau

Bella möchte eine Brücke bauen, über einen Bach. Sie braucht: Hammer, Nägel, Bretter und ein Seil. Im Keller findet sie einen Hammer und ein Seil.



Die anderen Sachen muss sie einkaufen. Unten siehst du drei Geschäfte, und was sie verkaufen.

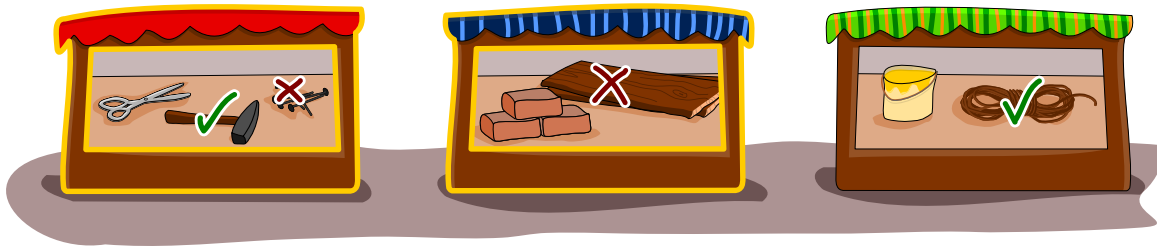
Wo kann Bella die anderen Sachen einkaufen?





## Lösung

So ist es richtig:



## Dies ist Informatik!

Die Geschäfte in dieser Biberaufgabe verkaufen insgesamt sieben Sachen: Schere, Hammer, Nägel, Ziegel, Bretter, Seil und Eimer. Das ist eine ganze Menge! Die beiden Sachen, die Bella einkaufen muss, sind ein *Teil dieser Menge*. Das kann man so malen: Man zeigt alle Sachen der gesamten Menge und markiert für jede Sache, ob sie zu Bellas Einkaufs-Teilmenge gehört oder nicht:



Genauso kann man malen, was die Geschäfte verkaufen, zum Beispiel das Geschäft links:



So sieht man auf den ersten Blick, was Bella im Geschäft links einkaufen kann: Die Nägel haben einen grünen Haken in der Einkaufs-Teilmenge und in der Verkaufs-Teilmenge.

Auch Computerprogramme müssen häufig Mengen vergleichen. Das können sie so machen wie oben gezeigt: Für jede Sache, die vorkommen kann, wird ein *Bit* benötigt. In einem Bit kann ein Computer einen von zwei Werten speichern, wie «ja» und «nein». In diesem Fall wird gespeichert, ob die Sache zu einer Menge gehört («ja») oder nicht («nein»). Dann kann ein Programm zwei Mengen so vergleichen: Es prüft, ob das Bit für eine Sache in der einen Menge «ja» ist und das Bit für die gleiche Sache in der anderen Menge auch «ja» ist. Eine solche Prüfung zweier Bits kann ein Computer besonders schnell erledigen. In der Informatik ist die Beschreibung von Mengen mit Bits deshalb sehr verbreitet.

## Stichwörter und Webseiten

- Menge: [https://de.wikipedia.org/wiki/Menge\\_\(Datenstruktur\)](https://de.wikipedia.org/wiki/Menge_(Datenstruktur))
- Bits: <https://de.wikipedia.org/wiki/Bit>
- Bitstrings: <https://de.wikipedia.org/wiki/Bitkette>





## 4. Lieblingsgeschenk

Die Biberfamilie hat drei Geschenke für ihre drei Kinder. Jedes Kind nennt zuerst sein Lieblingsgeschenk und dann das zweitliebste. Die Geschenke sollen richtig zugeteilt werden:

1. Möglichst viele Kinder sollen ihr Lieblingsgeschenk bekommen.
2. Die übrigen sollen das zweitliebste bekommen.

*Gib den Kindern die richtigen Geschenke.*



1:  , 2: 



1:  , 2: 

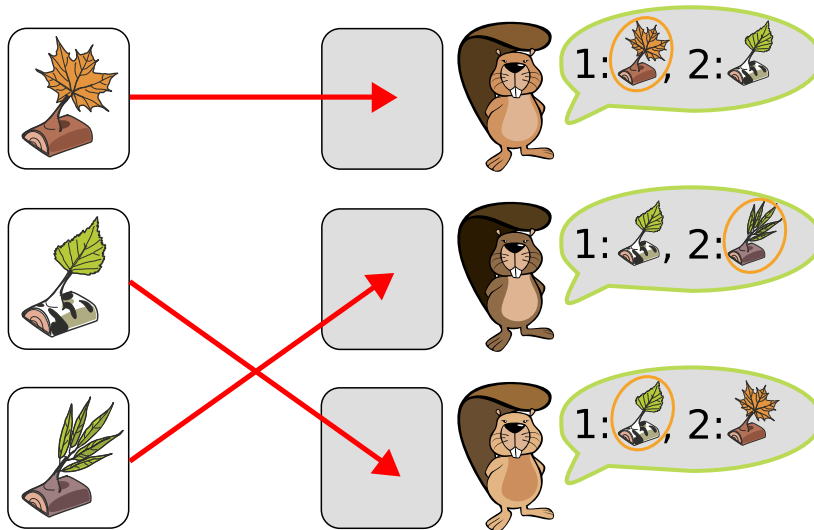


1:  , 2: 



## Lösung

Hier ist die einzige Zuordnung der Geschenke, die beide Bedingungen erfüllt.



Das dritte Geschenk wünscht sich nur der zweite Biber, deshalb muss er es bekommen. Ansonsten würde ein anderer etwas bekommen, das weder das Lieblingsgeschenk noch das zweitliebste ist. Für die anderen beiden Geschenke ist die Aufteilung dann klar: Jeder kann sein Lieblingsgeschenk bekommen.

## Dies ist Informatik!

Bei dieser Aufgabe handelt sich um ein eindeutiges *Zuordnungsproblem*: Wir möchten die Geschenke so zuordnen, dass alle Kinder ein Geschenk bekommen und es kein Kind ohne Geschenk gibt. Dabei haben die Kinder nicht nur einen einzelnen Wunsch, sondern geben eine Reihenfolge von Vorlieben an. Solche Zuordnungsprobleme mit Reihenfolgen von Vorlieben können sehr kompliziert werden. Die Informatik hilft uns dabei, solche Problem möglichst rasch zu lösen.

Eine Möglichkeit ist den Zuordnungen einen Wert zu geben: Das Lieblingsgeschenk hat Wert 1 und das zweitliebste Geschenk den Wert 2. Wir möchten den Gesamtwert minimieren. Eine *Zuordnung* (engl. *Matching*) ist *optimal*, wenn es keine andere Zuordnung mit mehr erfüllten Erstplatzierungen gibt. In der Informatik wird so eine Zuordnung als *Rang-Maximal-Matching* bezeichnet. Es gibt viele Matching-Probleme. Eines davon wird als *Problem der stabilen Paarung* (engl. *Stable Marriage Problem*) bezeichnet. Klingt interessant? Informatik ist eben ein ganz vielseitiges Fach.

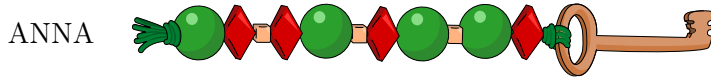
## Stichwörter und Webseiten

- Zuordnungsproblem: <https://de.wikipedia.org/wiki/Zuordnungsproblem>
- Rang-Wahl: <https://de.wikipedia.org/wiki/Rang-Wahl>

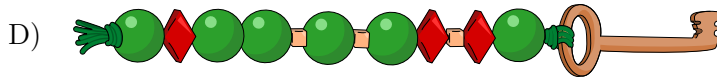
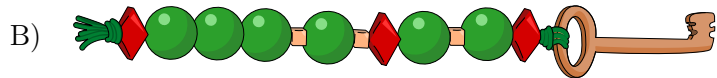
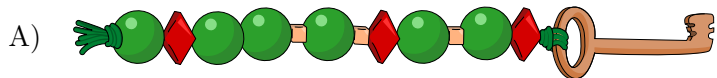


## 5. Schlüsselanhänger

ANNA, BELLA und LENA machen Schlüsselanhänger mit ihren Namen. Für die Buchstaben verwenden sie zwei Sorten von Perlen: ● und ◆. Einzelne Buchstaben trennen sie mit dieser Perle: ■.



Welchen Schlüsselanhänger hat LENA gemacht?





## Lösung

Die richtige Antwort ist A)

Das Wort LENA beginnt mit L. In BELLA ist L der dritte und vierte Buchstabe und wir erkennen in BELLA für L die Perlenfolge . Nur Antworten A) und D) beginnen mit diesem Buchstaben und können die Lösung sein. Der zweite Buchstabe von LENA, also E, ist auch der zweite von BELLA, dort finden wir im Anhänger jeweils die Perle . Sowohl bei A) als auch bei D) haben wir als zweiten Buchstaben, also könnten sie beide weiterhin die Lösung sein. Als nächstes wollen wir die Perlen für den Buchstaben N herausfinden. In ANNAs Anhänger finden wir für N die Perlenfolge . Und das sind nur in der Lösung A) die nächsten Perlen.

Eine andere Möglichkeit, die Perlen für LENAs Anhänger herauszufinden, ist es, eine Tabelle mit den Perlen für die bereits bekannten Buchstaben zu erstellen. Von ANNAs Anhänger finden wir für A die Perlenfolge und für N . Von BELLAs Anhänger finden wir die Perlenfolge für B: , für E: und L: .

Buchstaben	Perlen
A	
N	
B	
E	
L	

Daher können wir den Anhänger für LENA aus den Perlen , und formen, wenn wir zusätzlich die einzelnen Buchstaben mit der Perle trennen. So erhalten wir den Anhänger , was die Antwort A) ist. Wenn wir die anderen gegebenen Anhänger mit Hilfe der Codierungstabelle entschlüsseln, finden wir für B) BENA, C) NENA und für D) LEAN.

## Dies ist Informatik!

Information wird *codiert*, um Nachrichten unter bestimmten Bedingungen übertragen zu können oder um die Information geheim (*verschlüsselt*) zu übertragen. In dieser Aufgabe basiert die Codierung auf dem Morsecode. Der Punkt • des Morsecode wird durch die runde Perle dargestellt und der Strich – durch . Der Buchstabe A ist im Morsecode •–, also im Perlencode . Um beliebige Texte zu codieren, brauchen wir Codes für alle Buchstaben des Alphabets. Der Morsecode entstand im 19. Jahrhundert. Samuel Morse erfand 1837 einen einfachen elektromagnetischen Schreibtelegraphen. Der verwendete Code umfasste damals nur die zehn Ziffern 0 bis 9; die übertragenen Zahlen mussten mit Hilfe einer Tabelle in Buchstaben und Wörter übersetzt werden. Alfred Lewis Vail, ein Mitarbeiter von Morse, entwickelte ab 1838 den ersten Code, der auch Buchstaben umfasste. Der Code wurde entwickelt, um Texte akustisch, optisch oder elektrisch über Telegraphenleitungen zu übertragen. Ein Punkt ist eine kurze Übertragungsdauer und ein Strich eine dreimal so lange. Zwischen den Buchstaben muss eine Pause eingehalten werden. Eine längere Pause trennt Wörter. Noch heute wird



der Morsecode für das Notsignal SOS verwendet. SOS im Morsecode **••• — — — •••** (3x kurz, 3x lang, 3x kurz) kann sehr leicht durch Rufen, Klopfen, oder mit der Taschenlampe übertragen werden. In der elektronischen Datenverarbeitung werden Zeichen über einen Zahlenwert codiert, um sie zu übertragen oder zu speichern.

## Stichwörter und Webseiten

- Zeichenkodierung: <https://de.wikipedia.org/wiki/Zeichenkodierung>
- Morsecode: <https://de.wikipedia.org/wiki/Morsecode>





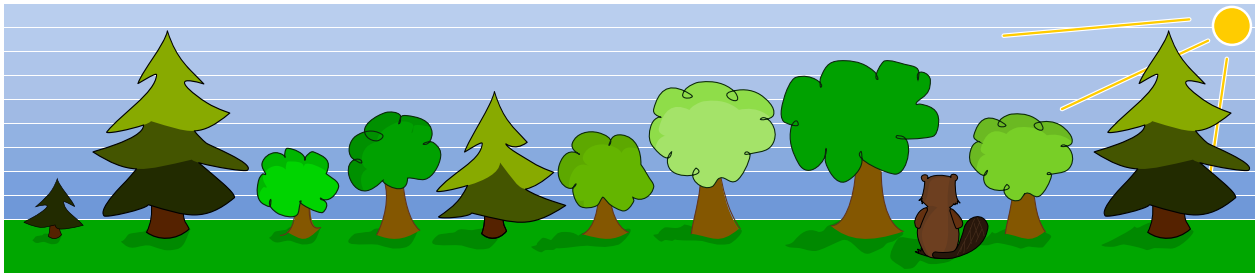
## 6. Baum fällt!

Ein Biber möchte einen Damm bauen. Damit er immer die richtigen Bäume fällt, hat er sich zwei Bedingungen überlegt:

Er fällt einen Baum nur dann, wenn ...

- ... direkt links daneben ein kleinerer Baum wächst und ...
- ... direkt rechts daneben ein grösserer Baum wächst.

*Welche Bäume wird der Biber fällen?*



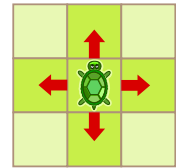






## 7. Schildkrötenpfad

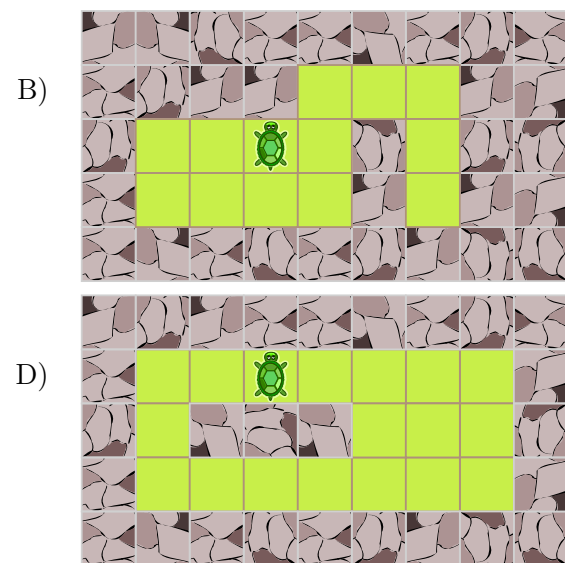
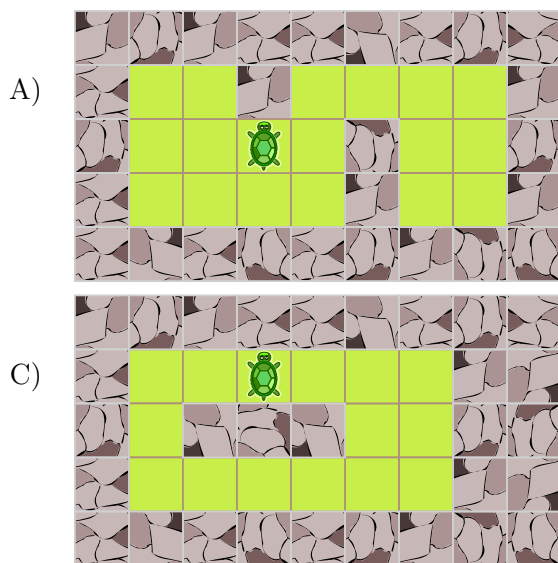
Eine Schildkröte soll verschiedene Gärten abgrasen. Jeder Garten ist in Quadrate unterteilt, die entweder mit Gras oder Steinen bedeckt sind. Die Schildkröte kann keine Steine überqueren. Sie kann sich aber von einem Grasfeld zu einem anderen Grasfeld direkt daneben bewegen.

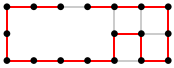


Die Schildkröte soll die Gärten vollständig abgrasen. Sie startet auf dem Feld, auf dem sie im Bild steht. Am Ende soll sie in jedem Grasfeld genau einmal gewesen sein.

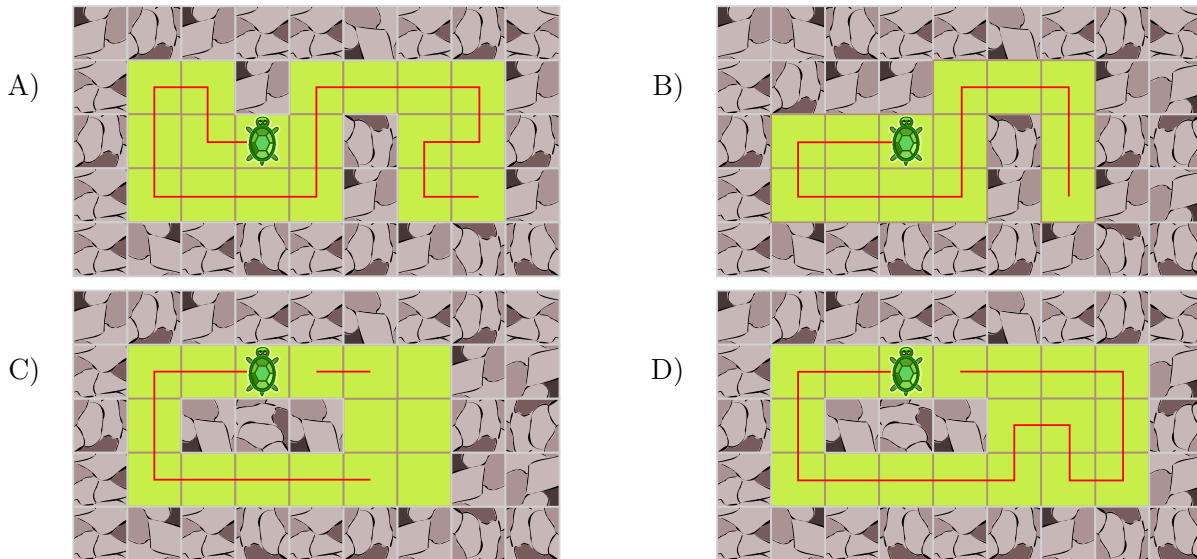
Leider kann die Schildkröte so einen der Gärten nicht vollständig abgrasen.

*Welcher ist es?*





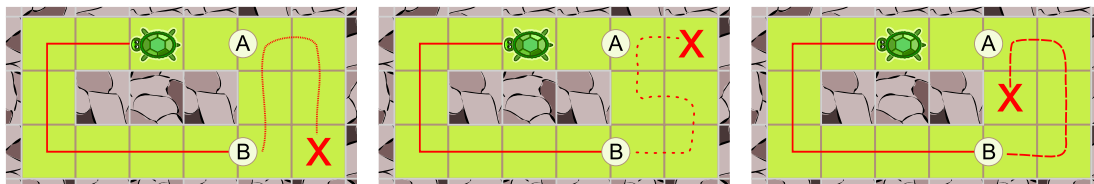
## Lösung



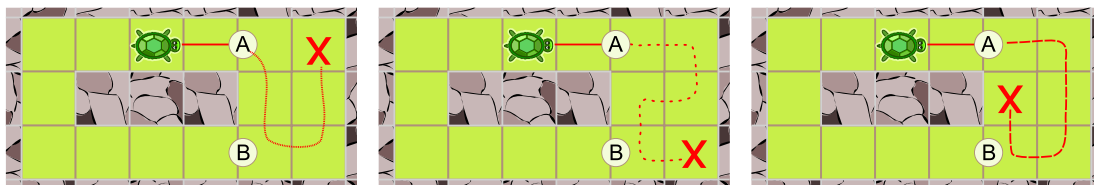
Die Gärten A, B und D kann die Schildkröte vollständig abgrasen.

Den Garten C kann die Schildkröte so nicht abgrasen. Die Schildkröte hat von ihrem Startpunkt aus nur 2 Möglichkeiten:

- Geht sie zuerst nach links, kommt sie zu Punkt B. Von dort müsste sie die 6 Felder rechts so abgrasen, dass sie am Ende Punkt A erreicht. Aber keiner der von B aus möglichen Pfade endet bei A.



- Geht die Schildkröte zuerst nach rechts, kommt sie zu A und müsste die 6 Felder so abgrasen, dass sie am Ende Punkt B erreicht. Jetzt kann man gleich argumentieren wie vorher, man muss nur oben und unten vertauschen. Es gibt also auch so keinen geeigneten Pfad.

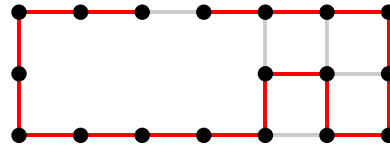


## Dies ist Informatik!

Die Schildkröte soll einen Weg durch ihren Garten finden und dabei jedes Grasfeld genau einmal besuchen. Das Problem hinter dieser Aufgabe ist ein sogenanntes *Hamiltonpfadproblem*.



Ein Schildkrötengarten (also die Grasquadrate) kann so betrachtet werden: Jedes Grasquadrat ist ein *Knoten* (dargestellt als Punkt). Der Garten D sieht dann so aus:



Für derartige Strukturen (von Informatikern, aber auch von Mathematikern *Graphen* genannt) stellte sich im 19. Jahrhundert Sir William Rowan Hamilton die Frage, ob es einen Pfad entlang der Kanten gibt, der jeden Knoten genau einmal besucht. Einen solchen Pfad nennt man deshalb auch *Hamilton-Pfad*. Die Frage, ob es überhaupt einen Hamilton-Pfad gibt, ist im Allgemeinen sehr schwer zu lösen. Niemand kennt einen *Algorithmus*, der für beliebige Graphen effizient (in mehr oder weniger nützlicher Frist) entscheiden könnte, ob es im vorgegebenen Graphen einen Hamilton-Pfad gibt oder nicht. Wir wissen auch nicht, ob es einen solchen effizienten Algorithmus überhaupt geben kann. Dies gilt für alle sogenannten *NP-vollständigen Probleme*, wovon das Hamilton-Pfad-Problem eines der berühmtesten ist.

## Stichwörter und Webseiten

- Graph, Hamiltonpfad: <https://de.wikipedia.org/wiki/Hamiltonkreisproblem>



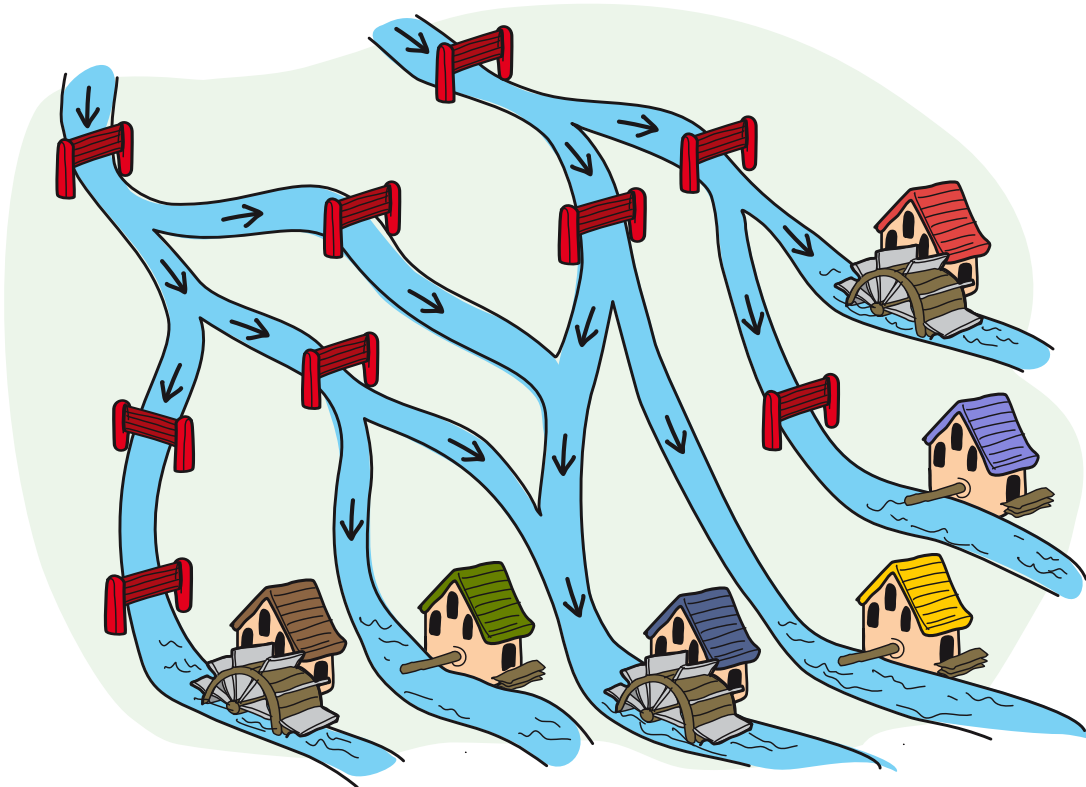


## 8. Wasser auf die Mühle

Müller Mert hat sechs Mühlen. Bei drei davon muss er noch das Mühlrad einbauen. Dafür darf kein Wasser mehr zu diesen Mühlen fließen. Zu den anderen Mühlen soll aber weiterhin Wasser fließen.

Das Wasser kann nur nach unten fließen. Ein geschlossener Schieber stoppt das Wasser.

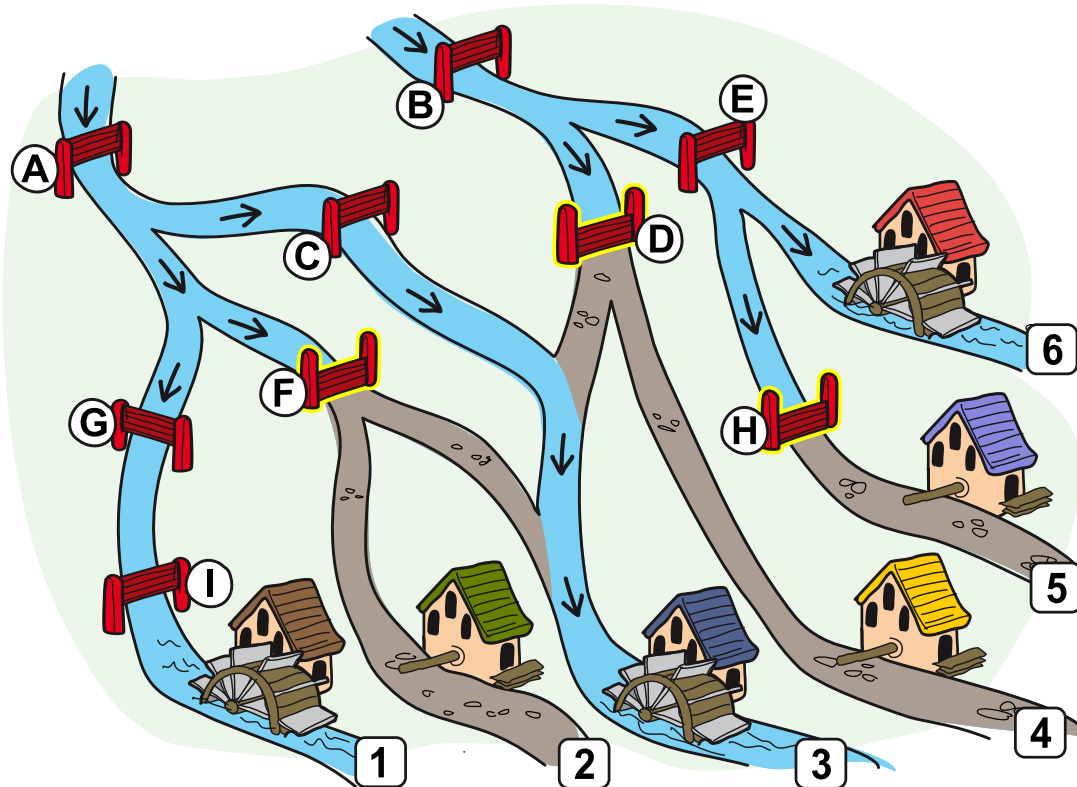
*Welche Schieber soll Mert schliessen?*





## Lösung

Die richtige Antwort ist: Es sind drei Schieber zu schliessen, die in der folgenden Zeichnung mit D, F und H bezeichnet wurden.



Dies ist die einzige Möglichkeit, mit der kein Wasser mehr zu den Mühlen 2, 4 und 5 ohne Mühlrad fließt, während die Mühlen 1, 3 und 6 weiter Wasser bekommen:

- Die Schieber A, G und I müssen alle offen bleiben, da sonst kein Wasser mehr zur Mühle 1 fließen würde.
- Die Schieber B und E müssen ebenfalls geöffnet bleiben, da sonst kein Wasser mehr zur Mühle 6 fließen würde.
- Weil die Schieber B und E offen bleiben, muss der Schieber H geschlossen werden, da sonst Wasser zur Mühle 5 fließen würde.
- Weil der Schieber A offen bleibt, muss der Schieber F geschlossen werden, da sonst Wasser zur Mühle 2 fließen würde.
- Weil der Schieber B offen bleibt, muss der Schieber D geschlossen werden, da sonst Wasser zur Mühle 4 fließen würde.
- Weil die Schieber D und F geschlossen werden, muss der Schieber C offen bleiben, da sonst kein Wasser mehr zur Mühle 3 fließen würde.



## Dies ist Informatik!

In dieser Aufgabe wird das Fließen des Wassers zu den Mühlen durch *Bedingungen* geregelt. Zum Beispiel fließt genau dann Wasser zur Mühle 6, wenn die beiden Schieber B und E offen stehen. Und hier noch ein zweites, etwas komplizierteres Beispiel: Zur Mühle 3 fließt genau dann Wasser, wenn mindestens eine oder gleich beide der folgenden Bedingungen erfüllt sind:

- Der Schieber A ist offen und einer der beiden Schieber C oder F ist offen.
- Die beiden Schieber B und D sind offen.

Solche zusammengesetzten Bedingungen werden mit den *logischen Operatoren* UND (als Symbol:  $\wedge$ ) bzw. ODER (als Symbol:  $\vee$ ) erzielt. Solche Operatoren verknüpfen Wahrheitswerte wie wahr oder falsch. Sind also A und B zwei Wahrheitswerte, so kann man angeben, welche Wahrheitswerte die zusammengesetzten Ausdrücke «A UND B» bzw. «A ODER B» haben:

A	B	A UND B	A ODER B
falsch	falsch	falsch	falsch
wahr	falsch	falsch	wahr
falsch	wahr	falsch	wahr
wahr	wahr	wahr	wahr

In der Informatik (und auch in der Mathematik) wird also die Aussage «A ODER B» auch dann als richtig angesehen, wenn beide, A und B, richtig sind. Die Aussage «Es fließt Wasser zur Mühle 6.» ist gleichbedeutend mit:

«Der Schieber B ist offen.» UND «Der Schieber E ist offen.».

Die Aussage «Es fließt Wasser zur Mühle 3.» aus unserem zweiten Beispiel ist gleichbedeutend mit:

(«Der Schieber A ist offen.» UND («Der Schieber C ist offen.» ODER «Der Schieber F ist offen.»))  
ODER («Der Schieber B ist offen.» UND «Der Schieber D ist offen.»).

Jedes UND und jedes ODER verbindet zwei Aussagen. Die Klammer machen klar, in welcher Reihenfolge die Aussagen verbunden werden. Beim Programmieren ist das korrekte Formulieren von Bedingungen wichtig. Die Verknüpfung mit den logischen Operatoren und Klammern ist nützlich, um komplexere Bedingungen zu formulieren. Sowohl bei Verzweigung mit Hilfe von `if` als auch bei `while`-Schleifen verwenden wir Bedingungen, um den Programmablauf zu steuern.

## Stichwörter und Webseiten

- Bedingte Anweisung: [https://de.wikipedia.org/wiki/Bedingte\\_Anweisung\\_und\\_Verzweigung](https://de.wikipedia.org/wiki/Bedingte_Anweisung_und_Verzweigung)
- Boolesche Variable: <https://de.wikipedia.org/wiki/Boolean>
- Boolesche Operatoren: [https://de.wikipedia.org/wiki/Boolescher\\_Operator](https://de.wikipedia.org/wiki/Boolescher_Operator)



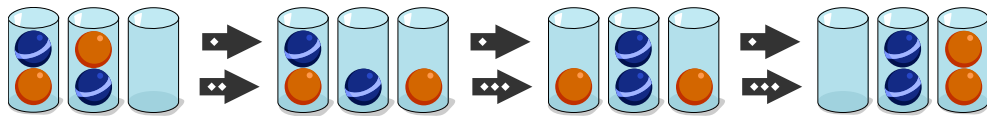




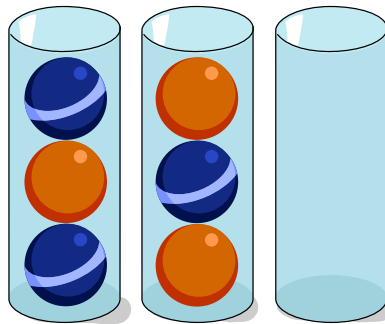
## 9. Kugelspiel

Die Biber möchten Kugeln nach ihrer Farbe ordnen. Am Ende sollen sich alle Kugeln in zwei Gläsern befinden. In einem Glas sollen die Kugeln dieselbe Farbe haben. Dabei sind diese drei Regeln zu befolgen:

- ➡ Regel 1: In einem Schritt kann nur die oberste Kugel eines Glases bewegt werden.
- ➡➡ Regel 2: Eine Kugel kann in ein leeres Glas bewegt werden.
- ➡➡➡ Regel 3: Eine Kugel kann in ein Glas bewegt werden, wenn dort noch Platz frei ist und die darunter liegende Kugel dieselbe Farbe hat.



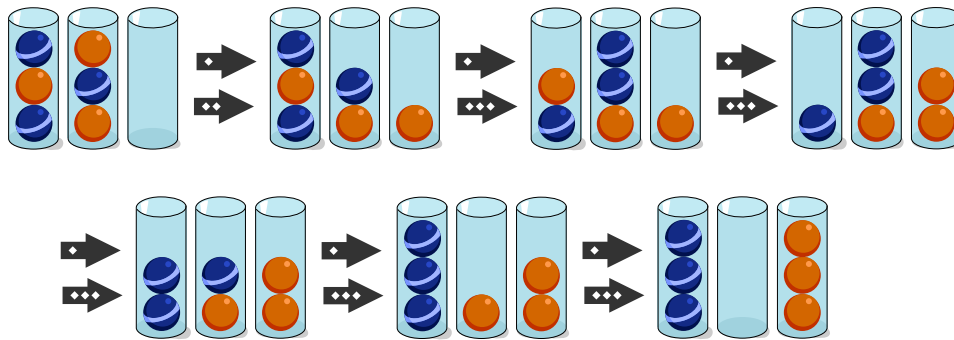
Ordne die Kugeln, indem du sie nach den drei Regeln bewegst.





## Lösung

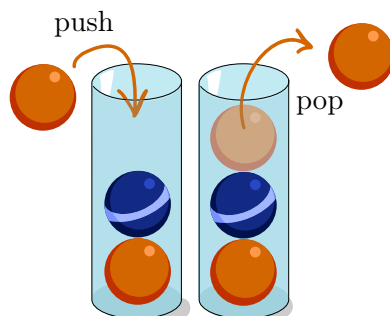
Die Kugeln können in der folgenden Reihenfolge bewegt werden:



Um die Kugeln neu zu ordnen, benötigt man mindestens 6 Schritte. Es gibt auch noch andere Möglichkeiten, die Kugeln in nur 6 Schritten neu zu ordnen.

## Dies ist Informatik!

In dieser Aufgabe bewegst du die Kugeln ähnlich wie der Computer in einem *Stapelspeicher* Daten verwaltet: Er kann nur *oben ein Element* (in der Aufgabe eine Kugel) *hinzufügen* (engl. *push*) und nur das *oberste Element wieder entfernen* (engl. *pop*).



Auf die unteren Elemente kann der Computer nur zugreifen, wenn zuerst die Kugeln oberhalb entfernt werden. Und das Element, das als letztes gespeichert wurde, wird der Computer wieder als erstes entfernen. Dieses Prinzip nennen die Informatiker *Last-in-First-out-Prinzip* (kurz *LIFO*).

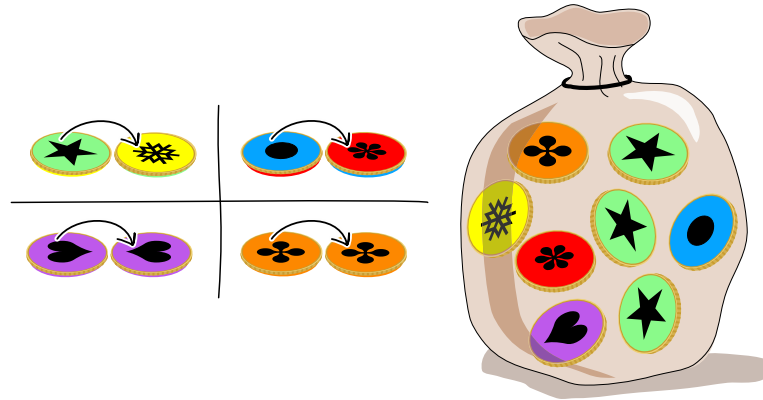
## Stichwörter und Webseiten

- Stapelspeicher, Kellerspeicher: <https://de.wikipedia.org/wiki/Stapelspeicher>



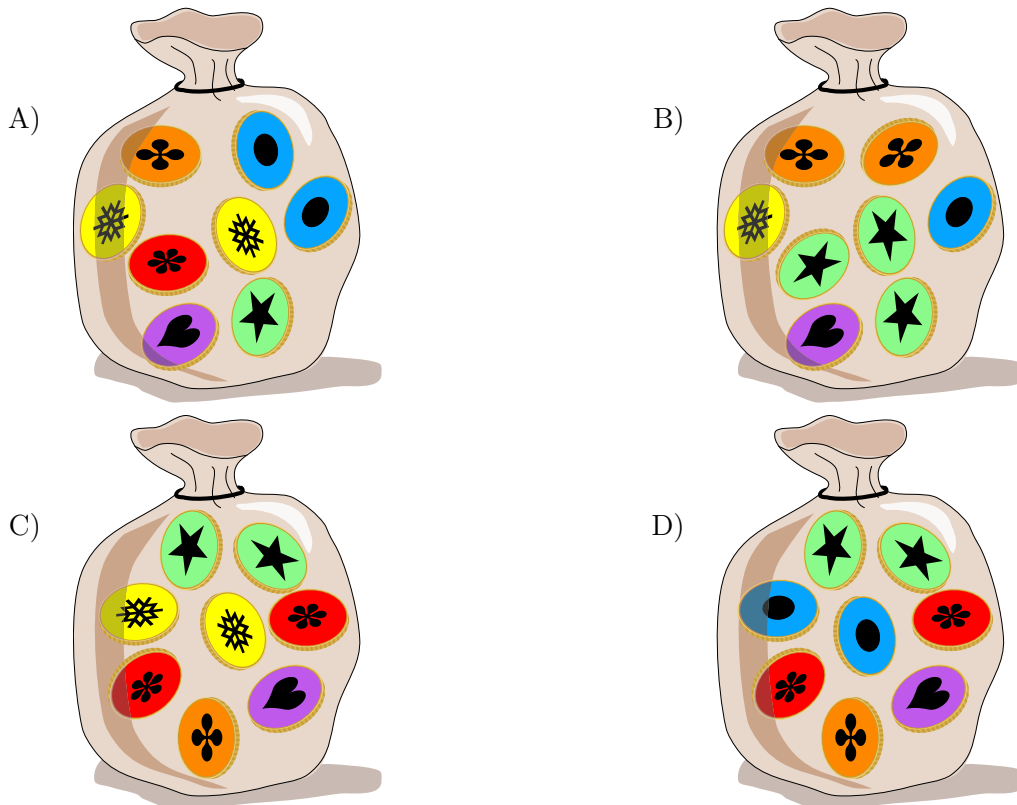
## 10. Sack mit Münzen

In Emils Land gibt es 4 verschiedene Arten von Münzen. Du siehst hier die beiden Seiten jeder Münzenart und auch Emils Sack mit seinen Münzen.



Sein Sack mit Münzen wurde nun geschüttelt.

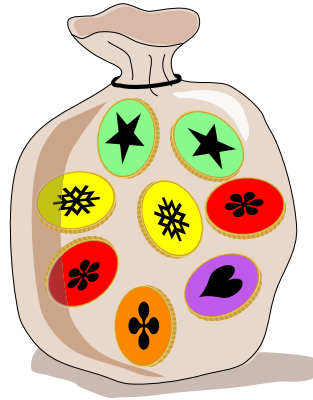
Welcher ist Emils Sack?





















## Lösung

Die richtige Antwort ist C:



In Emils Sack sind

- 4 Münzen von der Art  ,
- 2 von der Art  ,
- 1 von der Art   und
- 1 von der Art  .

	Emils Sack	Sack A	Sack B	Sack C	Sack D
 	4	3	4	4	2
 	2	3	1	2	4
 	1	1	2	1	1
 	1	1	1	1	1

Nur Sack C hat für jede Münzenart die gleiche Anzahl Münzen wie Emils Sack. Daher ist C die einzige richtige Lösung.

## Dies ist Informatik!

In dieser Aufgabe muss man die Münzarten erkennen, ohne dass man beide Seiten der Münzen sieht. Man hat nur eine unvollständige Information. Objekte der realen Welt werden in einem Computersystem mit ihren wesentlichen Merkmalen gespeichert. Oft genügt es, nur einen Teil dieser Merkmale zu kennen, um ein Objekt erkennen zu können. Eine Kamera in einem autonomen Fahrzeug sieht stets nur Teile der Realität und das Computersystem muss trotzdem in der Lage sein, Fahrzeuge



und Verkehrsteilnehmer zu erkennen und auf die jeweilige Verkehrssituation richtig zu reagieren. Die künstliche Intelligenz in Computersystemen lernt allmählich und immer besser aus Fragmenten ganze Objekte zu erkennen, ähnlich wie Menschen.

## Stichwörter und Webseiten

- Multimenge: <https://de.wikipedia.org/wiki/Multimenge>
- Unstrukturierte Daten: [https://de.wikipedia.org/wiki/Unstrukturierte\\_Daten](https://de.wikipedia.org/wiki/Unstrukturierte_Daten)

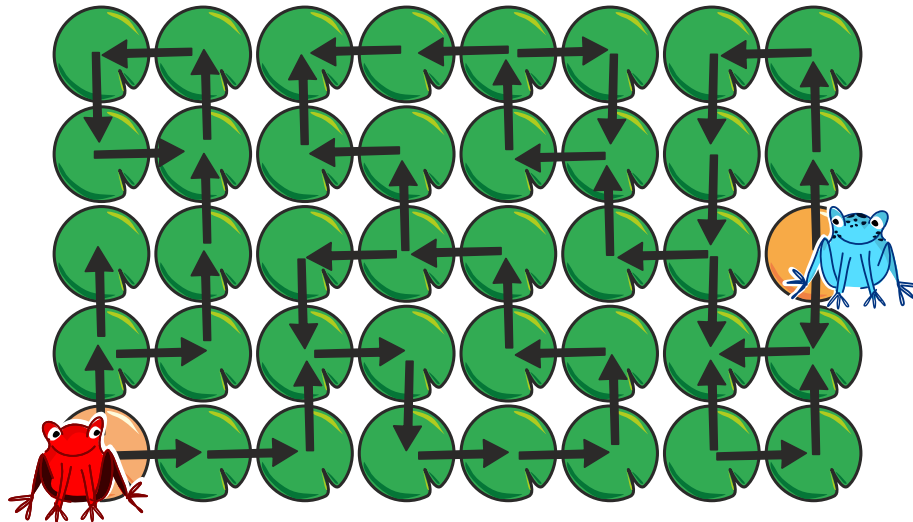




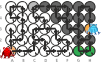
# 11. Treffen sie sich?

Auf einem See können zwei Frösche von Seerosenblatt zu Seerosenblatt springen – aber nur entlang der Pfeile.

*Auf welchem Seerosenblatt können sie sich treffen?*

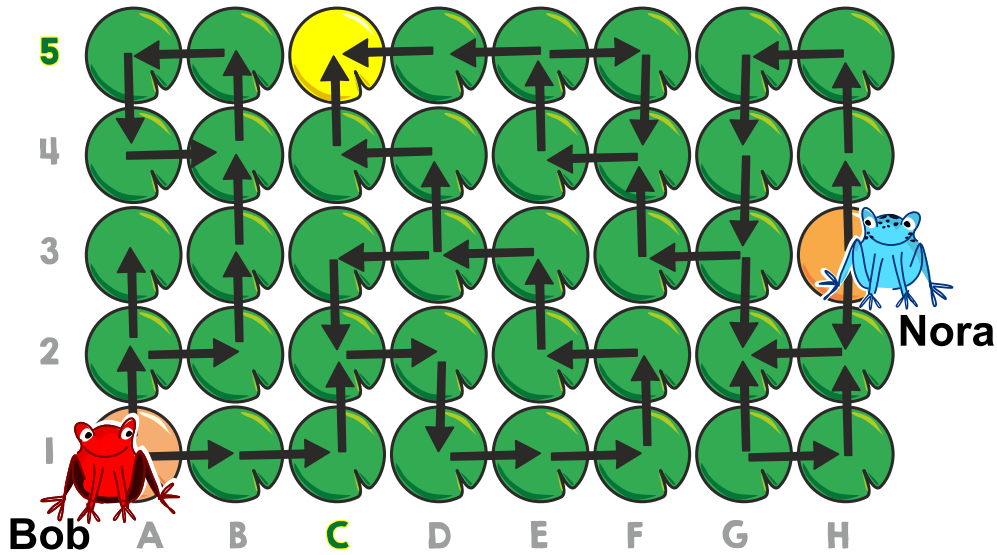


Man kann auf die Blätter klicken. Klickt man auf ein Blatt, wird dieses ausgewählt und gleichzeitig ein bereits ausgewähltes Blatt wieder deaktiviert.



## Lösung

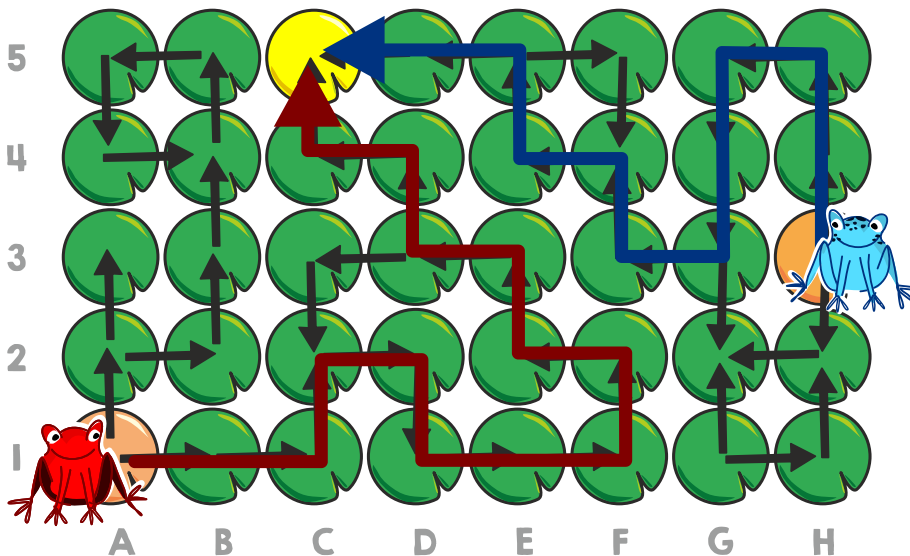
Die Frösche können sich nur auf dem Blatt C5 treffen.



Auf seiner Startposition hat der rote Frosch Bob zwei Möglichkeiten: Geht er nach «oben», gerät er entweder in die Sackgasse A3 oder bleibt im Kreis stecken, der bei B4 beginnt. Geht er anfangs «rechts» (nach B1), kann er zunächst bis D3 weiterspringen. Dort kann er «links» in einen Kreis springen, der ihn wieder zu D3 bringt, oder nach «oben», was ihn weiter zu C5 bringt – eine weitere Sackgasse.

Der blaue Frosch Nora hat am Start auch zwei Wahlmöglichkeiten. Geht sie nach «unten», gerät sie in die Sackgasse G2. Falls sie nach «oben» startet, erreicht sie zunächst G3. Von dort kann sie entweder wieder in die Sackgasse G2 geraten oder nach «links» gehen und schliesslich E5 erreichen. Von dort geht es wieder in einen Kreis, der sie zu E5 bringt, oder zur Sackgasse in C5.

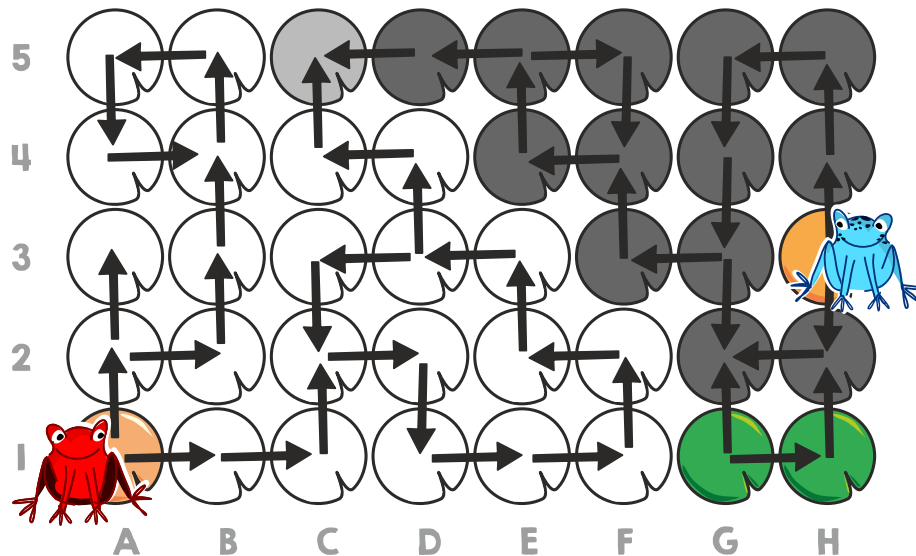
Wir wissen bereits, dass auch Bob C5 erreichen kann, also können sie sich dort treffen. Die Zeichnung zeigt die Wege, wie sie beide zu C5 gelangen können.







Das garantiert allerdings noch nicht, dass sie sich nicht auch woanders treffen könnten. Die nächste Zeichnung zeigt alle Blätter, die Bob (weiss) and Nora (dunkelgrau) erreichen können, wenn sie den Pfeilen auf jede erdenkliche Art folgen. Wir sehen, dass nur C5 von beiden erreicht werden kann.



## Dies ist Informatik!

Wie kann das letzte Bild erstellt werden? Die Blätter, die von einem Frosch erreicht werden können, können mit einer *Breiten- oder Tiefensuche* gefunden werden. Dies sind zwei der wichtigsten Standardverfahren in der Informatik. Mit ihrer Hilfe kann man die dunkelgrauen und die weissen Blätter bestimmen. Schliesslich müssen nur noch die Blätter gefunden werden, welche von beiden Fröschen erreicht werden können.

## Stichwörter und Webseiten

- Breitensuche: <https://de.wikipedia.org/wiki/Breitensuche>
- Tiefensuche: <https://de.wikipedia.org/wiki/Tiefensuche>





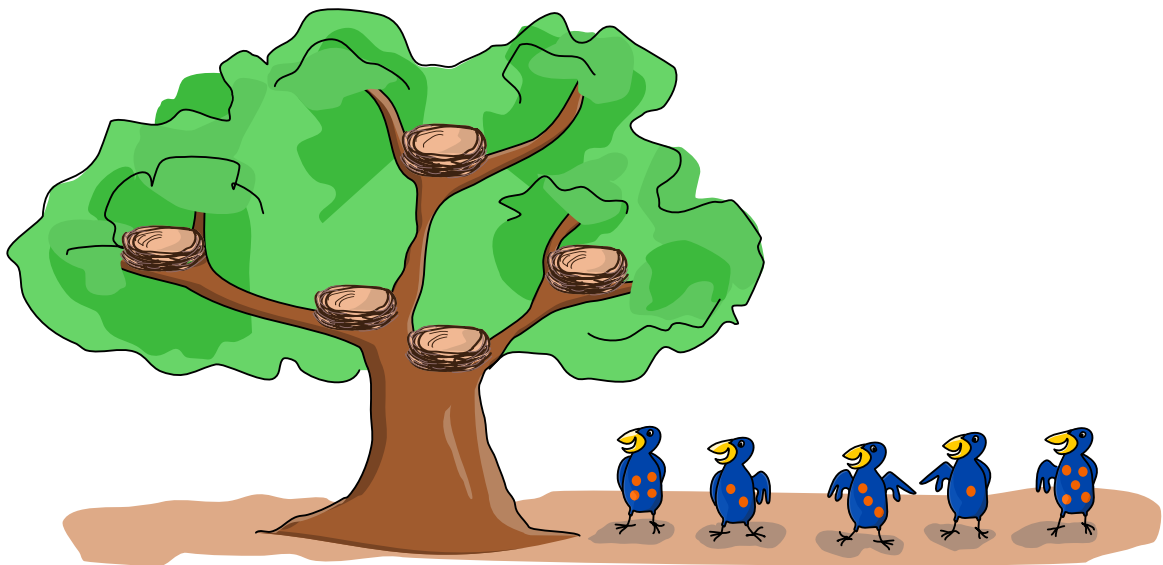
## 12. Dottis

Dottis sind Vögel mit Punkten. Neben einem Baum stehen fünf Dottis. Einer nach dem anderen - in der Reihenfolge von links nach rechts - klettern sie in den Baum und ziehen in die leeren Nester. Der mit den vier Punkten ist der erste. Jeder Dotti geht so vor:

Er beginnt unten am Baum. Er führt solange die folgenden Schritte aus, bis er ein leeres Nest gefunden hat:

1. Er klettert hoch, bis er ein Nest findet.
2. Wenn das Nest leer ist, dann zieht er in dieses Nest und bleibt dort.
3. Sonst klettert er weiter, und zwar, wenn der im Nest sitzende Dotti ...
  - ... mehr Punkte als er selbst hat, dann nach links.
  - ... gleich viele oder weniger Punkte hat, dann nach rechts.

*Wo sind die Dottis ganz am Ende? Setze jeden Dotti in das richtige Nest.*

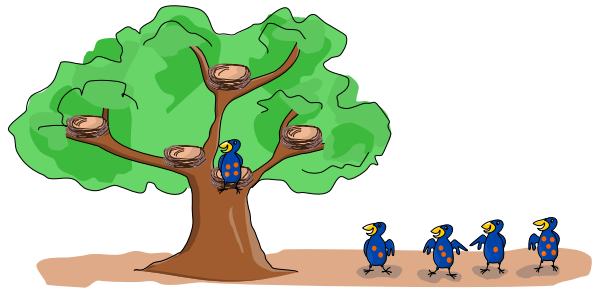




## Lösung

So kommt man auf die richtige Lösung:

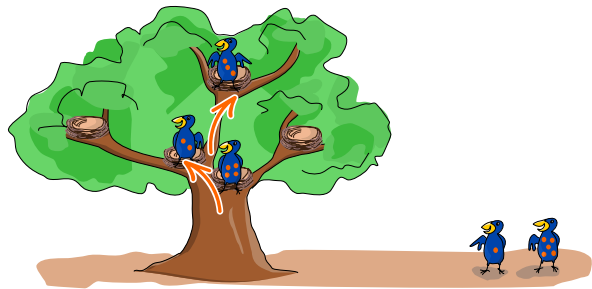
Der erste Dotti, der mit 4 Punkten, zieht in das unterste Nest und bleibt dort.



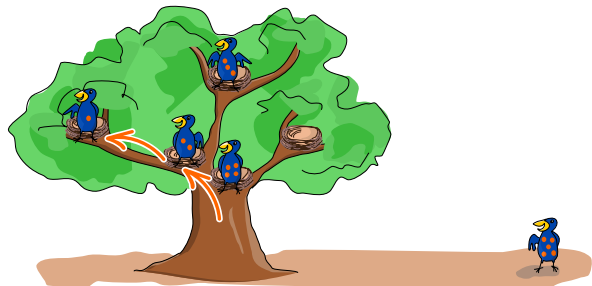
Der zweite Dotti hat 2 Punkte. Im untersten Nest sitzt der erste Dotti mit 4 Punkten. Weil 4 mehr als 2 ist, klettert der zweite Dotti nach links weiter und zieht in das erste freie Nest.



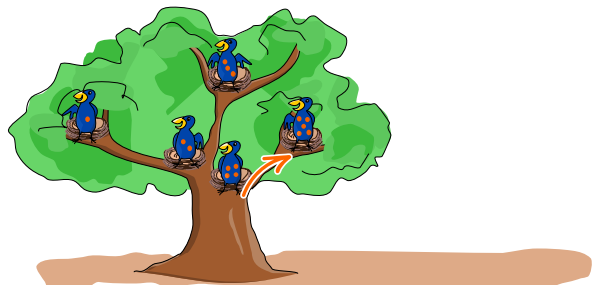
Der dritte Dotti hat 3 Punkte. Dieser klettert beim untersten Nest, wo der Dotti mit 4 Punkten sitzt, nach links, weil 4 mehr als 3 ist. Im nächsten Nest sitzt der Dotti mit 2 Punkten. Weil 3 mehr als 2 ist, klettert der dritte Dotti nach rechts weiter. Er zieht dann in das nächste freie Nest. Das ist das höchste Nest.



Der vierte Dotti hat 1 Punkt. Weil alle anderen Dottis mehr Punkte haben, klettert er bei jedem belegten Nest nach links. Er kommt dann beim Nest ganz links an und zieht dort ein.



Der letzte Dotti hat 5 Punkte. Weil kein Dotti mehr Punkte hat, klettert er bei jedem belegten Nest nach rechts. Das tut er einmal beim untersten Nest und zieht somit in das leere Nest ganz rechts.





## Dies ist Informatik!

Wenn sich die Dottis nach diesem Verfahren in die Nester setzen, hat das einen interessanten Vorteil: Ein bestimmter Dotti kann dann schnell gefunden werden. Wenn der Dotti, den du suchst, weniger Punkte hat als der, auf den du gerade schaust, musst du im linken Teil des Baums weitersuchen. Ansonsten suchst du rechts weiter. Mit jeder Prüfung eines Vogels kannst du also den Suchbereich auf eine von zwei Hälften einschränken. Deshalb wirst du deinen Dotti schnell finden.

Es gibt viele Arten, auf die man Daten organisieren kann; man spricht von verschiedenen *Datenstrukturen*. Die Datenstruktur in dieser Biberaufgabe ist ein *binärer Suchbaum*. Das Wort «binär» kommt vom lateinischen Wort «bis» für «zweimal». Denn am Ende eines Astes (dort, wo in der Aufgabe ein Nest sitzt) führen höchstens zwei kleinere Äste weiter. Binäre Suchbäume werden in Computerprogrammen verwendet, wenn viele Daten schnell gefunden werden müssen. Sie sind meistens viel grösser als der kleine Baum in der Aufgabe. Ausserdem gibt es noch einen Unterschied: Der Baum in der Aufgabe hatte eine feste Anzahl von fünf Dottis. Dagegen kann man bei einem binären Suchbaum üblicherweise immer weiter Daten einfügen. Zum Einfügen wird einfach an das Ende eines Astes ein neuer Ast angehängt und so der Baum vergrössert. Datenstrukturen, die sich so verändern können, nennt man *dynamische Datenstrukturen*.

## Stichwörter und Webseiten

- Binärer Suchbaum: [https://de.wikipedia.org/wiki/Binärer\\_Suchbaum](https://de.wikipedia.org/wiki/Binärer_Suchbaum)

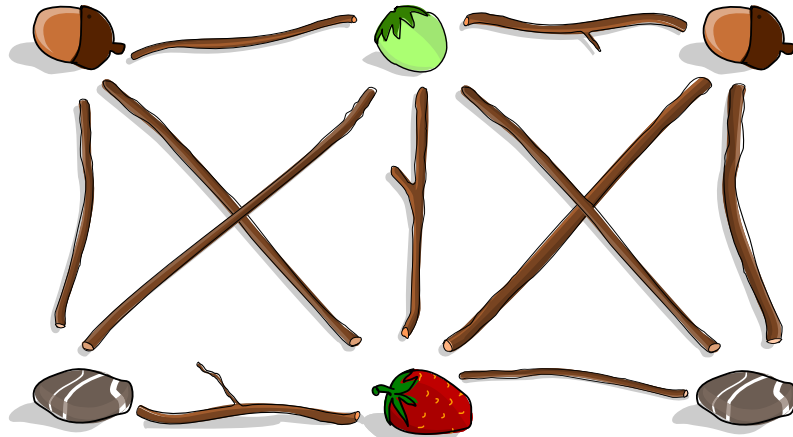




## 13. Erdbeerklau

Anja will im Garten ein Kunstwerk schaffen und hat dafür verschiedene Sachen gesammelt: Mehrere Eicheln, Haselnüsse, Steine und eine Erdbeere. Sie legt einige der Sachen auf den Rasen.

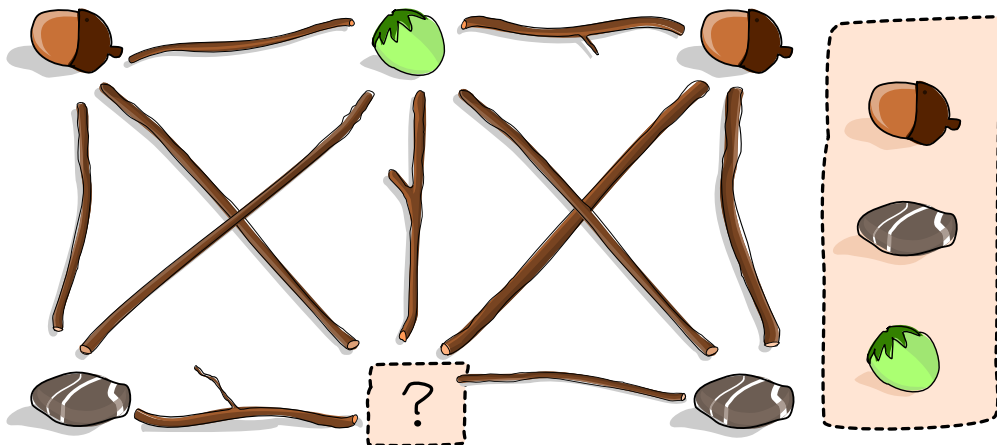
Danach legt Anja Äste zwischen diese Sachen. Dabei befolgt sie folgende Regel: Ein Ast darf nicht zwischen zwei gleichen Sachen liegen – zum Beispiel nicht zwischen zwei Eicheln. Hier ist das fertige Kunstwerk:



Während Anja weg ist, kommt ihr Bruder und isst die Erdbeere.

*Kannst du ihm helfen, die Tat zu verschleiern?*

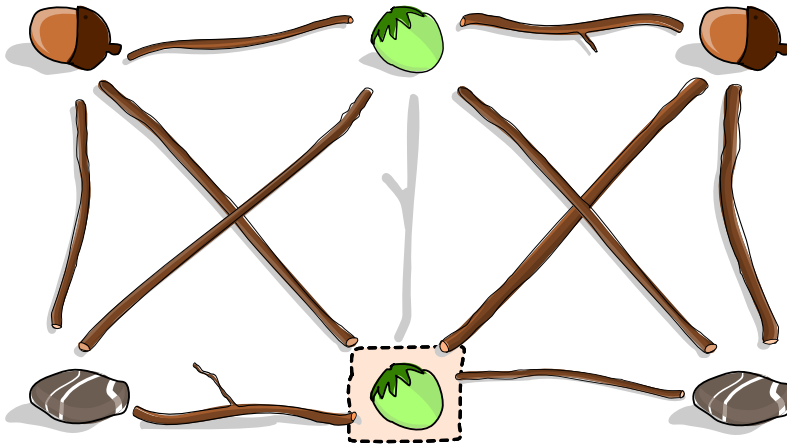
*Platziere eine andere Sache an die Stelle der Erdbeere und entferne genau einen Ast. Am Ende soll Anjas Regel auch für das veränderte Kunstwerk gelten.*





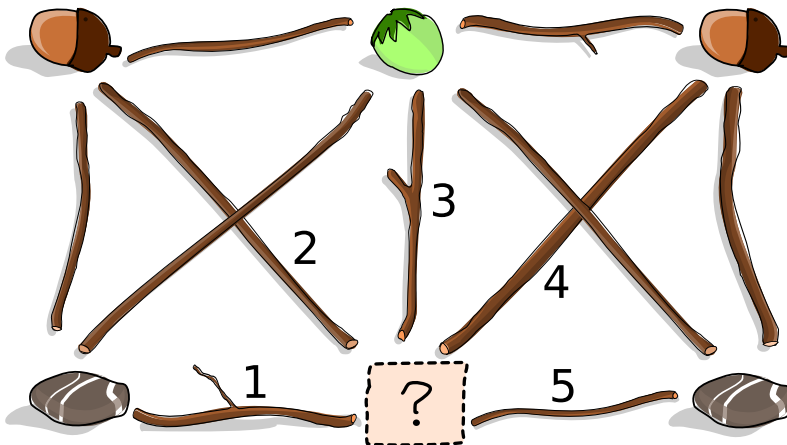
## Lösung

Wenn man die Erdbeere durch eine Haselnuss ersetzt, verletzt der Ast 3 in der Mitte Anjas Regel: Er liegt zwischen zwei gleichen Sachen, nämlich zwei Haselnüssen. Deshalb muss dieser Ast entfernt werden.



Bei den beiden anderen möglichen Ersetzungen muss man mehr als einen Ast entfernen:

- Wird die Erdbeere durch eine Eichel ersetzt, muss man die Äste 2 und 4 entfernen.
- Wird die Erdbeere durch einen Stein ersetzt, muss man die Äste 1 und 5 entfernen.



## Dies ist Informatik!

Anjas Kunstwerk kann als *Graph* dargestellt werden. Ein Graph besteht aus *Knoten* (den Plätzen für die Sachen) sowie aus *Kanten* (den Ästen), die jeweils zwei Knoten miteinander verbinden. Graphen sind sehr vielseitig und werden bei vielen Informatik-Aufgabenstellungen zur Modellierung verwendet. Wenn zwei Knoten direkt durch eine Kante verbunden sind, sind sie *Nachbarn* voneinander. Eine Gruppe von Knoten, in der jeder Knoten Nachbar von jedem anderen ist, heisst *Clique*. Im unserem Graphen haben wir zwei Cliques mit vier Knoten: die rechte und die linke Hälfte des Graphen. (Die Haselnuss oben und das Fragezeichen gehören zu beiden Cliques.) Aus Anjas Regel folgt, dass alle Knoten einer Clique mit unterschiedliche Sachen belegt sein müssen. Um die Regel einzuhalten,





brauchen wir also mindestens so viele unterschiedliche Sachen wie eine Clique Knoten hat. Nachdem die Erdbeere entfernt wurde, haben wir aber nur noch 3 verschiedene Sachen. Also dürfen jetzt noch Cliques mit höchstens 3 Knoten übrig bleiben, wenn die Regel weiterhin erfüllt bleiben soll. Es muss also eine Kante (ein Ast) entfernt werden, sodass beide Cliques mit vier Knoten kaputt gehen.

Anjas Regel entspricht einer Regel im sogenannten *Färbungsproblem* für Graphen: Wir ordnen jedem Knoten eines Graphen eine Farbe zu, wobei Nachbarn unterschiedliche Farben haben müssen. (Die Farben entsprechen den verschiedenen Arten von Sachen.) Das Ziel ist meistens, möglichst wenig Farben zu benutzen. Das Problem, wie man einen Graphen mit der minimalen Anzahl von Farben einfärbt, hat viele Anwendungen. Einige Beispiele sind die Planung von Sportwettkämpfen, das Entwerfen eines Sitzplans und sogar das Lösen eines Sudoku-Rätsels.

## Stichwörter und Webseiten

- Färbepblem: [https://de.wikipedia.org/wiki/Färbung\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Färbung_(Graphentheorie))
- minimale Kantenfärbung, Kantenfärbung: <https://de.wikipedia.org/wiki/Kantenfärbung>
- Clique: [https://de.wikipedia.org/wiki/Clique\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Clique_(Graphentheorie))

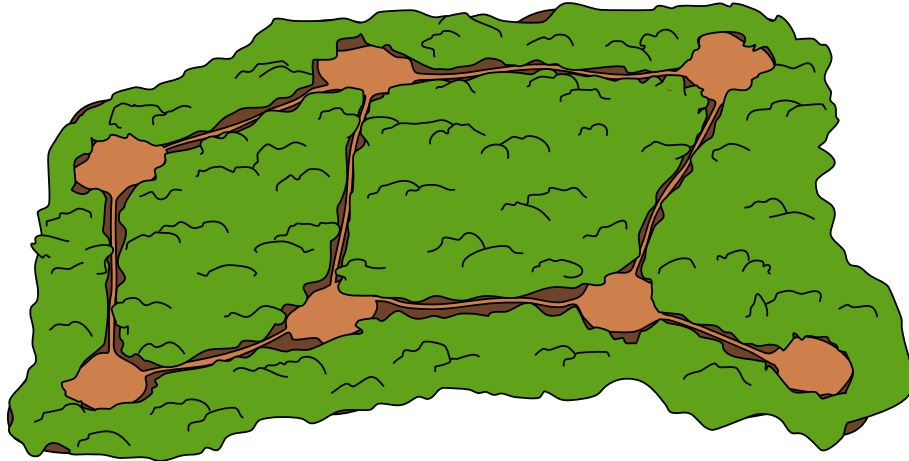




## 14. Tiere beobachten

Die Förster wollen die Tiere auf den Wegen im Wald beobachten. Von jeder Lichtung aus können sie alle abgehenden Wege bis zur nächsten Lichtung beobachten. Es sollen möglichst wenig Förster alle Wege beobachten.

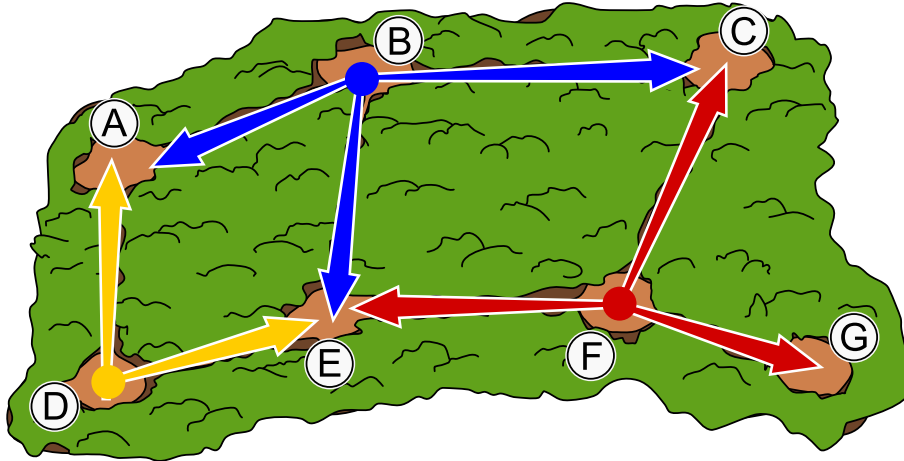
*Wähle möglichst wenige Lichtungen, von denen die Förster alle Wege beobachten können!*





## Lösung

Das Bild zeigt die minimale Lösung, bei der Förster auf nur drei Lichtungen stehen und alle Wege beobachten können.



Es gibt acht Wege, die beobachtet werden müssen. Wenn nur zwei Förster alle Wege beobachten könnten, müsste es eine Lichtung geben, von der mindestens vier Wege fortführen. Aber eine solche Lichtung gibt es nicht in diesem Wald. Daher sind zwei Förster zu wenig.

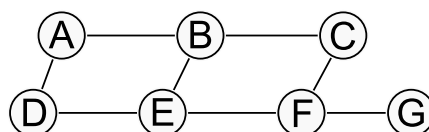
Es sind also mindestens drei Förster zur Beobachtung aller Wege notwendig. Daher ist die hier angegebene Lösung eine Lösung mit der kleinstmöglichen Anzahl an Förstern. Tatsächlich gibt es keine weitere Lösung mit genau drei Förstern.

Aus der Anzahl der zu überblickenden Wege und der Tatsache, dass es keine Lichtungen mit mehr als drei wegführenden Wegen gibt, können wir schliessen: Jeder Förster muss mindestens zwei Wege überblicken, die die anderen Förster nicht überblicken.

Um die Sackgasse zwischen Lichtung F und G zu überblicken, muss ein Förster auf der Lichtung F platziert werden. Um den Weg zwischen Lichtung B und C zu überblicken, muss der zweite Förster von der Lichtung B aus beobachten. Um die letzten zwei Wege mit nur einem Förster zu überblicken, muss dieser auf der Lichtung D platziert werden. Somit kommt man zwingend auf die angegebene Lösung und es kann keine weiteren geben.

## Dies ist Informatik!

Beziehungen zwischen Dingen (z.B. Wege zwischen Lichtungen) können als sogenannter *Graph* dargestellt werden. Ein Graph besteht aus *Knoten* (hier: die Lichtungen), dargestellt als Kreise, und *Kanten* (hier: die Wege), dargestellt als Linien zwischen den Knoten. Der Graph dieser Aufgabe schaut so aus:





Bei dieser Biberaufgabe soll man in dem Graphen eine kleinste Anzahl an Knoten finden, sodass jede Kante bei mindestens einem dieser Knoten beginnt oder endet. So eine Teilmenge der Knoten nennen Informatiker eine *minimale Knotenüberdeckung* (engl. *minimal vertex cover*). Solche Knotenüberdeckungsprobleme finden wir im Alltag zum Beispiel beim Suchen der besten Stellen für Strassenlampen oder für das geschickte Platzieren von Überwachungskameras.

## Stichwörter und Webseiten

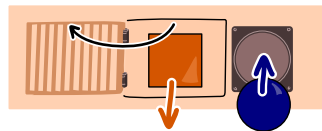
- Graph: [https://de.wikipedia.org/wiki/Graph\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Graph_(Graphentheorie))
- Knotenüberdeckung: <https://de.wikipedia.org/wiki/Knotenüberdeckung/>





# 15. Geburtstagsrätsel

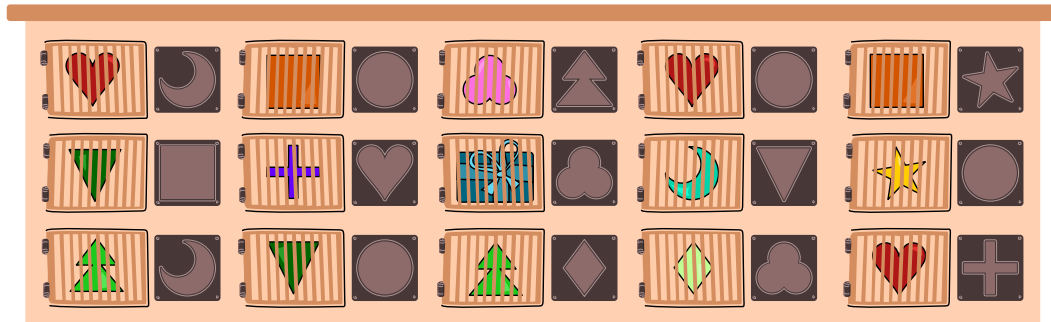
Bastian bekommt zum Geburtstag eine Kiste mit 15 Türen. Hinter der mittleren Tür ist ein weiteres Geschenk. Hinter den anderen Türen sind Bausteine. Zu jeder Tür gehört ein Loch, rechts neben der Tür. Bastian kann eine Tür öffnen, indem er in das Loch einen Baustein gleicher Form einwirft – wie einen Schlüssel.



Zu Beginn hat Bastian diesen runden Baustein: 

Er will höchstens fünf Türen öffnen, um das Geschenk zu erreichen.

*Welche Tür muss Bastian dafür zuerst öffnen?*



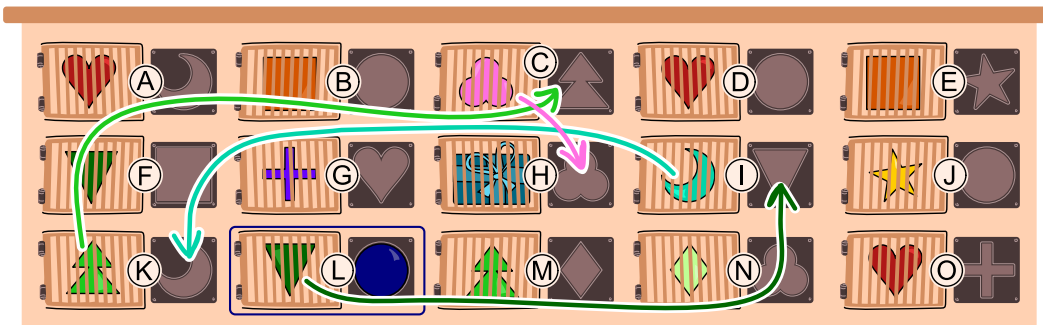


## Lösung

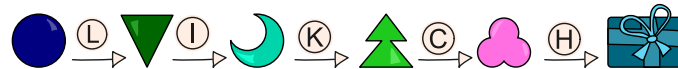
Bastian muss zuerst die blau markierte Tür öffnen:



Im folgenden Bild sind die Türen mit Buchstaben versehen und die Pfeile zeigen, wie Bastian mit insgesamt 5 Türöffnungen das Geschenk erreicht.



Wir können die Reihenfolge, in der er die fünf Türen öffnet, auch wie folgt darstellen.



Es gäbe auch andere Wege zum Geschenk, zum Beispiel folgenden.



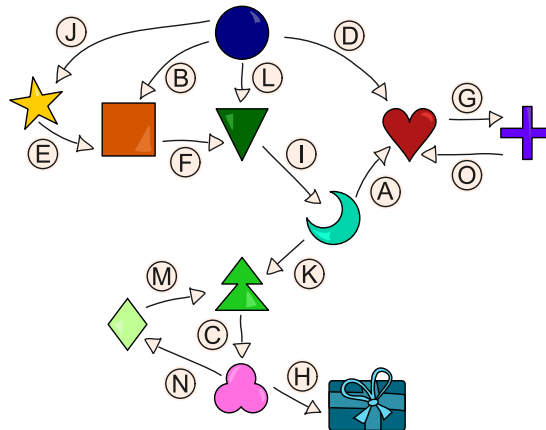
Doch diese Wege sind alle zu lang, es müssten mehr als fünf Türen geöffnet werden. Alle Möglichkeiten durchzuprobieren, ist ziemlich aufwendig.

Im vorliegenden Fall findet man den kürzesten Weg und damit die richtige Lösung am schnellsten mit einer sogenannten *Rückwärtssuche*: Man beginnt bei der Tür mit dem Geschenk und schaut dann jeweils, welchen Baustein man benötigt.

## Dies ist Informatik!

Mit mehr Zeit und Aufwand können wir die Situation in der Aufgabe auch als *Graph* darstellen:







Ein Graph besteht allgemein aus *Knoten* (Kreisen) und *Kanten* (Linien) zwischen den Knoten. In unserem Fall haben wir einen Knoten für jede Form und das Geschenk. Die Kanten sind hier Pfeile (auch *gerichtete* Kanten genannt) und entsprechen den Türen. Jeder Pfeil führt von der Form zum Öffnen der Tür zu der Form hinter der Tür.

Die Informatik arbeitet sehr gerne mit Graphen. Einerseits bieten sie oft anschauliche Darstellungen von abstrakten Zusammenhängen.

Andererseits existieren fertige Algorithmen, die uns Fragen zu Graphen sehr effizient beantworten. Bei komplizierteren Aufgaben kann sich der Aufwand für das Aufstellen des Graphen deshalb schnell lohnen.

In der vorliegenden Aufgabe suchen wir einen Weg der Länge höchstens 5 vom erhaltenen Baustein  zum Geschenk . Ein guter Algorithmus dafür ist die sogenannte *Breitensuche*. Dieser funktioniert sowohl für Graphen mit gerichteten Kanten, wie in der Aufgabe, also auch für Graphen mit ungerichteten Kanten.

## Stichwörter und Webseiten

- Gerichteter Graph: [https://de.wikipedia.org/wiki/Gerichteter\\_Graph](https://de.wikipedia.org/wiki/Gerichteter_Graph)
- Breitensuche: <https://de.wikipedia.org/wiki/Breitensuche>

























# 16. Lieblingsgeschenk

Die Biberfamilie hat fünf Geschenke für ihre fünf Kinder. Jedes Kind nennt zuerst sein Lieblingsgeschenk und dann das zweitliebste. Die Geschenke sollen richtig zugeteilt werden:

1. Möglichst viele Kinder sollen ihr Lieblingsgeschenk bekommen.
2. Die übrigen sollen das zweitliebste bekommen.

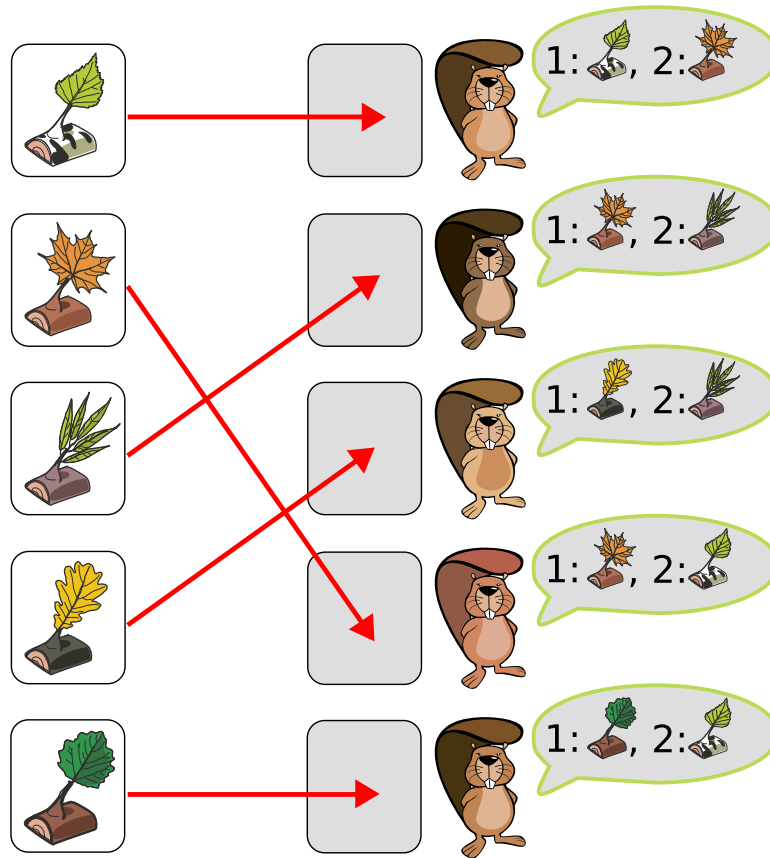
*Gib den Kindern die richtigen Geschenke.*

	<input type="checkbox"/>		1:  , 2: 
	<input type="checkbox"/>		1:  , 2: 
	<input type="checkbox"/>		1:  , 2: 
	<input type="checkbox"/>		1:  , 2: 
	<input type="checkbox"/>		1:  , 2: 



## Lösung

Hier ist die einzige Zuordnung der Geschenke, die beide Bedingungen erfüllt.



Die obige Grafik ordnet vier Bibern ihr Lieblingsgeschenk zu und einem Biber sein zweitliebstes Geschenk. Nicht alle Biberkinder können ihr Lieblingsgeschenk bekommen, da zwei Biber dasselbe Lieblingsgeschenk haben. Daher ist keine Zuordnung möglich, bei der mehr Biber ihr Lieblingsgeschenk bekommen. Beachte: Wenn du die Zuteilung von oben nach unten vornimmst und das zweite Geschenk dem zweiten Biber zuteilst, dann bekommt der vierte Biber keines seiner bevorzugten Geschenke. In dieser Aufgabe genügt es also nicht, für jeden einzelnen Biber die jeweils beste aktuelle Auswahl zu treffen.

Eine Lösungsstrategie ist, dass du zuerst alle Geschenke zuordnest, die nur von einem Kind das Lieblingsgeschenk sind. Danach bleiben nur noch zwei Kinder mit demselben Lieblingsgeschenk übrig. Bei diesen schaust du, welchem das zweitliebste Geschenk zugeordnet werden kann. Dem anderen teilst du sein Lieblingsgeschenk zu.

## Dies ist Informatik!

Bei dieser Aufgabe handelt sich um ein eindeutiges *Zuordnungsproblem*: Wir möchten die Geschenke so zuordnen, dass alle Kinder ein Geschenk bekommen und es kein Kind ohne Geschenk gibt. Dabei haben die Kinder nicht nur einen einzelnen Wunsch, sondern geben eine Reihenfolge von Vorlieben



an. Solche Zuordnungsprobleme mit Reihenfolgen von Vorlieben können sehr kompliziert werden. Die Informatik hilft uns dabei, solche Probleme möglichst rasch zu lösen.

Eine Möglichkeit ist den Zuordnungen einen Wert zu geben: Das Lieblingsgeschenk hat Wert 1 und das zweitliebste Geschenk den Wert 2. Wir möchten den Gesamtwert minimieren. Eine *Zuordnung* (engl. *Matching*) ist *optimal*, wenn es keine andere Zuordnung mit mehr erfüllten Erstplatzierungen gibt. In der Informatik wird so eine Zuordnung als *Rang-Maximal-Matching* bezeichnet. Es gibt viele Matching-Probleme. Eines davon wird als *Problem der stabilen Paarung* (engl. *Stable Marriage Problem*) bezeichnet. Klingt interessant? Dann solltest du Informatik studieren!

## Stichwörter und Webseiten

- Zuordnungsproblem: <https://de.wikipedia.org/wiki/Zuordnungsproblem>
- Rang-Wahl: <https://de.wikipedia.org/wiki/Rang-Wahl>



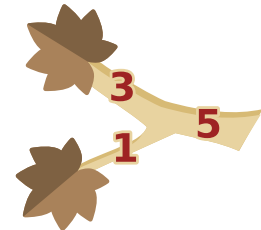


## 17. Rette den Baum!

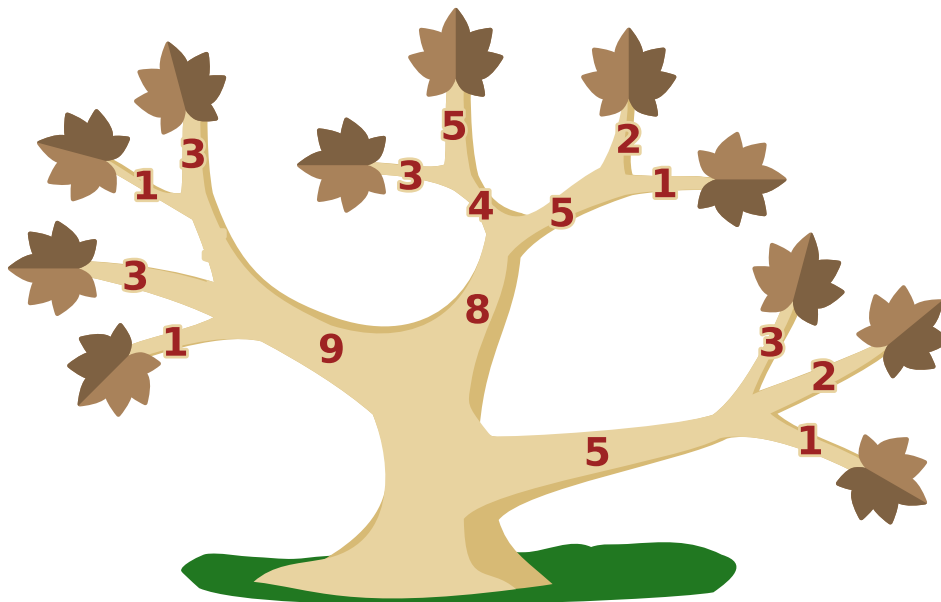
Ein Baum in Brunos Garten ist krank, alle Blätter sind vertrocknet. Bruno will den Baum retten. Dazu muss er einige Äste absägen, so dass am Ende alle Blätter entfernt sind. Dann können neue Äste mit neuen Blättern wachsen. Bruno möchte so schnell wie möglich fertig sein.

Das Bild zeigt ein Beispiel:

Um die beiden Blätter zu entfernen, kann Bruno entweder die beiden Äste mit den Blättern absägen oder nur den einen Ast, von dem die beiden anderen abzweigen. Die Zahlen geben für jeden Ast an, wie lange das Absägen dauert. Bruno wird also die beiden Äste mit den Blättern absägen, da  $3 + 1 < 5$ . Unten siehst du den gesamten Baum.



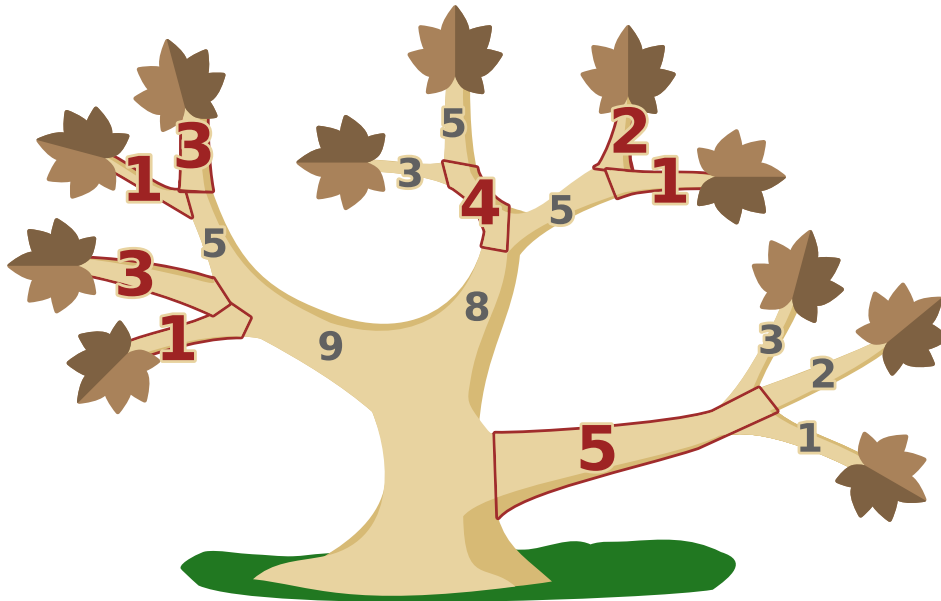
Welche Äste wird Bruno absägen, um so schnell wie möglich fertig zu sein?





## Lösung

So ist es richtig: Bruno sägt die rot markierten Äste ab, um so schnell wie möglich fertig zu sein:



Aber warum ist das so? Zunächst können wir ausrechnen, wie viel Zeit Bruno benötigt, wenn er nur die Äste mit Blättern absägt – damit wäre er ja fertig:

$$1 + 3 + 1 + 3 + 3 + 5 + 2 + 1 + 3 + 2 + 1 = 25$$

Nun gehen wir in Richtung Stamm weiter und überlegen immer wieder neu, ob es schneller sein könnte, den Ast abzusägen, von dem die bisherigen Äste direkt oder indirekt abzweigen. Nach dem ersten solchen Schritt ergibt sich die folgende Rechnung (die Funktion «min» berechnet das Minimum ihrer Argumente):

$$\begin{aligned} & 1 + 3 + \min(5, 1 + 3) + \min(4, 3 + 5) + \min(5, 2 + 1) + \min(5, 3 + 2 + 1) \\ &= 1 + 3 + (1 + 3) + 4 + (2 + 1) + 5 \\ &= 20 \end{aligned}$$

Dabei rechnen wir die Gesamtzeit zuerst noch nicht aus, damit wir besser sehen, welche Äste abzusägen sind. Nach dem nächsten Schritt sind wir schon am Stamm angekommen:

$$\begin{aligned} & \min(9, 1 + 3 + 1 + 3) + \min(8, 4 + 2 + 1) + 5 \\ &= (1 + 3 + 1 + 3) + (4 + 2 + 1) + 5 \\ &= 20 \end{aligned}$$

Schneller kann Bruno nicht fertig sein.





## Dies ist Informatik!

Stellen wir uns einmal vor, die abgesägten Stücke von Brunos Baum würden nicht direkt auf die Erde fallen – so wie beim Lösen dieser Biberaufgabe am Bildschirm. Dann könnte man sagen, dass der Baum durch das Absägen in nur zwei Teile zerlegt wird: Der eine Teil enthält alle abgesägten Stücke, insbesondere also alle Blätter, und der andere Teil enthält den Stamm und alle davon ausgehenden Äste bis zu den Sägestellen. Diese Zweiteilung bzw. dieser *Schnitt* durch den Baum ist minimal in Bezug auf die Zeit, die Bruno fürs Absägen aufbringen muss.

Auch die Informatik kennt Bäume und verwendet sie zur Darstellung von Objekten, die auf bestimmte Weise miteinander verbunden sind. Die Objekte werden als *Knoten* bezeichnet, die Verbindungen als *Kanten*. Dabei gibt es zwischen zwei Knoten immer nur einen Weg entlang der Kanten – so wie es im echten Baum von einem Blatt oder einer Astgabelung immer nur einen Weg entlang der Äste zum Stamm gibt. Verzichtet man auf diese Bedingung, ist allgemeiner von einem *Graphen* die Rede.

In einem allgemeinen Graphen ist ein *minimaler Schnitt*, also die Zerlegung in zwei oder auch mehrere Teile mit minimalen Kosten, nicht so einfach zu berechnen, wie wir es hier für einen Baum vorgemacht haben, aber auch nicht allzu schwierig. Das ist gut, denn es gibt interessante Anwendungen. Minimale Schnitte können etwa bei der Zerlegung von Bilddateien in ähnliche Teile verwendet werden. In speziellen Graphen, den *Flussnetzen*, mit denen unter anderem Datenflüsse durch Netzwerke modelliert werden können, entsprechen die Kosten eines minimalen Schnitts dem maximal möglichen Fluss durch das gesamte Netz.

## Stichwörter und Webseiten

- Baum: [https://de.wikipedia.org/wiki/Baum\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Baum_(Graphentheorie))
- Schnitt: [https://de.wikipedia.org/wiki/Schnitt\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Schnitt_(Graphentheorie))
- Flüsse und Schnitte in Netzwerken:  
[https://de.wikipedia.org/wiki/Flüsse\\_und\\_Schnitte\\_in\\_Netzwerken](https://de.wikipedia.org/wiki/Flüsse_und_Schnitte_in_Netzwerken)
- Max-Flow-Min-Cut-Theorem:  
<https://de.wikipedia.org/wiki/Max-Flow-Min-Cut-Theorem>





## 18. Bibliothek

Susi ist mit Tim in der Biber-Bibliothek. Sie wollen ein Buch ausleihen: «Dämme bauen, aber gern!»

Tim geht zu Regal 1, greift in Reihe 3, Fach 6 und holt das Buch heraus. Susi ist beeindruckt. Tim erklärt Susi, wie man den Ort eines Buches bestimmt:

Man nimmt von jedem Wort im Titel den Anfangsbuchstaben und bestimmt seine Position im Alphabet. Nach und nach werden diese Positionen addiert, aber vor jedem Addieren wird der bisher erreichte Wert mit 3 multipliziert. Für das gewünschte Buch ergibt sich 136. Schon ist klar, wo das Buch steht.

a	b	c	d	e	f	g	h	i	j	k	l	m
1	2	3	4	5	6	7	8	9	10	11	12	13
n	o	p	q	r	s	t	u	v	w	x	y	z
14	15	16	17	18	19	20	21	22	23	24	25	26

Dämme bauen, aber gern!

$((4 \cdot 3 + 2) \cdot 3 + 1) \cdot 3 + 7$

Nun stellt Susi für ihre Lieblingsbücher die entsprechenden Rechnungen auf. In einem Fall hat sie aber einen Fehler gemacht.

In welchem?

A)	<p>Gutes gegen Biber-Fieber</p> $((7 \cdot 3 + 7) \cdot 3 + 2) \cdot 3 + 6$	B)	<p>Bäume fallen für Dummies</p> $((2 \cdot 3 + 6) + 6) \cdot 3 + 4$
C)	<p>Der Herr der Dämme</p> $((4 \cdot 3 + 8) \cdot 3 + 4) \cdot 3 + 4$	D)	<p>Bebretti: Der erste Fall</p> $((2 \cdot 3 + 4) \cdot 3 + 5) \cdot 3 + 6$



## Lösung

Susi hat fast alles richtig gemacht: Sie hat immer die richtigen Positionswerte addiert, und sie hat die Zwischenergebnisse immer mit 3 multipliziert – mit einer Ausnahme: In Antwort B hat sie Letzteres einmal vergessen.

$$\begin{array}{c} \text{Bäume fallen für Dummies} \\ ((2 \cdot 3 + 6) \cdot 3 + 6) \cdot 3 + 4 \end{array}$$

## Dies ist Informatik!

Mit den «Orts-Bestimmungs-Ausdrücken» ermöglicht die Bibliothek ihren Besuchern, die Standorte der Bücher genau zu bestimmen. So muss niemand lange suchen. Eines müssen die Bibliothek und Besucher aber beachten: Für verschiedene Bücher können die Ausdrücke und damit auch deren Ergebnisse gleich sein. Zum Beispiel stehen «Bäume fallen für Dummies» und «Biber finden Fichten dufte» im gleichen Fach. Die Fächer dürfen also nicht zu klein sein, oder sie müssen flexibel angepasst werden können.

Auch bei Daten, die in Computerspeichern abgelegt werden, ist es eine gute Idee, wenn ihr Speicherort direkt aus den Daten selbst berechnet werden kann. Dafür wurden in der Informatik *Hash-Funktionen* entwickelt: mathematische Funktionen, die aus dem Inhalt der Daten bzw. eines Teils der Daten einen Wert berechnen, der direkt den Speicherort angibt – so wie bei den Buchtiteln in dieser Biberaufgabe. Gute Hash-Funktionen sorgen dafür, dass sich in möglichst wenigen Fällen der gleiche Wert ergibt. Kommt eine solche Kollision doch einmal vor, kennt die Informatik gute Methoden, damit umzugehen.

## Stichwörter und Webseiten

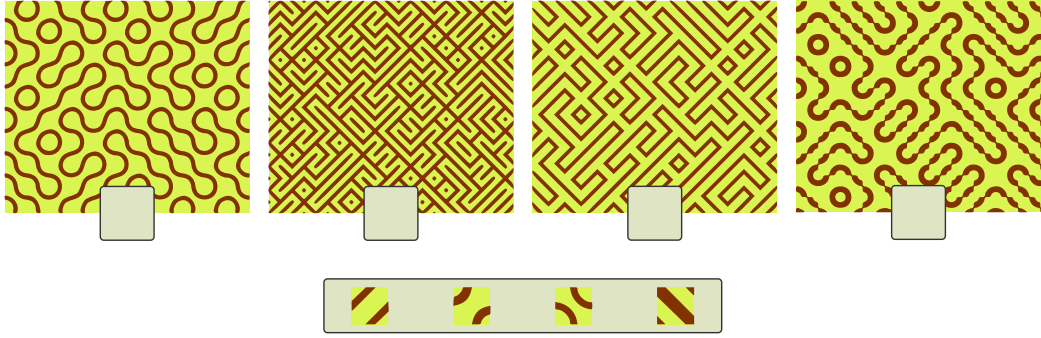
- Hashfunktion: <https://de.wikipedia.org/wiki/Hashfunktion>
- Hashtabelle: <https://de.wikipedia.org/wiki/Hashtabelle>
- [http://www.abenteuer-informatik.de/PDF/ai2020\\_oa leseversion.pdf](http://www.abenteuer-informatik.de/PDF/ai2020_oa leseversion.pdf), Kapitel 11: Ordnung im Chaos



## 19. Fliesenmuster

Die folgenden Muster wurde jeweils durch eine einzelne Fliese erzeugt. Die einzelnen Fliesen sind vergrößert dargestellt.

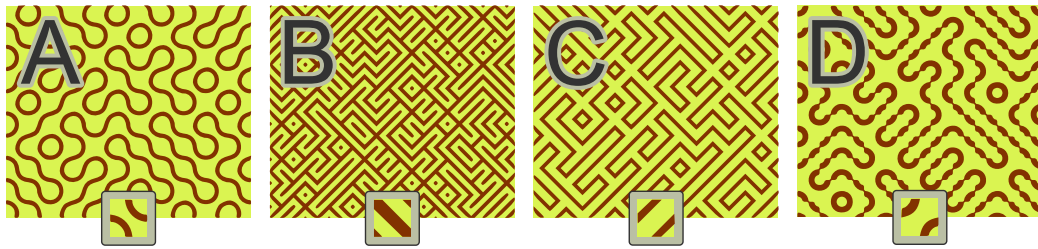
Ordne die Fliesen ihren möglichen Mustern zu.



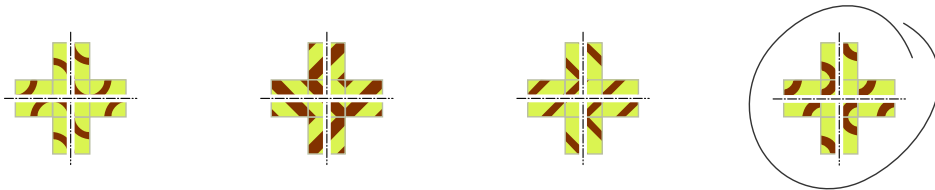


## Lösung

Das ist die richtige Zuordnung:



Legt man jeweils 5 Fliesen aneinander und vergleicht sie genauer, erkennt man deutliche Unterschiede:



Fliese hat als einzige der Fliesen vier Seiten, die nicht genau zueinander passen. Nur dadurch können Linien mit veränderlicher Breite wie in Muster D entstehen. Fliese ist die einzige, die quadratischen Punkte in Muster B erzeugen kann, nämlich mit jeweils vier zusammenstossenden Dreiecken. Zudem hat sie den grössten Flächenanteil von Braun gegenüber Gelb, genau wie B; auch daran kann man die Zusammengehörigkeit erkennen. Somit verbleibt für die runden Formen von Fliese nur Muster A als mögliches Ergebnis und für die geraden Formen von Fliese nur Muster C.

## Dies ist Informatik!

Diese Fliesen sind nach Sébastien Truchet (\* 1657; † 1729) benannt, der an verschiedenen Varianten dieser Fliesen gearbeitet hat. Kacheln mit 4 gleichen Seiten bilden eine Untermenge von Truchet-Kacheln (Truchet-Kacheln müssen aber nicht unbedingt 4 gleiche Seiten haben, wie in 3 von den Mustern oben). Dass mit ganz einfachen Bausteinen komplexe Muster erstellt werden können, ist eine interessante Eigenschaft, die uns in der Informatik immer wieder begegnet. Truchet-Kacheln werden in der Mathematik und der Informatik untersucht und in Computerspielen verwendet, um Labyrinth oder Dekorationen zu erstellen.

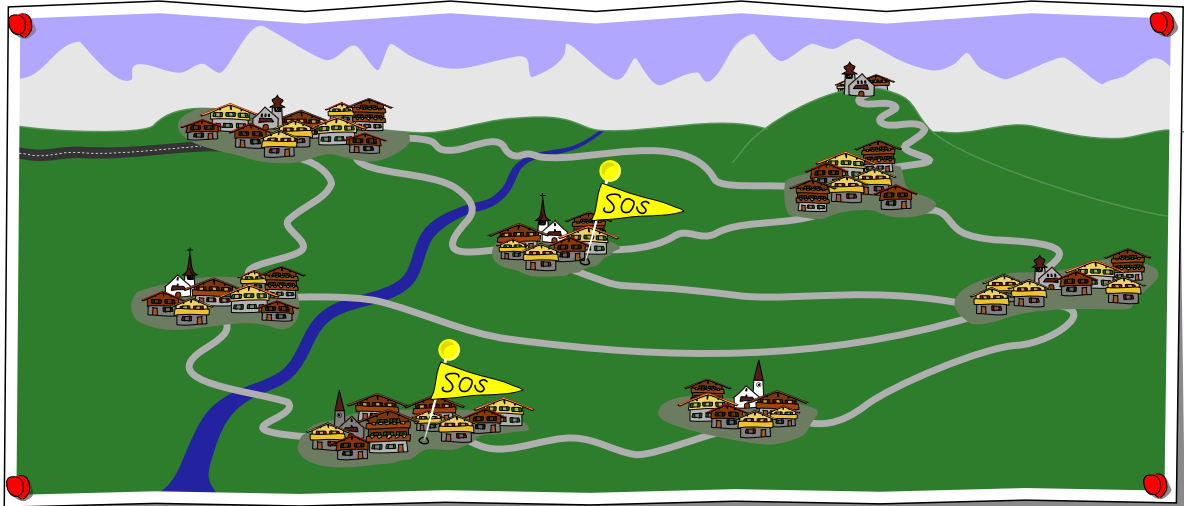
## Stichwörter und Webseiten

- Truchet-Kacheln: [https://en.wikipedia.org/wiki/Truchet\\_tiles](https://en.wikipedia.org/wiki/Truchet_tiles)






## 20. SOS aus den Bergen

Einige Bergdörfer werden aus der grossen Stadt über folgendes Strassennetz versorgt:



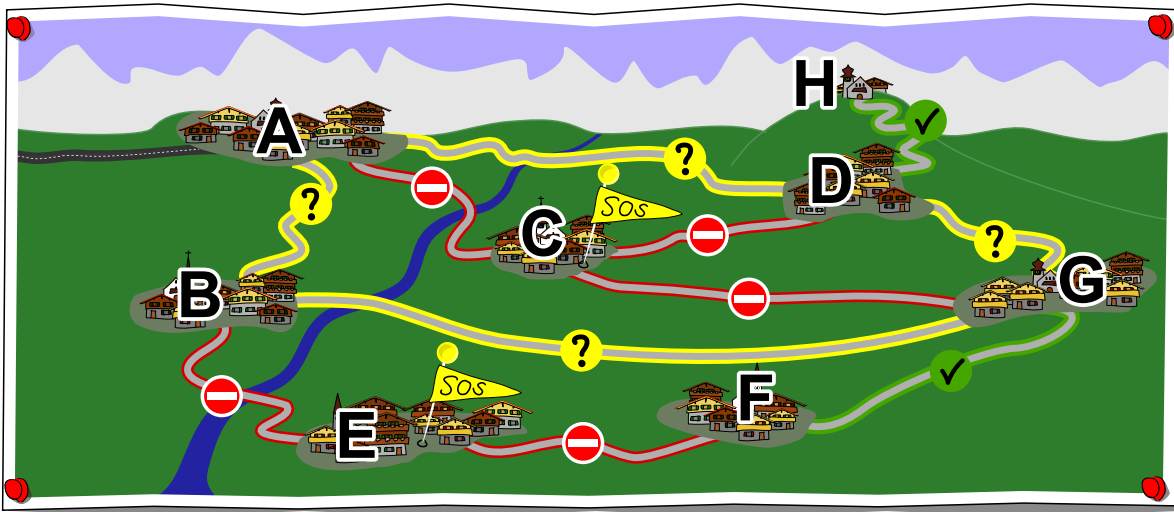
Nach einem Unwetter melden mehrere Dörfer, dass diese nicht mehr erreichbar sind, nämlich jene mit den SOS-Markierungen. Wir können daraus schliessen, dass einige Strassen blockiert sind.

Gib für jede Strasse zwischen den Dörfern in diesem Strassennetz an, ob diese (1) blockiert ist , (2) befahrbar ist , oder (3) ob wir nicht ohne weitere Informationen sagen können, ob die Straße befahrbar oder blockiert ist .



## Lösung

Die Karte zeigt, was wir über die Verbindungen im Strassennetz wissen:



Wir beginnen mit dem Aufspüren der blockierten Strassen. Die zwei Strassen, die zu Dorf E führen, können blockiert sein, da ansonsten Dorf E noch erreichbar wäre. Ebenso sind die drei Strassen zu Dorf C blockiert, da sonst Dorf C noch erreichbar wäre.

Als nächstes suchen wir die Strassen, die befahrbar sein müssen. Die Strasse zwischen Dorf G und F muss befahrbar sein, da ansonsten, aufgrund der blockierten Strasse zwischen Dorf F und E, das Dorf F nicht erreichbar wäre. Auch die Strasse zwischen der Kirche H und dem Dorf D muss befahrbar sein, da H erreichbar ist und nur über D erreicht werden kann.

Nun bleiben die nur möglicherweise befahrbaren Strassen übrig. Da die Dörfer B, G und D mehrfach mit dem Dorf A verbunden sind, können wir nicht sagen können, welche der verbleibenden Strassen befahrbar sind. So könnte das Dorf B beispielsweise über Dorf A, aber auch Dorf G erreicht werden. Dasselbe gilt auch für Dorf D. Das Dorf G kann entweder über Dorf B oder D versorgt werden. Irgendeine der Strassen im Kreislauf A – B – G – D – A könnte also blockiert sein und diese 4 Dörfer könnten trotzdem alle erreichbar bleiben.

## Dies ist Informatik!

So wie in Strassennetzen können auch bei Computernetzwerken Verbindungen fehlerhaft, überlastet oder ganz defekt sein. Um Ausfälle zu verhindern, werden oft Sicherheitsmassnahmen, wie z. B. mehrere Verbindungen zu einem Ort, eingeplant. Dies nennt man *Redundanz*.

Das Beheben von Fehlern in einem System ist eine Aufgabe, die Informatiker sehr oft erledigen müssen, nicht nur in Computernetzwerken, sondern auch in der Softwareentwicklung. Um einen Fehler zu beheben, muss man seine genaue Quelle identifizieren, und dieser Prozess erfolgt meist schrittweise in mehreren Schritten. Einige Programmierer glauben, dass man nie alle Fehler und Bugs in einem Programm finden kann.





## Stichwörter und Webseiten

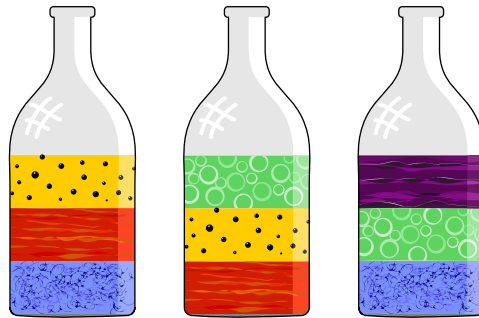
- Redundanz: [https://de.wikipedia.org/wiki/Redundanz\\_\(Technik\)](https://de.wikipedia.org/wiki/Redundanz_(Technik))
- Debuggen: <https://de.wikipedia.org/wiki/Debuggen>



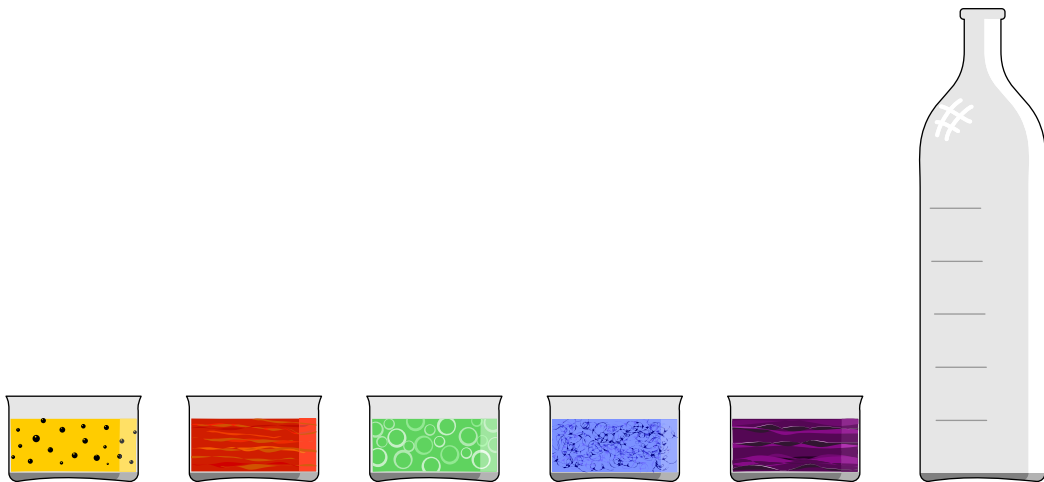


## 21. Schichte nach Dichte!

Mark hat Flaschen mit jeweils drei farbigen Flüssigkeiten, die übereinander geschichtet sind. Er weiss, dass sich die Flüssigkeiten mit geringerer Dichte immer über Flüssigkeiten mit grösserer Dichte bewegen. Nun möchte er sehen, wie es aussieht, wenn man alle farbigen Flüssigkeiten in eine Flasche gibt.



*Ordne die fünf farbigen Flüssigkeiten in der Flasche, so wie sie am Ende angeordnet sind!*





## Lösung

Das Bild zeigt die richtige Anordnung der fünf farbigen Flüssigkeiten in der grossen Flasche.



Du findest die Reihenfolge nach folgendem Verfahren: Schritt für Schritt entfernst du in Gedanken aus den drei gegebenen Flaschen die Flüssigkeiten, die nicht auf einer anderen Flüssigkeit liegen, und gibst sie in die grosse Flasche.

Zu Beginn hat nur in den beiden Flaschen 1 und 3 blaue Flüssigkeit und dort ist sie ganz unten, sie liegt also nirgendwo auf einer anderen Flüssigkeitsschicht. Die rote Flüssigkeit ist zwar in Flasche 2 ganz unten. Aber in Flasche 1 liegt sie auf der blauen Flüssigkeit und muss deshalb eine geringere Dichte als die blaue haben. Also wird als erstes die blaue Flüssigkeit aus den Flaschen entfernt und in die grosse Flasche gegeben.

Jetzt ist die rote Flüssigkeit die einzige, die nicht auf einer anderen Flüssigkeit liegt. Sie wird aus den Flaschen 1 und 2 entfernt und in die grosse Flasche gegeben. Danach kommt die gelbe, dann die grüne und zum Schluss die violette Flüssigkeit, die die geringste Dichte hat und über der keine andere Flüssigkeit liegt.

## Dies ist Informatik!

Bei der Lösung dieser Aufgabe hast du die Anordnung der Flüssigkeiten in den drei Flaschen der Aufgabenstellung ausgewertet und die Flüssigkeiten nach ihrer Dichte sortiert.

Ein Stoff hat viele messbare Eigenschaften: Siedetemperatur, Schmelztemperatur, elektrische Leitfähigkeit und eben die Dichte. In diesem Fall wurde die Dichte als Kriterium verwendet, Stoffe zu sortieren.

In vielen Computerprogrammen spielt das Sortieren von Daten eine wichtige Rolle. Das Verfahren, das bei dieser Aufgabe zur Ermittlung der Reihenfolge der Flüssigkeitsschichten verwendet wurde, nennt man *topologisches Sortieren*. Es wird zum Sortieren von Objekten angewendet, für die Beziehungen der Art Vorgänger/Nachfolger bekannt sind.



## Stichwörter und Webseiten

- Sortieren, Ordnung: <https://de.wikipedia.org/wiki/Sortierung>
- Topologisches Sortieren: [https://de.wikipedia.org/wiki/Topologische\\_Sortierung](https://de.wikipedia.org/wiki/Topologische_Sortierung),  
[https://www.ac.tuwien.ac.at/files/teaching/ss12/AD1/top\\_sortieren.pdf](https://www.ac.tuwien.ac.at/files/teaching/ss12/AD1/top_sortieren.pdf)



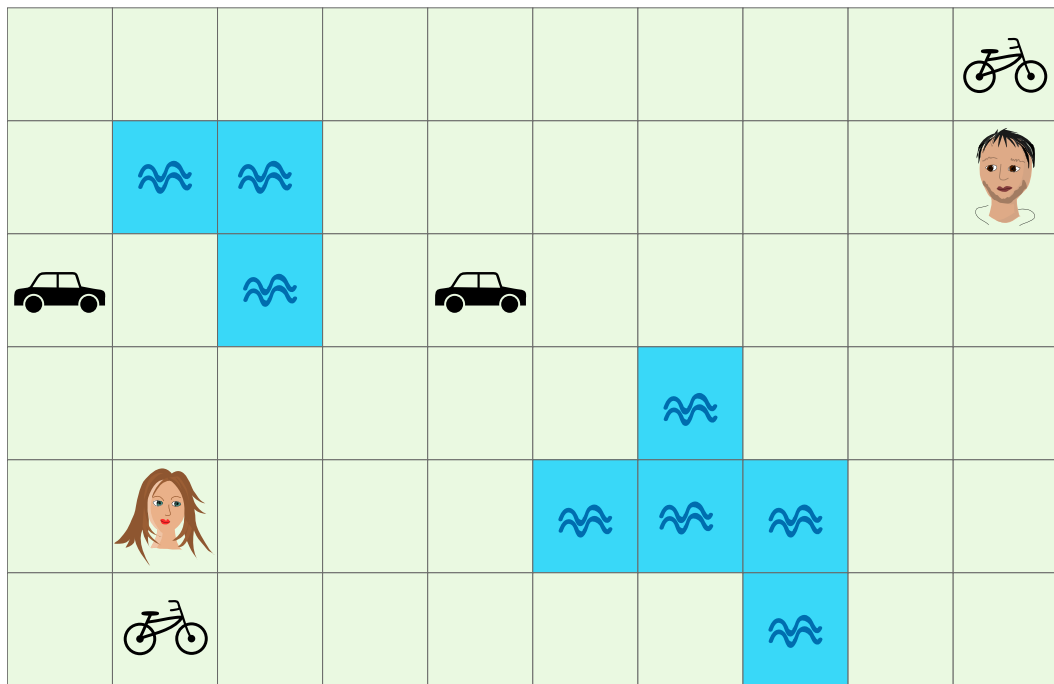


## 22. Es pressiert!

Zwei Freunde wollen sich möglichst bald treffen. Von einem Feld können sie sich zu einem benachbarten Feld links, rechts, oben oder unten bewegen.

Zu Fuss benötigen sie dafür 1 Minute. Wenn sie auf ein Feld mit einem Fahrzeug gelangen, können sie es benutzen.

Mit einem Fahrrad schaffen sie in einer Minute 2 Felder und mit einem Auto 5 Felder. Dabei sind Richtungsänderungen möglich. Wasserflächen können sie nicht überqueren.



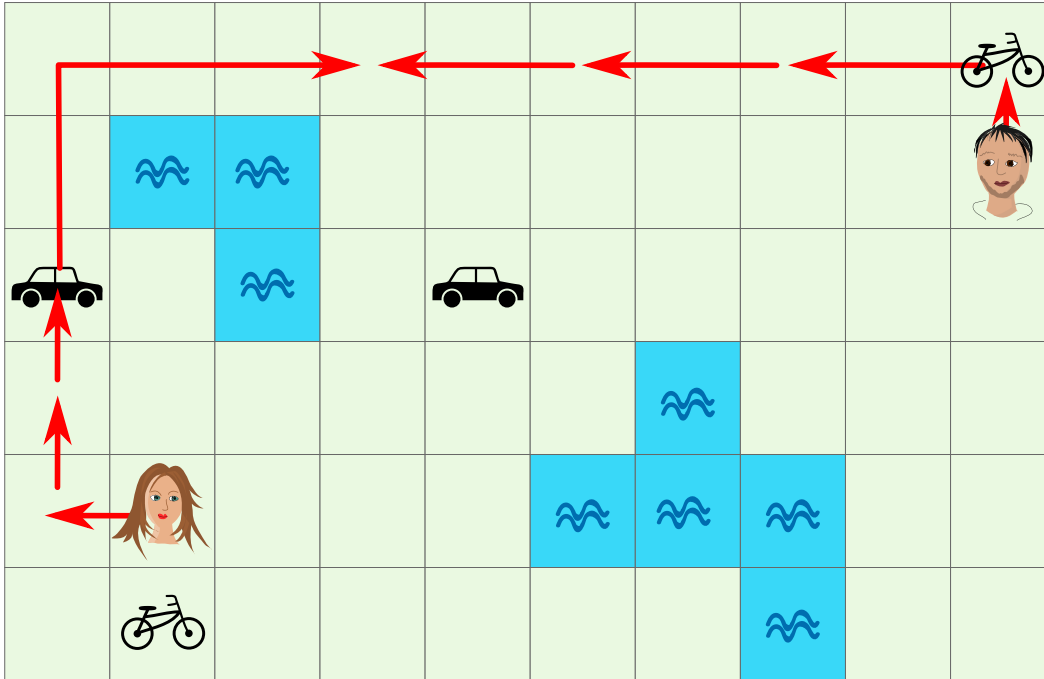
Wie viele Minuten benötigen die beiden Freunde mindestens, um sich auf einem Feld zu treffen?

- A) 1 Minute
- B) 2 Minuten
- C) 3 Minuten
- D) 4 Minuten
- E) 5 Minuten
- F) 6 Minuten



## Lösung

Die richtige Antwort ist D) 4 Minuten. Das Bild zeigt eine Route, mit der sich die beiden Freunde in 4 Minuten auf einem Feld treffen können.



Nun muss noch bewiesen werden, dass sie sich nicht in 3 Minuten treffen können: Die beiden Freunde sind 11 Felder voneinander entfernt. In 3 Minuten können sie aber zu Fuss insgesamt nur 6Felder näher zueinander kommen. Wenn einer das Fahrrad erreicht hat und der andere zu Fuss geht, dann können sie in 3 Minuten 9 Felder näher zueinander kommen, was auch nicht reicht. Selbst wenn beide zu einem Fahrrad gehen, reicht es nicht. Denn dann könnten sie in 3 Minuten 12 Felder näher zueinander kommen, die beiden Fahrräder sind aber 13 Felder voneinander entfernt.

Also bleibt nur die Option, ein Auto zu benützen. In 3 Minuten kann nur das Mädchen ein Auto erreichen. Es bleibt dann aber keine Zeit mehr, das Auto zu benutzen. Und in 3 Minuten kann der Junge kein Feld mit einem Auto erreichen.

## Dies ist Informatik!

Wie hast du die Aufgabe gelöst? Hast du zufällig eine schnelle Route gefunden und gehofft, dass es keine schnellere gibt? Oder hast du viele Möglichkeiten ausprobiert und dir die schnellste gemerkt?

Computerprogramme, die für diese Art von Problemen entwickelt worden sind, arbeiten meist nach einem Verfahren, das man *Breitensuche* nennt. Bei dieser Aufgabe geht die Breitensuche folgendermassen:







zwischen Start und Ziel. Eine Bahnverbindung mit Umsteigen kann schneller sein als eine direkte Busverbindung. In der Informatik kennt man mehrere Verfahren, um die beste Lösung zu einem Problem dieser Art zu finden. Abgesehen von der Breitensuche, die gerade beschrieben worden ist, gibt es auch einen Ansatz, den man *Branch and Bound* nennt (engl. für *Verzweigen und Begrenzen*).

Bei der Breitensuche wird jede Teillösung, die mit einer bestimmten Anzahl von Arbeitsschritten erreicht wird, berücksichtigt. Bei *Branch and Bound* verfolgt man Teillösungen nicht weiter, wenn man weiss, dass sie nicht zur optimalen Lösung führen können.

Wenn ein Problem zu komplex wird, würde es auch für den schnellsten Computer der Welt zu lange dauern, alle Möglichkeiten durchzuspielen, um die beste Lösung zu finden. In der Praxis reicht es bei einem Navigationssystem häufig aus, eine sehr gute Route zu finden, auch wenn es nicht die bestmögliche ist. (Wenn du dein Ziel in 78 Minuten erreichen kannst, macht es dir vermutlich nichts aus, wenn man es theoretisch auch in 77 Minuten erreichen könnte.)

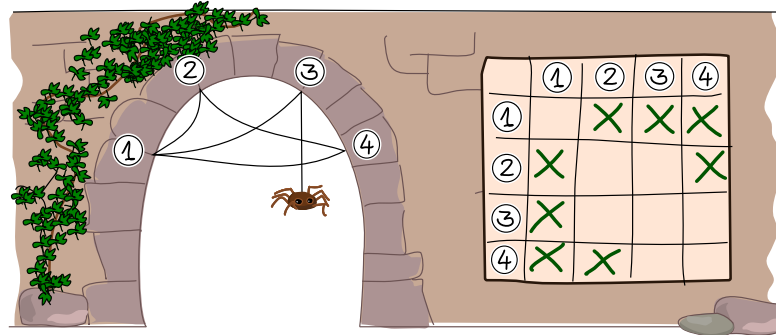
## Stichwörter und Webseiten

- Breitensuche: <https://de.wikipedia.org/wiki/Breitensuche>
- Branch and Bound Algorithmus: <https://de.wikipedia.org/wiki/Branch-and-Bound>



# 23. Theklas Netze

Spinne Thekla möchte möglichst viele verschiedene Netze bauen. Deshalb hat sie sich ein Verfahren ausgedacht, den genauen Aufbau ihrer Netze festzuhalten.

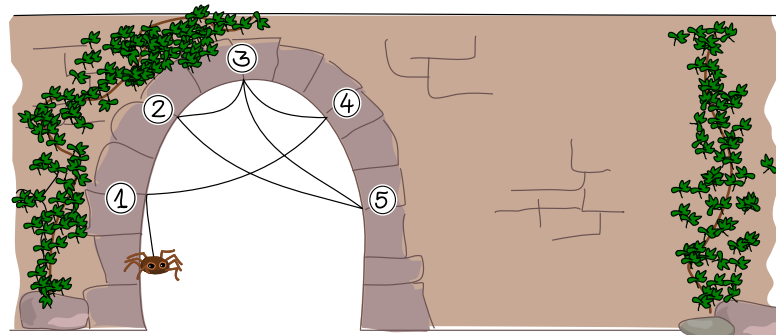


Das macht sie so: Sie nummeriert die Endpunkte des Netzes von 1 bis  $N$  und verwendet Felder in einem Raster nach folgender Regel:

- Wenn es einen Faden gibt, der Endpunkt  $x$  mit Endpunkt  $y$  verbindet, dann wird das Feld in Spalte  $x$  und Zeile  $y$  mit einem Kreuzchen markiert.

Ein Faden der Endpunkt  $x$  mit Endpunkt  $y$  verbindet, verbindet auch Endpunkt  $y$  mit Endpunkt  $x$ .

Thekla baut nun dieses Netz:



Wie hält Thekla den Aufbau dieses Netzes fest?

A)

	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤		X	X		

B)

	①	②	③	④	⑤
①		X		X	
②	X		X		
③		X		X	X
④	X		X		
⑤			X		

C)

	①	②	③	④	⑤
①	X			X	
②			X		X
③		X		X	X
④	X		X	X	
⑤		X	X		

D)

	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤			X		



## Lösung

Antwort A ist richtig, denn alle Felder sind gemäss der Regel richtig markiert.

	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤		X	X		

Bei Antwort B wurde eine zusätzliche Verbindung fälschlich eingezeichnet (Endpunkt 1 nach Endpunkt 2 in beide Richtungen) und eine wurde vergessen (Endpunkt 2 nach Endpunkt 5 in beide Richtungen).

	①	②	③	④	⑤
①		<del>X</del>		X	
②	<del>X</del>		X		<del>X</del>
③		X		X	X
④	X		X		
⑤		<del>X</del>	X		

Antwort C: Nach der hier vorgeschriebenen Regel können in der Diagonale von links oben nach rechts unten keine Markierung vorhanden sein. Denn das wären ja Verbindungen eines Endpunkts mit sich selbst. Dies könnte zwar in manchen Netzen durchaus erlaubt sein kann, kommt in unserem Spinnennetz aber nicht vor. Bei Antwort C hätten wir aber 2 solche Verbindungen (bei Endpunkt 1 und bei Endpunkt 4).

	①	②	③	④	⑤
①	<del>X</del>			X	
②			X		X
③		X		X	X
④	X		X	<del>X</del>	
⑤		X	X		

Antwort D: Alle Darstellungen von Netzen sollten symmetrisch sein bezüglich der Diagonalen von links oben nach rechts unten. In dieser Antwort findet sich zwar die Verbindung von Endpunkt 2 nach Endpunkt 5, die entsprechende Verbindung zurück von Endpunkt 5 nach Endpunkt 2 fehlt aber.

	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤		<del>X</del>	X		

## Dies ist Informatik!

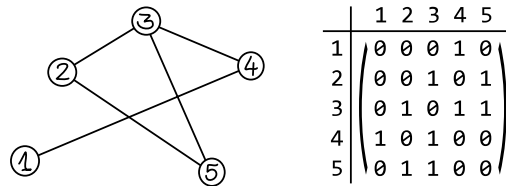
Das Spinnennetz kann als *Graph* betrachtet werden, ein Konzept, das häufig in der Informatik verwendet wird.

Ein Graph besteht aus *Knoten* (den Endpunkten des Netzes) und *Kanten* (den Fäden zwischen den Endpunkten). Graphen werden zum Beispiel auch verwendet, um Objekte und die Beziehungen zwischen Objekten darzustellen. Ein Graph könnte beispielsweise zeigen, wie Personen in sozialen Medien befreundet sind, oder Flüge zwischen Ländern.

In dieser Aufgabe wird gezeigt, wie man die Struktur eines Spinnennetzes in einem Raster speichern kann. Gewisse Eigenschaften, wie zum Beispiel das genaue Aussehen des Netzes, gehen dabei verloren.



In vielen Fällen ist man aber nicht an den genauen geometrischen Eigenschaften eines Netzes interessiert, sondern nur an seiner Struktur. Die wesentlichen Informationen bleiben erhalten: Wie viele Knoten gibt es? Und zwischen welchen Knotenpaaren gibt es eine Kante?



Die vorgestellte Möglichkeit ist nur eine von vielen Möglichkeiten, die Struktur eines Netzes festzuhalten. Die Methode ist nicht sehr sparsam, denn es werden für jede Verbindung beide Richtungen gespeichert, was nicht notwendig wäre, und auch die freien Diagonalfelder wären gar nicht notwendig. Dafür weist dieses Verfahren den Vorteil auf, dass Darstellungsfehler teilweise erkannt werden können. Antwort C und Antwort D konnten zum Beispiel als falsch erkannt werden, ohne auf das Netz Bezug zu nehmen.

Die vorgestellte Darstellungsform wird *Adjazenzmatrix* genannt.

## Stichwörter und Webseiten

- Adjazenzmatrix: <https://de.wikipedia.org/wiki/Adjazenzmatrix>









## 24. Frucht auf Frucht

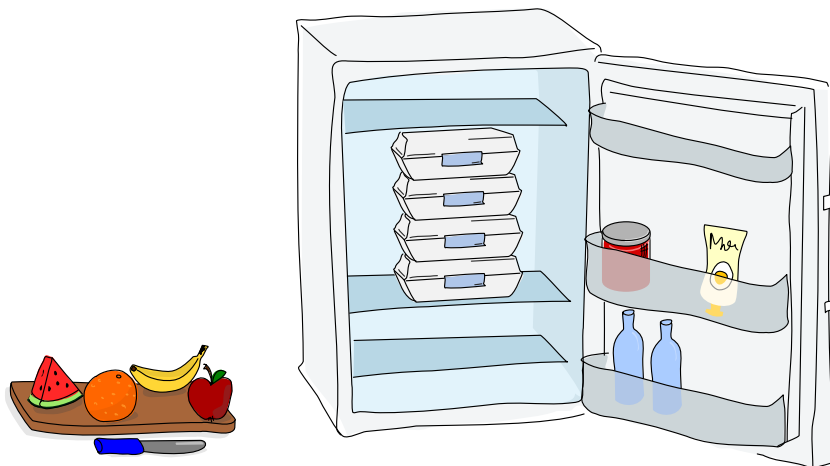
Paps, Mams, Dorie und Ron Biber packen abends für den nächsten Tag vier Frühstücksboxen, jede mit einer anderen Frucht: Apfel, Banane, Orange und Wassermelone. Die Boxen werden im Kühlschrank aufeinander gestapelt. Morgens sind die Bibers noch sehr müde und nehmen sich beim Verlassen des Baus einfach die oberste Box, ohne sie genauer anzuschauen.

Man weiss nicht genau, in welcher Reihenfolge die Bibers den Bau verlassen, aber auf jeden Fall geht Mams vor Dorie und Paps immer als Letzter.

Die Familienmitglieder mögen unterschiedliche Früchte. Die Tabelle gibt an, was jedes Familienmitglied mag.

				
<b>Paps</b>	—	—	✓	—
<b>Mams</b>	✓	—	✓	✓
<b>Dorie</b>	✓	✓	✓	—
<b>Ron</b>	✓	✓	—	✓

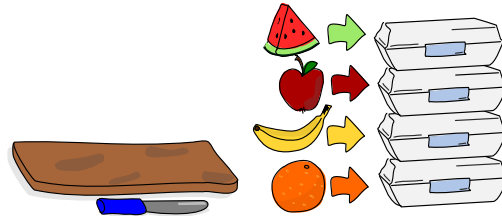
Packe die Früchte so in die Boxen, dass alle Bibers eine Box mit Früchten nehmen, die sie mögen.





## Lösung

Es gibt nur eine Möglichkeit, die Früchte so zu verteilen, dass alle garantiert etwas bekommen, das sie mögen:



Paps mag nur Orangen und geht als Letzter. Also kommen Orangen in die unterste Box. Ron geht als erster, zweiter oder dritter. Mams geht ausserdem vor Dorie, daher ist die Reihenfolge klar, wenn man weiss, wann Ron den Bau verlässt. Folgende drei Reihenfolgen sind möglich:

1. Mams Mams Ron
2. Dorie Ron Mams
3. Ron Dorie Dorie
4. Paps Paps Paps

Mams, Dorie und Ron könnten also alle als zweite gehen. In die zweite Box muss daher eine Frucht hinein, die alle drei mögen, und das erfüllt nur der Apfel. Für die oberste Box bleiben also nur Bananen und Wassermelone. Da Mams Bananen nicht mag, müssen wir hier die Wassermelone zuordnen. Damit bleiben die Bananen für die dritte Box.

## Dies ist Informatik!

Die richtige Reihenfolge ist in vielen Bereichen der Informatik sehr relevant: Viele Berechnungen erfordern Zwischenergebnisse, die zunächst ermittelt werden müssen, bevor man zum Endresultat kommt. Werden die Rechenschritte auf unterschiedlichen Computern durchgeführt, könnten ohne sorgfältige Planung sogenannte *Deadlocks* entstehen. Das sind Situationen, in denen zwei oder mehr Computer aufeinander warten und auf diese Weise das Programm nie zum Ende kommt. Die falsche Reihenfolge führt meistens aber einfach zu Fehlern (wie bei den Bibers zu Missmut über die erwischten Früchte). Wenn zum Beispiel etwas mit der Formel  $Z \leftarrow (A + B) \cdot (A - B)$  berechnet werden soll, kann man das in folgende Schritte eines Programms aufteilen:

```
Eingabe A
Eingabe B
Berechne  $X \leftarrow A + B$ 
Berechne  $Y \leftarrow A - B$ 
Berechne  $Z \leftarrow X \cdot Y$ 
```

Versucht man aber z. B. den Rechenschritt  $Z \leftarrow X \cdot Y$  auszuführen, bevor man  $X$  berechnet hat, führt dies zu einem Fehler und dem Abbruch des Programms. Oder es wird ein Standardwert für  $X$  verwendet, was meistens zu einem falschen Resultat führt. Beim Programmieren ist also die Reihenfolge, in der die Anweisungen durchgeführt werden, relevant.





## Stichwörter und Webseiten

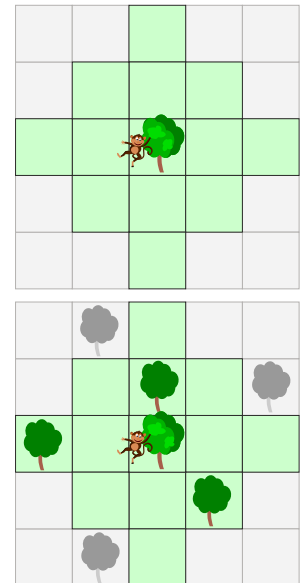
- Deadlock: [https://de.wikipedia.org/wiki/Deadlock\\_\(Informatik\)](https://de.wikipedia.org/wiki/Deadlock_(Informatik))





## 25. Kletteräffchen Koko

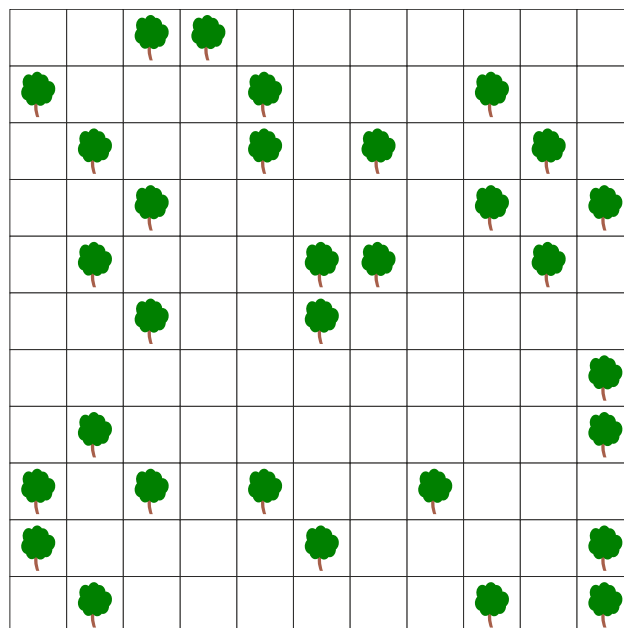
Kletteräffchen Koko kann von einem Baum so weit springen, wie es der grüne Bereich



In folgendem Beispiel erreicht Koko die farbigen Bäume mit einem Sprung. Mit zwei Sprüngen sind auch die beiden grauen Bäume oben erreichbar, nicht aber der graue Baum unten.

Es gibt Gruppen von Bäumen, zwischen denen sich Koko mit mehreren Sprüngen beliebig bewegen kann, ohne jemals den Boden zu berühren.

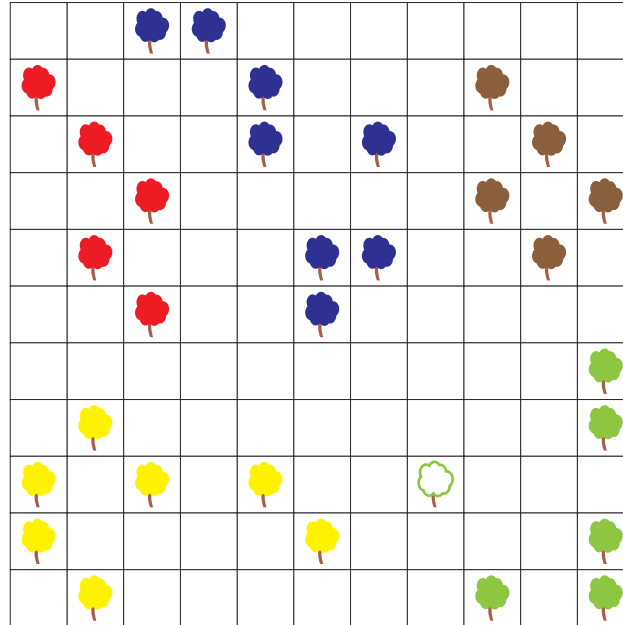
*Markiere alle Bäume der grössten solchen Gruppe.*





## Lösung

Im Bild unten haben zwei Bäume dieselbe Farbe, wenn Koko vom einen zum anderen gelangen kann, ohne den Boden zu berühren.

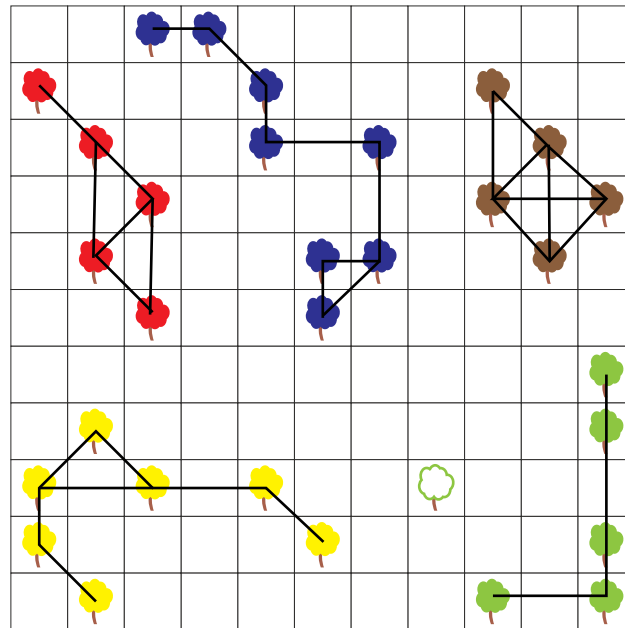


Wir sehen, dass die blaue Baumgruppe mit ihren acht Bäumen die grösste ist.

## Dies ist Informatik!

Wenn Koko von einem Baum direkt zum nächsten springen kann, sind sie quasi miteinander verbunden. Wir können dies als eine Linie zwischen den Bäumen darstellen, so wie unten gezeigt. Wir haben also einen Graphen mit Bäumen als Knoten und Kanten zwischen verbundenen Bäumen. Koko kann genau dann springend von einem Baum zu einem anderen gelangen, wenn die Kanten einen Weg zwischen den beiden Bäumen bilden.

Wir nennen eine Gruppe von Knoten *zusammenhängend*, wenn sie alle über Kanten miteinander verbunden sind. Wenn wir eine solche Gruppe nicht mehr grösser machen können, ohne den Zusammenhang zwischen ihnen zu verlieren, dann sprechen wir von einer *Zusammenhangskomponente*. Ein Graph lässt sich eindeutig in solche Zusammenhangskomponenten aufteilen, unten sind sie farblich markiert.



Eine Zusammenhangskomponente lasst sich einfach bestimmen, indem man bei einem beliebigen Knoten beginnt und dann alle ber Kanten erreichbaren Knoten sucht.




## Stichwortер und Webseiten




- Zusammenhangskomponenten,  
[https://de.wikipedia.org/wiki/Zusammenhang\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Zusammenhang_(Graphentheorie))





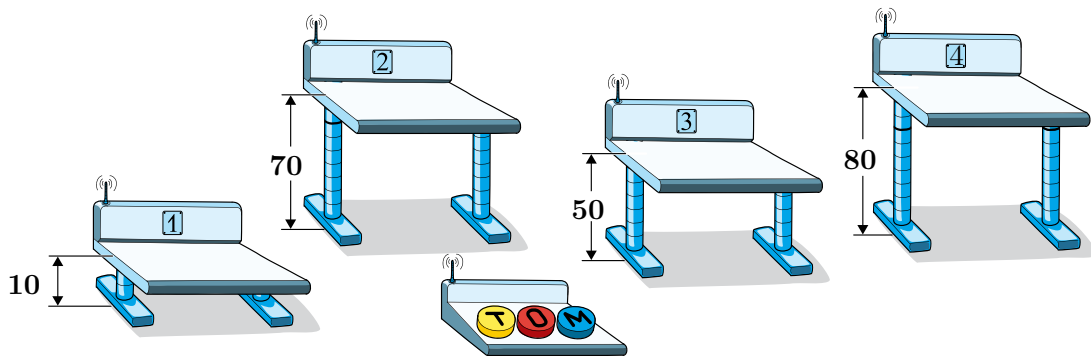
## 26. Verflixte Pulte

Im Schulzimmer hat es Pulte mit elektrisch einstellbarer Höhe. Für den Unterricht sollten alle Pulte auf die Höhe 60 cm eingestellt sein. Mit den Tasten ,  und  einer Fernbedienung kann die Höhe der Pulte verändert werden. Jemand hat mit der Fernbedienung gespielt und sie umprogrammiert. Jetzt funktionieren die drei Tasten folgendermassen:













-  erhöht jedes der Pulte 1, 2 und 3 um jeweils 10 cm.
-  senkt jedes der Pulte 2, 3 und 4 um jeweils 10 cm.
-  erhöht jedes der Pulte 1, 3 und 4 um jeweils 10 cm.

Diese Aktionen werden jedes Mal ausgeführt, wenn die Taste gedrückt wird.

Momentan sind die Höhen der Pulte 1, 2, 3 und 4 auf 10 cm, 70 cm, 50 cm und 80 cm eingestellt:
















Wie kann die Höhe für alle vier Pulte auf 60 cm eingestellt werden?


- A) Drücke 4 × , 5 ×  und 1 × .
- B) Drücke 5 × , 1 ×  und 0 × .
- C) Drücke 3 × , 4 ×  und 2 × .
- D) Drücke 2 × , 4 ×  und 6 × .







## Lösung

Die korrekte Antwort ist C) Drücke  $3 \times$  ,  $4 \times$   und  $2 \times$  . Bei der Fernbedienung stellst du fest, dass alle drei Tasten mit jedem Mal Drücken die Höhe um 10 cm verändern, also immer um denselben Betrag. Zwei der Tasten lassen die Pulte höher werden ( und ) und nur eine Taste senkt die Pulte ab (). Zudem verändern alle drei Tasten die Höhe von jeweils drei Pulten, es bleibt also immer ein Pult, dessen Höhe unverändert bleibt. Die Taste  hat keinen Einfluss auf Pult 1, wir können also Pult 1 gar nicht nach unten fahren.



Pult 1 ist um 50 cm zu tief. Daraus schliessen wir, dass wir genau 5 Mal die Taste  oder  drücken müssen (die Anzahl der Tastendrucke auf  und auf  muss zusammengerechnet genau 5 sein). Das kann man als Gleichung ausdrücken:  $T + M = 5$ . Damit können wir Lösung D) ausschliessen, weil dort  $T + M = 8$  gilt. Nach der Tastenfolge von Lösung D) hätte das Pult 1 die Höhe  $10 + 20 + 60 = 90$  cm, nämlich die Ausgangshöhe 10 cm plus  $2 \cdot 10$  cm für  plus  $6 \cdot 10$  cm für .

Pult 2 ist 10 cm zu hoch.  hat keinen Einfluss auf Pult 2. Somit gilt für die richtige Lösung  $T - O = -1$ . Damit können wir die Lösung B) ausschliessen, denn damit hätte das Pult 2 am Ende die Höhe  $70 + 50 - 10 = 110$ .

Pult 3 ist 10 cm zu tief, also gilt:  $T - O + M = 1$ . Damit können die Lösungen A) und B) ausgeschlossen werden. Bei A) wäre die Höhe von Pult 3 am Ende wieder dieselbe, nämlich  $50 + 40 - 50 + 10 = 50$  cm; bei B) wäre die Höhe von Pult 3 am Ende  $50 + 50 - 10 = 90$  cm. Nun sind alle Lösungen ausgeschlossen bis auf Lösung C).

Es muss aber noch geprüft werden, ob Lösung C) auch für Pult 4 die richtige Höhe ergibt. Pult 4 ist 20 cm zu hoch und  hat keinen Einfluss auf die Höhe von Pult 4. Also muss zwei Mal  gedrückt werden und für jedes Drücken von  braucht es ein zusätzliches Drücken von . Mit der Tastenfolge von Lösung C) ist die Höhe von Pult 4 am Ende  $80 - 40 + 20 = 60$  cm.

Da wir schon vorher für die Pulte 1, 2 und 3 festgestellt hatten, dass Lösung C) funktioniert, sind wir sicher, dass diese Lösung funktioniert.

Alternativ kann die Lösung mit vier linearen Gleichungen gesucht werden. Für jedes Pult schreibt man mit einer Gleichung auf, welche Tasten die Höhe des Pultes verändern und was die gesuchte Höhenveränderung ist. Beispielsweise verändert sich die Höhe von Pult 1 nur mit  und  und die gewünschte Höhenanpassung ist 50 cm, was man mit 5 Tastendrücken erreichen kann (weil es pro Tastendruck 10 cm sind).

Da es vier Pulte und drei Tasten sind, ergeben sich vier lineare Gleichungen mit drei Unbekannten:

$$\begin{array}{l}
 T + M = 5 \\
 T - O = -1 \\
 T - O + M = 1 \\
 -O + M = -2
 \end{array}$$

Wenn man die dritte Gleichung von der ersten abzieht, erhält man  $O = 4$ . Eingesetzt in die zweite Gleichung erhalten wir  $T = 3$ . Nur für  $M = 2$  sind alle Gleichungen erfüllt. Somit ist es die einzige Lösung.





## Dies ist Informatik!

Dies ist eine typische Aufgabe aus dem Bereich der *diskreten Optimierung*, genauer spezifiziert, der *linearen Programmierung*. Diese Aufgabe ist gegeben durch eine Menge von Einschränkungen. In diesem Spezialfall kann man sie alle als lineare Gleichungen formulieren. Die Zielsetzung ist typisch für die Informatik. Man sucht eine Folge von Aktionen, die zu einem vorgegebenen Ziel führen. Man könnte sogar die ganze Aufgabe als die Suche nach einem Weg in einem vierdimensionalen Raum mit drei erlaubten Bewegungsaktionen beschreiben, nämlich vom Punkt  $(10, 70, 50, 80)$  zum Punkt  $(60, 60, 60, 60)$ . In dieser Aufgabe gibt es nur eine Lösung, aber solche Aufgaben haben oft viele Lösungen, was eine Optimierung als Zielsetzung ermöglicht. Dann sucht man das Minimum der linearen Funktion  $T + M + O$ .

## Stichwörter und Webseiten

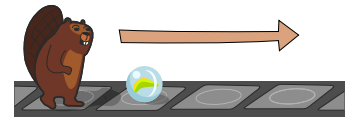
- Diskrete Optimierung, ganzzahlige lineare Optimierung:  
[https://de.wikipedia.org/wiki/Ganzzahlige\\_lineare\\_Optimierung](https://de.wikipedia.org/wiki/Ganzzahlige_lineare_Optimierung)
- Erreichbarkeit in gerichteten Konfigurationsgraphen:  
[https://de.wikipedia.org/wiki/Erreichbarkeitsproblem\\_in\\_Graphen](https://de.wikipedia.org/wiki/Erreichbarkeitsproblem_in_Graphen)





## 27. Murrelband

Der Biber bewegt sich Feld für Feld von links nach rechts über ein Band. Auf jedem Feld des Bands kann eine Murrel liegen.



Immer, wenn der Biber auf ein Feld mit einer Murrel kommt und er die Hände frei hat, hebt er die Murrel auf und trägt sie dann mit sich.



Beim ersten freien Feld legt er die Murrel wieder ab.



Der Biber kann immer nur eine Murrel tragen und auf jedem Feld hat nur eine Murrel Platz.

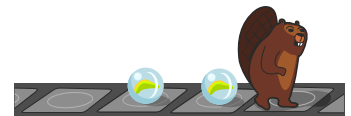
Trägt der Biber schon eine Murrel, wenn er ein Feld mit einer anderen Murrel erreicht, ...



... dann geht er an ihr vorbei ...



... und legt seine Murrel auf das nächste freie Feld.



Danach kann er die nächste Murrel wieder aufheben.

*Der Biber steht vor einem Bereich des Bandes, auf dem drei Murreln liegen. Auf welchen Feldern befinden sich die Murreln, wenn der Biber den Bereich überquert hat?*



- A)
- B)
- C)
- D)
- E)

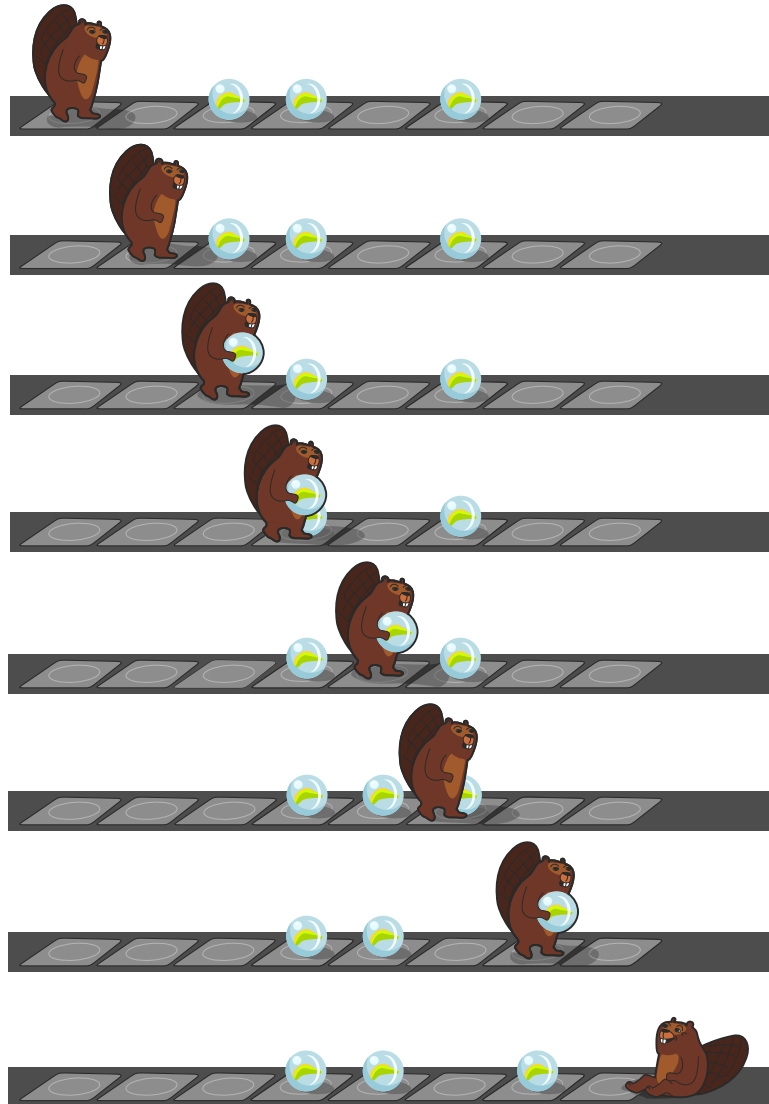


## Lösung

Die richtige Antwort ist C.



Die folgende Abbildung zeigt den Ablauf:



## Dies ist Informatik!

In der Informatik führen oft auch relativ einfache Operationen zu interessanten Ergebnissen. Diese Aufgabe ist ein gutes Beispiel dafür. Das Vorgehen des Bibers ist ein *Algorithmus*. Er beruht darauf, dass der Biber 2 verschiedene Zustände einnehmen kann (Murmel tragend oder nicht) und dass er auf seinem Weg 2 verschiedene Arten von Feldern vorfinden kann (belegte und leere).



So einfach das Beispiel in dieser Biberaufgabe ist, enthält es doch einige der wesentlichen Elemente einer *Turingmaschine*. Eine Turingmaschine (benannt nach dem Mathematiker Alan Turing) ist ein spezieller, sehr simpel strukturierter Computer. Eine Turingmaschine kann grundsätzlich alle Algorithmen ausführen, die ein herkömmlicher Computer ausführen kann. In der Praxis werden trotzdem keine Turingmaschinen als Computer verwendet, denn wir können Computer bauen, die zwar komplizierter, aber viel effizienter sind. Turingmaschinen sind vor allem in der Theorie nützlich. Durch ihre simple Struktur kann man relativ einfach Aussagen über Turingmaschinen beweisen. Und wenn man beweisen kann, dass eine Aufgabe für Turingmaschinen nicht lösbar ist, dann kann keiner unserer Computer sie lösen.

Eine Turingmaschine besteht aus:

- Einem beliebig langen *Band*, bestehend aus einzelnen *Feldern*. In jedem Feld kann ein *Symbol* stehen. Das sind bei unserem Beispiel die Felder, über die sich der Biber bewegt.
- Einer endlichen Menge von *Symbolen*. Oft benützt man nur 0 und 1 als Symbole. In unserem Beispiel steht eine Murrel für 1 und eine freie Stelle für 0.
- Einem Lese-Schreib-Kopf, der sich auf dem Band in beide Richtungen bewegen und dabei die Symbole auf dem Band lesen und auch neue Symbole schreiben kann. In unserem Beispiel hat der Biber die Rolle des Lese-Schreib-Kopfs.
- Einer endlichen Menge von sogenannten *Zuständen*. Das Verhalten des Lese-Schreib-Kopfs kann sich je nach Zustand ändern. In unserem Fall gibt es nur zwei Zustände, nämlich «Murrel tragend» und «nicht Murrel tragend».
- Einer Menge von Regeln: Was passiert, abhängig vom Zustand, wenn ein bestimmtes Symbol vom Band gelesen wird? Mögliche Aktionen sind: ein Wechseln des Zustands, das Schreiben eines neuen Symbols auf das Band und das Bewegen des Lese-Schreib-Kopfs um ein Feld nach links oder rechts.

## Stichwörter und Webseiten

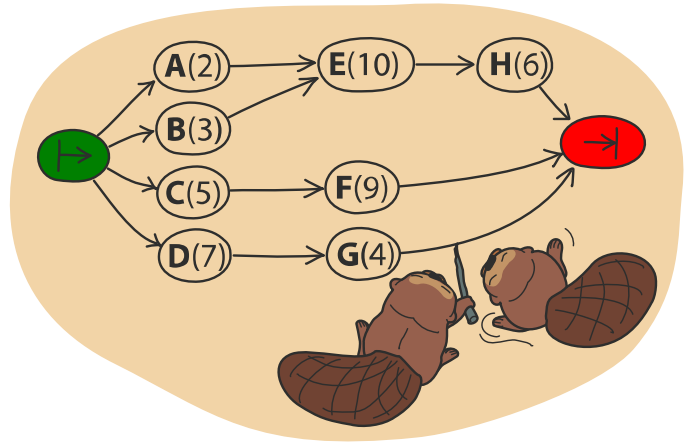
- Turingmaschine: <https://de.wikipedia.org/wiki/Turingmaschine>



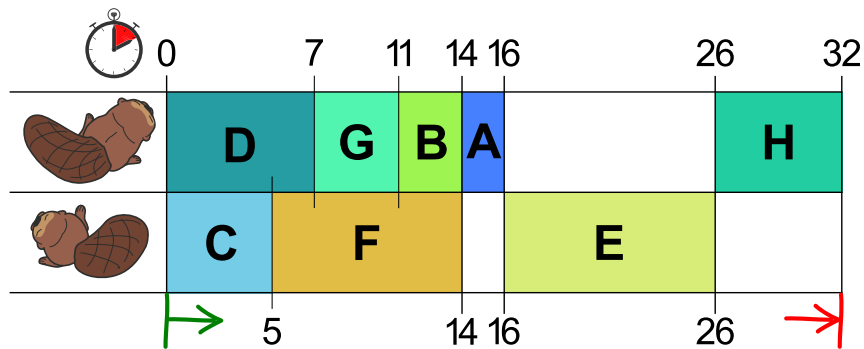


## 28. Biber-Arbeit

Das Bauen eines Biberdamms lässt sich in mehrere Teilaufgaben zerlegen (Bäume fällen, Äste entfernen, Stämme zum Wasser transportieren, usw.). Das Bild rechts zeigt alle 8 Teilaufgaben A, B, C, D, E, F, G, H, jeweils mit der Anzahl der Stunden, die man zu ihrer Erledigung braucht. Die Teilaufgaben sind nicht ganz unabhängig voneinander: Ein Pfeil von X nach Y bedeutet, dass Teilaufgabe X vollständig erledigt sein muss, bevor man mit Teilaufgabe Y anfängt.

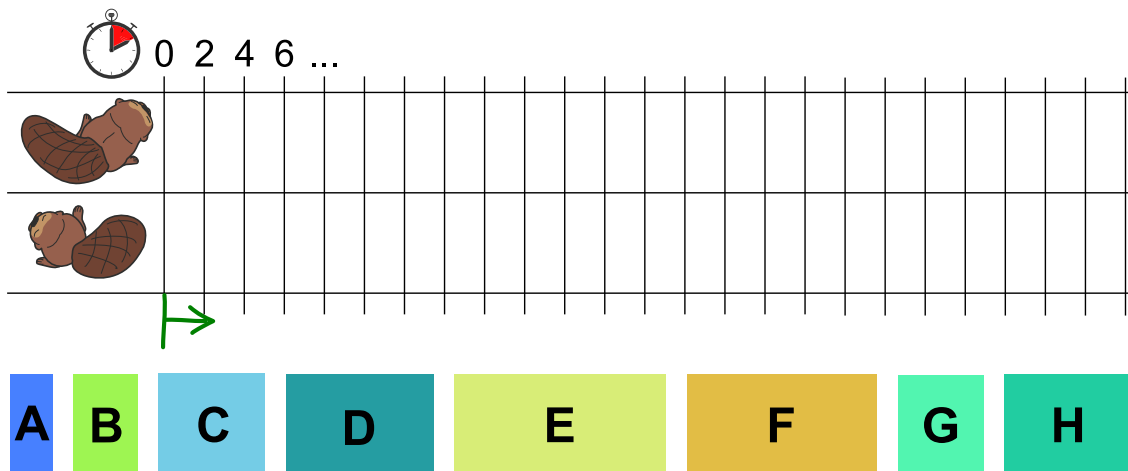


Biberin Ulla will dem Biber Otso helfen, den Damm schneller zu bauen. Sie teilen die Teilaufgaben untereinander auf und erstellen folgenden Arbeitsplan, der die Abhängigkeiten aus dem Bild oben erfüllt.



Damit würde der Damm in 32 Stunden fertig. Das geht aber schneller!

*Erstelle einen Arbeitsplan, mit dem der Damm in möglichst kurzer Zeit fertig wird.*

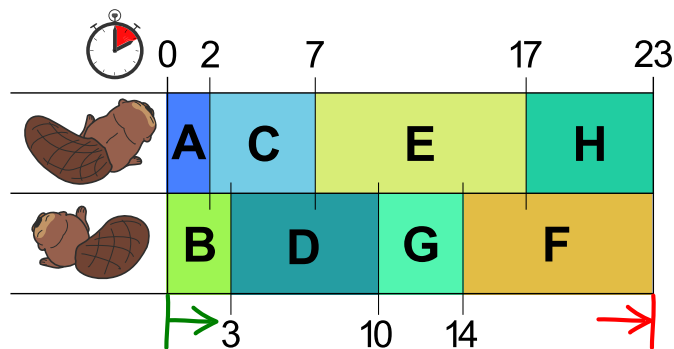




## Lösung

Im Arbeitsplan aus der Aufgabe hat der erste Biber eine lange Pause (10 Stunden), und der zweite Biber hat insgesamt 8 Stunden lang Leerlauf. Die beiden wären schneller fertig, wenn sie ständig arbeiteten.

Man kommt zu einem insgesamt schnelleren Arbeitsplan, wenn man darauf achtet, dass die beiden grössten Aufgaben E(10) und F(9) nicht vom selben Biber ausgeführt werden. Hier ist ein möglicher Arbeitsplan, der mit 23 Stunden auskommt. Schneller geht es nicht, denn die beiden Biber arbeiten ohne Pause.



## Dies ist Informatik!

Menschen sind ungeduldig, und deshalb wird oft verlangt, dass Arbeiten möglichst schnell erledigt werden. Wenn eine Arbeit aufgeteilt werden kann (ob nun unter mehreren Menschen, Maschinen oder Bibern), dann spielt die Verteilung der Teilaufgaben auf die Arbeitenden für die Schnelligkeit eine wichtige Rolle. Genauso müssen auch Computer ihre Berechnungsaufgaben oft geschickt auf verschiedene Prozessorkerne aufteilen.

Für solche *Scheduling-Probleme* gibt es viele verschiedene, von der Informatik gut untersuchte Strategien. Der erste Arbeitsplan in dieser Biberaufgabe wurde so erstellt, dass unter den anstehenden Teilaufgaben, jene mit der längsten Dauer einem gerade arbeitslosen Biber zugeteilt wurde – in diesem Fall eine schlechte Strategie. Oft funktioniert es besser, wenn kurze Teilaufgaben zuerst erledigt werden: Die Strategie *Shortest-Job-Next* (engl. für kürzeste Teilaufgabe zuerst) minimiert zudem auch die durchschnittliche Wartezeit bis zur Fertigstellung pro Teilaufgabe.

## Stichwörter und Webseiten

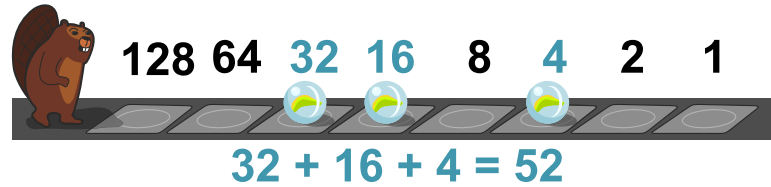
- Scheduling: <https://de.wikipedia.org/wiki/Scheduling>
- Shortest-Job-Next: <https://de.wikipedia.org/wiki/Shortest-Job-Next>





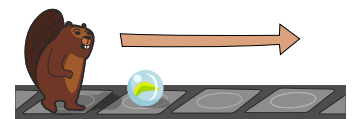
## 29. Marmelzählen

Die Biber haben eine besondere Art, Zahlen darzustellen.



Die verschiedenen Felder haben verschiedene Gewichtungen und eine Marmel auf dem Feld bestimmt, dass der Wert übernommen wird. Im Beispiel oben wird die Zahl 52 dargestellt.

Der Biber bewegt sich Feld für Feld von links nach rechts über ein Band. Auf manchen Feldern des Bands können Marmeln liegen.



Immer, wenn der Biber auf ein Feld mit einer Marmel kommt und er die Hände frei hat, hebt er die Marmel auf und trägt sie dann mit sich.



Beim ersten freien Feld legt er die Marmel wieder ab.



Der Biber kann immer nur eine Marmel tragen und auf jedem Feld hat nur eine Marmel Platz.

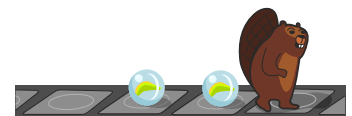
Trägt der Biber schon eine Marmel, wenn er ein Feld mit einer anderen Marmel erreicht, ...



... dann geht er an ihr vorbei ...



... und legt seine Marmel auf das nächste freie Feld.



Danach kann er die nächste Marmel wieder aufheben.

Welche Zahl wird durch die Marmeln dargestellt, wenn der Biber den Bereich überquert hat?

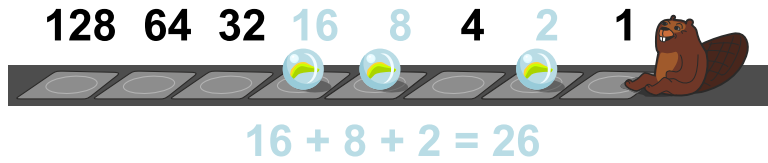


- A) 10
- B) 26
- C) 28
- D) 104

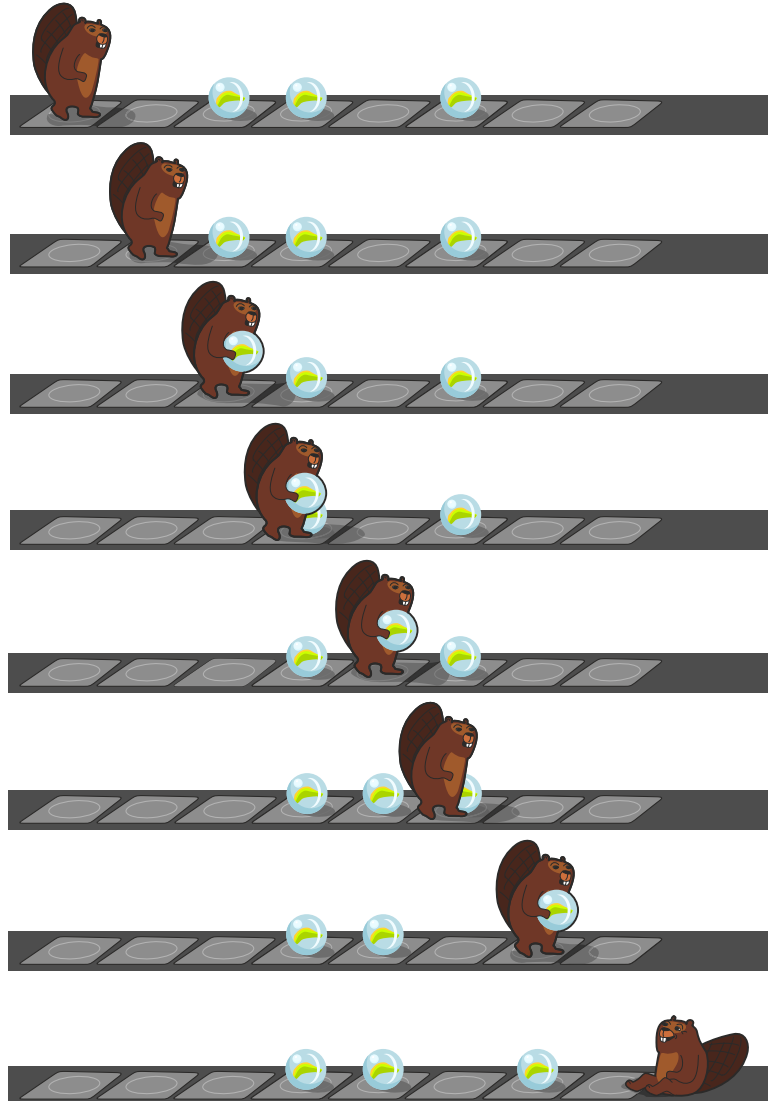


## Lösung

Die richtige Antwort ist B) 26



Die folgende Abbildung zeigt den Ablauf:



## Dies ist Informatik!

In der Informatik führen oft auch relativ einfache Operationen zu interessanten Ergebnissen. Diese Aufgabe ist ein gutes Beispiel dafür. Das Vorgehen des Bibers ist ein *Algorithmus*. Er beruht darauf, dass der Biber 2 verschiedene Zustände einnehmen kann (Murmel tragend oder nicht) und dass er auf seinem Weg 2 verschiedene Arten von Feldern vorfinden kann (belegte und leere).



Insgesamt ist das Resultat nach Ausführen des Algorithmus genau so, als hätte man jede Marmel auf dem Band um eins nach rechts verschoben. In der Zahlendarstellung der Biber entspricht dies einer Division durch 2. Würde der Biber von rechts nach links über das Band marschieren, würde die Zahl hingegen mit 2 multipliziert. Wenn in einer Reihe von Nullen und Einsen alles gleich viel nach links oder nach rechts vorschoben wird, bezeichnet man das in der Information oft als *Bitshift*.

So einfach das Beispiel in dieser Biberaufgabe ist, enthält es doch einige der wesentlichen Elemente einer *Turingmaschine*. Eine Turingmaschine (benannt nach dem Mathematiker Alan Turing) ist ein spezieller, sehr simpel strukturierter Computer. Eine Turingmaschine kann grundsätzlich alle Algorithmen ausführen, die ein herkömmlicher Computer ausführen kann. In der Praxis werden trotzdem keine Turingmaschinen als Computer verwendet, denn wir können Computer bauen, die zwar komplizierter, aber viel effizienter sind. Turingmaschinen sind vor allem in der Theorie nützlich. Durch ihre simple Struktur kann man relativ einfach Aussagen über Turingmaschinen beweisen. Und wenn man beweisen kann, dass eine Aufgabe für Turingmaschinen nicht lösbar ist, dann kann keiner unserer Computer sie lösen.

Eine Turingmaschine besteht aus:

- Einem beliebig langen *Band*, bestehend aus einzelnen *Feldern*. In jedem Feld kann ein *Symbol* stehen. Das sind bei unserem Beispiel die Felder, über die sich der Biber bewegt.
- Einer endlichen Menge von *Symbolen*. Oft benützt man nur 0 und 1 als Symbole. In unserem Beispiel steht eine Marmel für 1 und eine freie Stelle für 0.
- Einem Lese-Schreib-Kopf, der sich auf dem Band in beide Richtungen bewegen und dabei die Symbole auf dem Band lesen und auch neue Symbole schreiben kann. In unserem Beispiel hat der Biber die Rolle des Lese-Schreib-Kopfs.
- Einer endlichen Menge von sogenannten *Zuständen*. Das Verhalten des Lese-Schreib-Kopfs kann sich je nach Zustand ändern. In unserem Fall gibt es nur zwei Zustände, nämlich «Marmel tragend» und «nicht Marmel tragend».
- Einer Menge von Regeln: Was passiert, abhängig vom Zustand, wenn ein bestimmtes Symbol vom Band gelesen wird? Mögliche Aktionen sind: ein Wechseln des Zustands, das Schreiben eines neuen Symbols auf das Band und das Bewegen des Lese-Schreib-Kopfs um ein Feld nach links oder rechts.

## Stichwörter und Webseiten

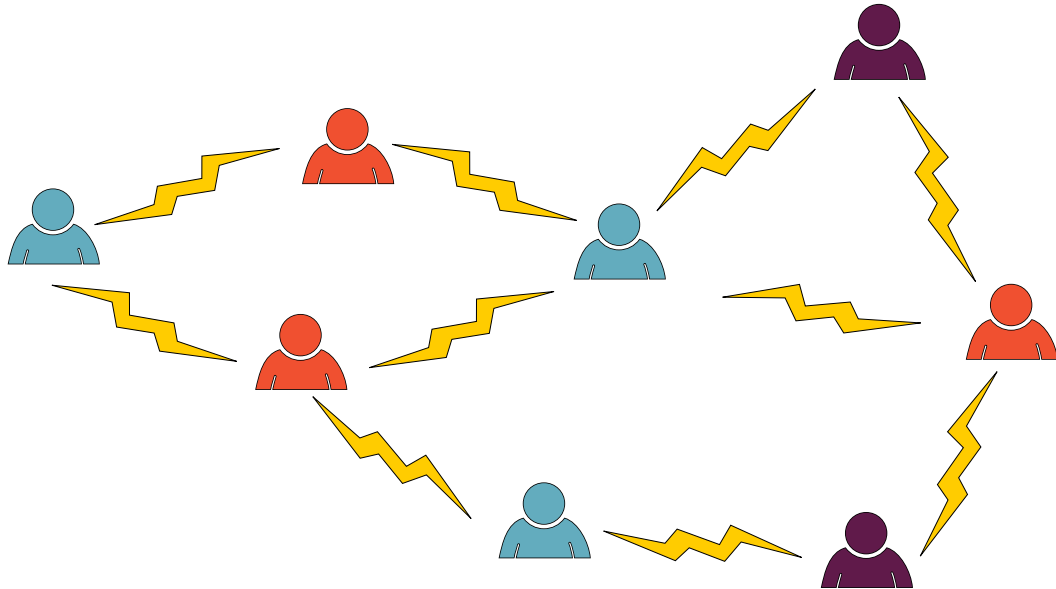
- Turingmaschine: <https://de.wikipedia.org/wiki/Turingmaschine>
- Bitweiser Operator, Bitweise Verschiebungen:  
[https://de.wikipedia.org/wiki/Bitweiser\\_Operator#Bitweise\\_Verschiebungen](https://de.wikipedia.org/wiki/Bitweiser_Operator#Bitweise_Verschiebungen)





## 30. Teamwork

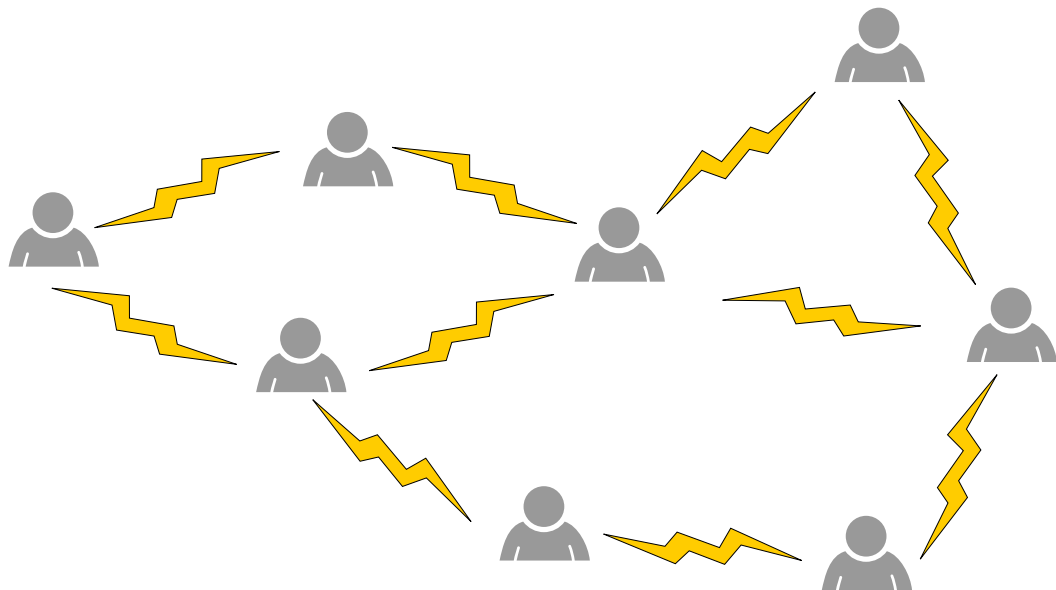
Für ein Projekt sollst du acht Personen in Gruppen aufteilen. Zwischen zwei Personen steht ein Blitz, wenn sie nicht zusammenarbeiten wollen. Dann möchtest du sie nicht derselben Gruppe zuordnen.



Mit den Abneigungen im Beispiel oben ist eine Aufteilung in drei Gruppen (rot, blau, violett) möglich. Zwischen zwei Personen derselben Farbe steht also nirgends ein Blitz.

Wenn du die richtigen beiden Personen zur Zusammenarbeit überzeugst (also einen Blitz entfernst), dann ist sogar eine Aufteilung in nur zwei Gruppen (nur zwei Farben) möglich.

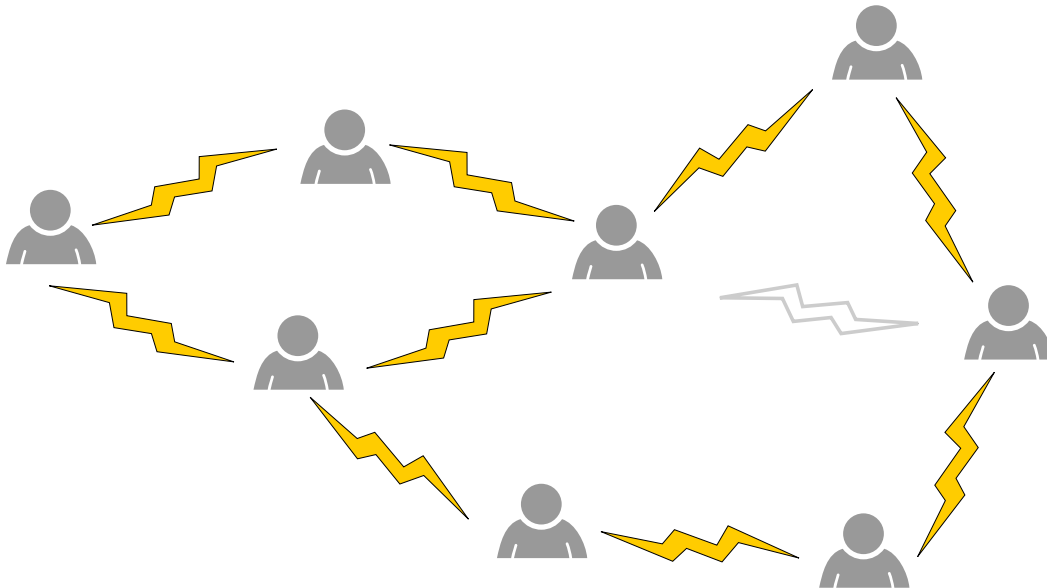
*Entferne den richtigen Blitz.*



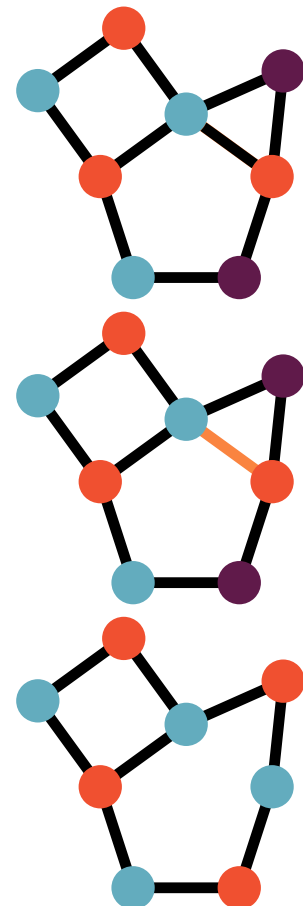


## Lösung

Die richtige Lösung ist:



Wir stellen die Situation noch etwas abstrakter als sogenannten *Graphen* dar, mit den Personen als *Knoten* (Kreise) und den Blitzen als *Kanten* (Linien).



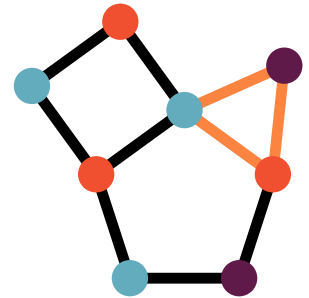
Die einzig mögliche Option ist die orange markierte Kante.

Nach dem Löschen dieser Kante, können wir die Knoten mit zwei Farben einfärben.

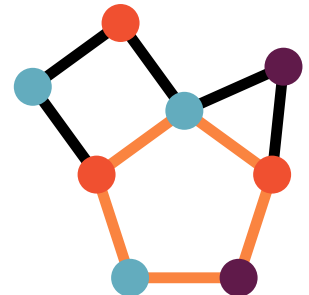
Jede Farbe steht für eine Gruppe. Wir sehen, dass nirgends zwei Personen der gleichen Gruppe die Zusammenarbeit verweigern: Benachbarte Knoten haben überall verschiedene Farben.



Um zu sehen, dass das Löschen dieser Linie die einzig mögliche Option ist, betrachten wir zunächst das orange markierte Dreieck.



Wenn eine Linie ausserhalb dieses Dreiecks gelöscht wird, benötigen wir immer noch drei Farben allein schon für die drei Knoten des Dreiecks.



Betrachten wir nun das orange markierte Fünfeck:

Wenn eine Kante ausserhalb dieses Fünfecks gelöscht wird, bleibt dieses intakt und es ist unmöglich, das Fünfeck nur mit zwei Farben zu färben: Wenn wir das Fünfeck im Uhrzeigersinn durchlaufen, müssen wir zwischen den beiden Farben abwechseln. Wenn wir jedoch den letzten Knoten erreichen, hat er dieselbe Farbe wie der erste Knoten, da die Anzahl der Kreise im Fünfeck ungerade ist, wie schon beim Dreieck.

Die einzige mögliche Lösung ist es somit, die gemeinsame Kante Dreiecks und des Fünfecks zu löschen.

## Dies ist Informatik!

Viele im Alltag auftretende Probleme können als sogenannte *Graphfärbungsprobleme* formuliert werden. In unserer Biberaufgabe repräsentieren die *Knoten* eines *Graphen* die Personen und eine *Kante* zwischen zwei Personen zeigt, dass sie sich weigern, in einer Gruppe zusammenzuarbeiten. Wenn wir die Knoten mit  $k$  Farben einfärben, kann dies als Zuordnung der Personen zu je einer von  $k$  Gruppen verstanden werden. Eine solche Färbung heisst *zulässig*, wenn für jedes Paar von zwei Knoten, die direkt durch eine Kante verbunden sind, diese beiden Knoten unterschiedliche Farben haben. In unserem Fall ist eine Färbung also genau dann zulässig, wenn in jeder Gruppe alle Personen zusammenarbeiten. Eine Kante wird als *kritisch* bezeichnet, wenn das Löschen dieser Kante eine Färbung des Graphen mit weniger Farben als zuvor ermöglicht. (Dabei dürfen alle Knoten des Graphen umgefärbt werden.) Für uns ist eine Kante also genau dann kritisch, wenn nach Überzeugung der entsprechenden beiden Personen zur Zusammenarbeit eine Aufteilung in weniger Gruppen möglich ist.



## Stichwörter und Webseiten

- Graphentheorie: <https://de.wikipedia.org/wiki/Graphentheorie>
- Färbung eines Graphen: [https://de.wikipedia.org/wiki/Färbung\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Färbung_(Graphentheorie))
- Kritischer Graph: [https://de.wikipedia.org/wiki/Kritischer\\_Graph](https://de.wikipedia.org/wiki/Kritischer_Graph)

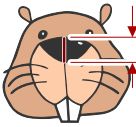
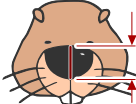





## 31. Zählen durch Nicken

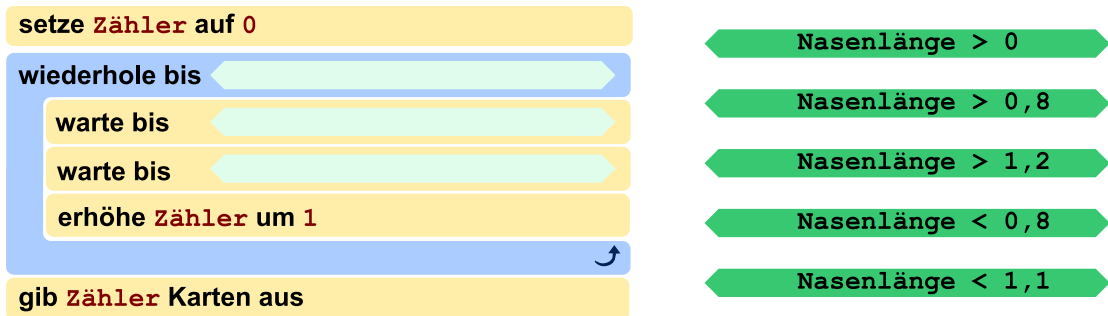
Ein neuer Eintrittskartenautomat soll so funktionieren: Ein Kunde nickt so oft mit dem Kopf – senkt also den Kopf und schaut dann wieder geradeaus – wie viele Karten er kaufen möchte. Danach hebt der Kunde den Kopf, und dann gibt der Automat die Karten aus.

Der Automat hat dazu eine Kamera eingebaut. Sie kann die Nasen der Kunden erkennen und misst ständig die Nasenlänge. Das Steuerungsprogramm des Automaten speichert das aktuelle Messergebnis unter dem Namen **Nasenlänge** und unterscheidet die Kopfhaltungen der Kunden mit Hilfe dieser Tabelle:

Kameramessung	Wert Nasenlänge	Kopfhaltung
	1	Der Kunde schaut geradeaus.
	1,3	Der Kunde hat den Kopf gesenkt.
	0,7	Der Kunde hat den Kopf gehoben.

Das Steuerungsprogramm ist fast fertig – siehe unten.

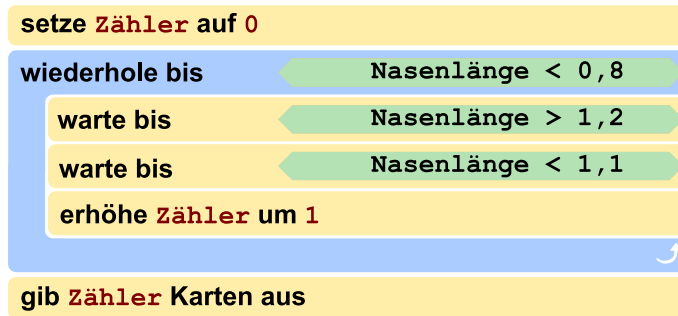
Vervollständige das Steuerungsprogramm!





## Lösung

Die einzige richtige Antwort ist:



Die Struktur des Programms ist vorgegeben: Es gibt eine zentrale Wiederholungsanweisung; oft auch *Schleife* genannt. Die letzte der in dieser Schleife wiederholten Anweisungen erhöht den Zähler der auszugebenden Karten. Folglich muss mit den zwei **warte bis**-Anweisungen ein Nicken des Kunden wahrgenommen werden: also dass der Kunde zuerst den Kopf gesenkt hat und dann wieder geradeaus schaut. Der unter *Nasenlänge* gespeicherte Wert muss also zuerst bei etwa 1,3 und dann wieder bei 1 liegen. Das entspricht den Bedingungen **Nasenlänge > 1,2** gefolgt von **Nasenlänge < 1,1**.

Anweisungen in der Schleife werden wiederholt, bis der Kunde den Kopf hebt: also ein Wert gemessen wird, der deutlich kleiner als 1 ist. Die einzige dazu passende Bedingung ist **Nasenlänge < 0,8**.

Vielleicht ist dir aufgefallen, dass das Programm nicht exakt die Werte aus der Tabelle verwendet. In der Praxis kann nämlich nicht kontinuierlich gemessen werden, sondern nur mit einer bestimmten Frequenz (beispielsweise 25 Mal pro Sekunde). Da kann es vorkommen, dass zum Beispiel der exakte Wert für das Geradeausschauen von 1,0 gar nicht gemessen wird, weil zuvor beispielsweise 0,95 und danach 1,03 gemessen wird.

## Dies ist Informatik!

*Maschinelles Sehen* (engl. *machine vision*, auch als *computer vision* bezeichnet) ist ein Teilgebiet der Informatik, in dem momentan intensiv geforscht wird. Sowohl theoretische Überlegungen als auch praktische Anwendungen sind von grosser Wichtigkeit.

Eine bedeutsame Anwendung des maschinellen Sehens ist, Behinderten eine bessere Möglichkeit zu geben, autonom mit ihrer Umwelt zu interagieren. Je nach Stärke der Behinderung kann ein Mensch beispielsweise nur noch sehr wenige Muskeln kontrollieren. Der weltbekannte Physiker Stephen Hawking (1942–2018) nutzte Bewegungen seiner Wangenmuskeln, um ein Programm zur Sprachausgabe zu steuern, nachdem er die Kontrolle über seine weitere Muskulatur grösstenteils verloren hatte.

Das konkrete Beispiel jedoch könnte auch für Musiker eingesetzt werden: sie nutzen in der Regel beide Hände, um ihr Instrument zu bedienen. Handelsübliche Geräte bieten hierfür ein Fusspedal.



Bestimmte Musiker wie Organisten nutzen jedoch auch die Füße zum Spielen und könnten durch ein einfaches Nicken beispielsweise Noten automatisch umblättern.

Im Gegensatz zum Beispiel dieser Aufgabe, wo die Werte konkret und fest einprogrammiert sind, wird maschinelles Sehen oftmals mit *maschinellern Lernen* (engl. *machine learning*) kombiniert. Dann wird das Programm auf bestimmte Gesten trainiert, indem ihm für jede Geste viele Beispiele und Gegenbeispiele gezeigt werden. So baut die Software eine statistische Einschätzung auf, was wie zu interpretieren ist.

## Stichwörter und Webseiten

- Maschinelles Sehen, Machine Vision, Computer Vision:  
[https://de.wikipedia.org/wiki/Computer\\_Vision](https://de.wikipedia.org/wiki/Computer_Vision)
- Maschinelles Lernen, Machine Learning:  
[https://en.wikipedia.org/wiki/Machine\\_learning](https://en.wikipedia.org/wiki/Machine_learning)
- [https://en.wikipedia.org/wiki/Stephen\\_Hawking#Disability](https://en.wikipedia.org/wiki/Stephen_Hawking#Disability)
- <https://de.wikipedia.org/wiki/Notenwender>



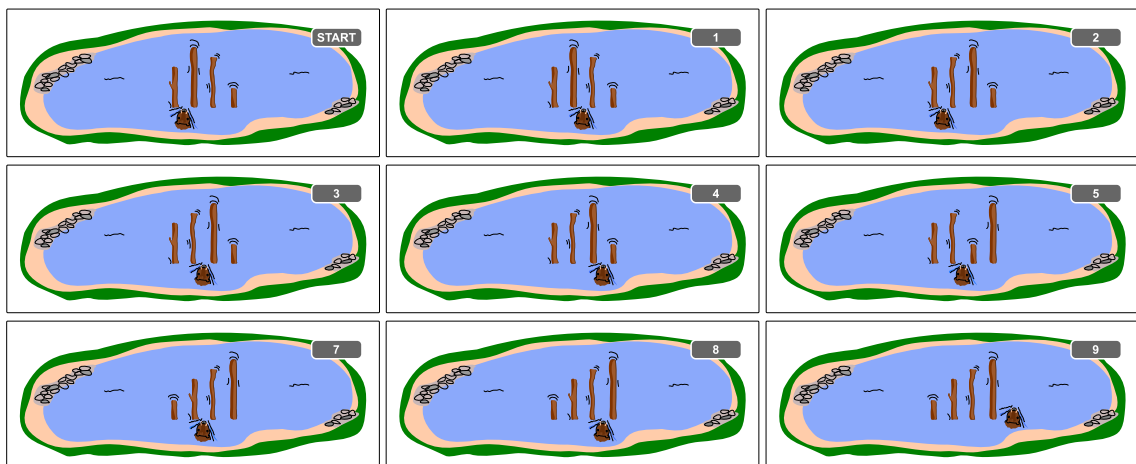


## 32. Beaver Sort

Biber Hamid sortiert die Baumstämme im See. Von links nach rechts sollen sie immer länger werden.

- Hamids Startposition ist zwischen den beiden Stämmen ganz links.
- Wenn er zwischen zwei benachbarten Stämmen ist, vergleicht er diese:
  - Wenn der rechte Stamm länger ist als der linke, schwimmt er eins nach rechts.
  - Wenn hingegen der linke Stamm länger ist, dann vertauscht er die beiden Stämme. Nach dem Vertauschen schwimmt er eins nach rechts, wenn er in der Startposition ist, und sonst eins nach links.
- So macht Hamid weiter, bis er rechts von allen Stämmen angekommen ist. Dann weiss er, dass alle Stämme korrekt sortiert sind.

Das Beispiel zeigt, wie Hamid 4 Baumstämme sortiert. Er macht dabei insgesamt 9 Vergleiche.



Die Anzahl der Vergleiche hängt davon ab, wie die Stämme am Anfang liegen. Für 4 Stämme muss Hamid mindestens 3 Vergleiche machen (wenn die Stämme bereits richtig sortiert sind) und höchstens 9 Vergleiche (wenn die Stämme alle genau verkehrt sortiert sind). Bei 4 Baumstämmen muss Hamid also mit 3 bis 9 Vergleichen rechnen.

*Hamid muss nun 40 verschieden lange Baumstämme sortieren. Mit wie vielen Vergleichen muss er rechnen?*

- A) Mit 0 bis 20 Vergleichen
- B) Mit 3 bis 40 Vergleichen
- C) Mit 39 bis 120 Vergleichen
- D) Mit 39 bis 1521 Vergleichen



## Lösung

Für 40 Stämme benötigt Hamid im besten Fall bereits 39 Vergleiche, nämlich dann, wenn alle Baumstämme bereits sortiert sind und er somit immer nach rechts schwimmt, bis er fertig ist. Damit können die Antworten A) und B) nicht richtig sein.

Im ungünstigsten Fall sind die Baumstämme am Anfang genau verkehrt herum sortiert. Hier ist die Anzahl der Vergleiche schwieriger zu bestimmen. Wir beobachten zuerst Folgendes: Die Baumstämme links von Hamid sind unter sich immer richtig sortiert. Wenn Hamid zu einem Baumstamm kommt, der kürzer ist als die  $k$  links von ihm, dann tauscht er ihn zuerst ganz nach links durch, wobei er  $k$  Vergleiche macht. Dann kommt er zurück und schwimmt sogar noch eins weiter nach rechts, wobei er  $k - 1$  weitere Vergleiche macht, aber nichts mehr vertauscht. Insgesamt sind das  $k + (k - 1) = 2k - 1$  Vergleiche. Dies passiert bei jeder der  $n - 1$  Positionen zwischen zwei Baumstämmen und die Anzahl Baumstämme links von Hamid ist dabei im Durchschnitt  $\frac{n}{2}$ . Somit macht er insgesamt  $(n - 1) \cdot (2 \cdot \frac{n}{2} - 1) = (n - 1)^2$ , also  $39^2 = 1521$  Vergleiche. Also ist Antwort C) mit höchstens 120 Vergleichen falsch und Antwort D) ist korrekt.

## Dies ist Informatik!

Das Sortieren von Elementen ist eine klassische Aufgabenstellung der Informatik. Effizientes Sortieren spart dabei viel Zeit. Der hier präsentierte Sortieralgorithmus heisst *Gnome Sort* und wurde ursprünglich von dem iranischen Informatiker Hamid Sarbazi-Azad als *Stupid Sort* vorgestellt. Später hat der niederländische Informatiker Dick Grune dieses Sortierverfahren als Gnome Sort bezeichnet, mit der Vorstellung, dass ein Gartenzweig (engl. *garden gnome*) so Blumentöpfe sortieren könnte.

Die Analyse der Laufzeit von Algorithmen (also das Herausfinden davon, wie lange ein Algorithmus brauchen kann, bis er fertig ist) ist in der Informatik sehr wichtig. Oft fragt man, was die Laufzeit im besten Fall, im durchschnittlichen Fall und im schlechtesten Fall ist, und zwar in Abhängigkeit von der Eingabegrösse  $n$ . (In unserem Fall nehmen wir dafür die Anzahl Baumstämme.) Um den Vergleich zwischen verschiedenen Laufzeiten möglichst einfach zu halten, begnügt man sich meistens damit, sie nur grob anzugeben. Gnome Sort braucht im besten Fall nur lineare Laufzeit - man schreibt dafür  $\mathcal{O}(n)$  -, im durchschnittlichen Fall quadratische Laufzeit - man schreibt dafür  $\mathcal{O}(n^2)$  - und im schlimmsten Fall ebenfalls quadratische Laufzeit - man schreibt also wieder  $\mathcal{O}(n^2)$ .

## Stichwörter und Webseiten

- Gnome Sort: <https://de.wikipedia.org/wiki/Gnomesort>
- Sortierverfahren: <https://de.wikipedia.org/wiki/Sortierverfahren>
- Laufzeitanalyse
- O-Notation, Landau-Symbole: <https://de.wikipedia.org/wiki/Landau-Symbole>

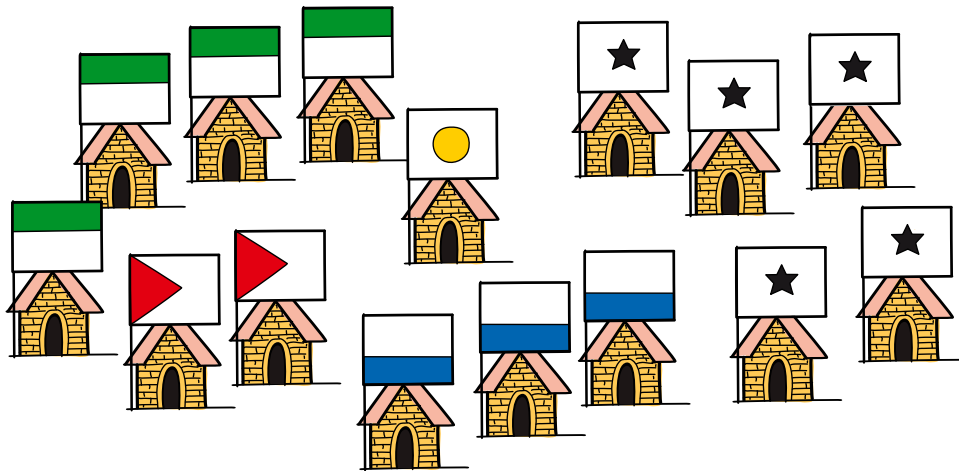


## 33. Die Clans von Beavaria

In Beavaria leben fünf einst verfeindete Clans mit jeweils einigen Häusern, wie man im Bild erkennen kann: Mac Intoshs, Apfeler, Mac Rosofts, Androidier und Freidosier. Da die Zeiten lange schon friedlich sind, beschliessen sie, das Vereinigungsritual durchzuführen. Die Regeln dafür sind:

- Es dürfen sich immer nur zwei Clans zur selben Zeit vereinigen.
- In jedem Haus der sich vereinigenden Clans wird eine Woche lang gefeiert, um den Pakt zu besiegeln. Die Dauer des Vereinigens in Wochen ist daher gleich der Anzahl der Häuser in den beiden Clans.
- Nach dieser Zeit sind die beiden Clans nur noch ein Clan. Nun kann das Vereinigen der Clans fortgesetzt werden.

Die Clans beschliessen, die Vereinigung in möglichst kurzer Zeit durchzuführen. Das geht nur, indem man die Reihenfolge der Vereinigungen sorgfältig plant.



*Wie viele Wochen dauert es mindestens bis alle Clans vereinigt sind?*

- A) 15 Wochen
- B) 33 Wochen
- C) 35 Wochen
- D) 50 Wochen
- E) 120 Wochen

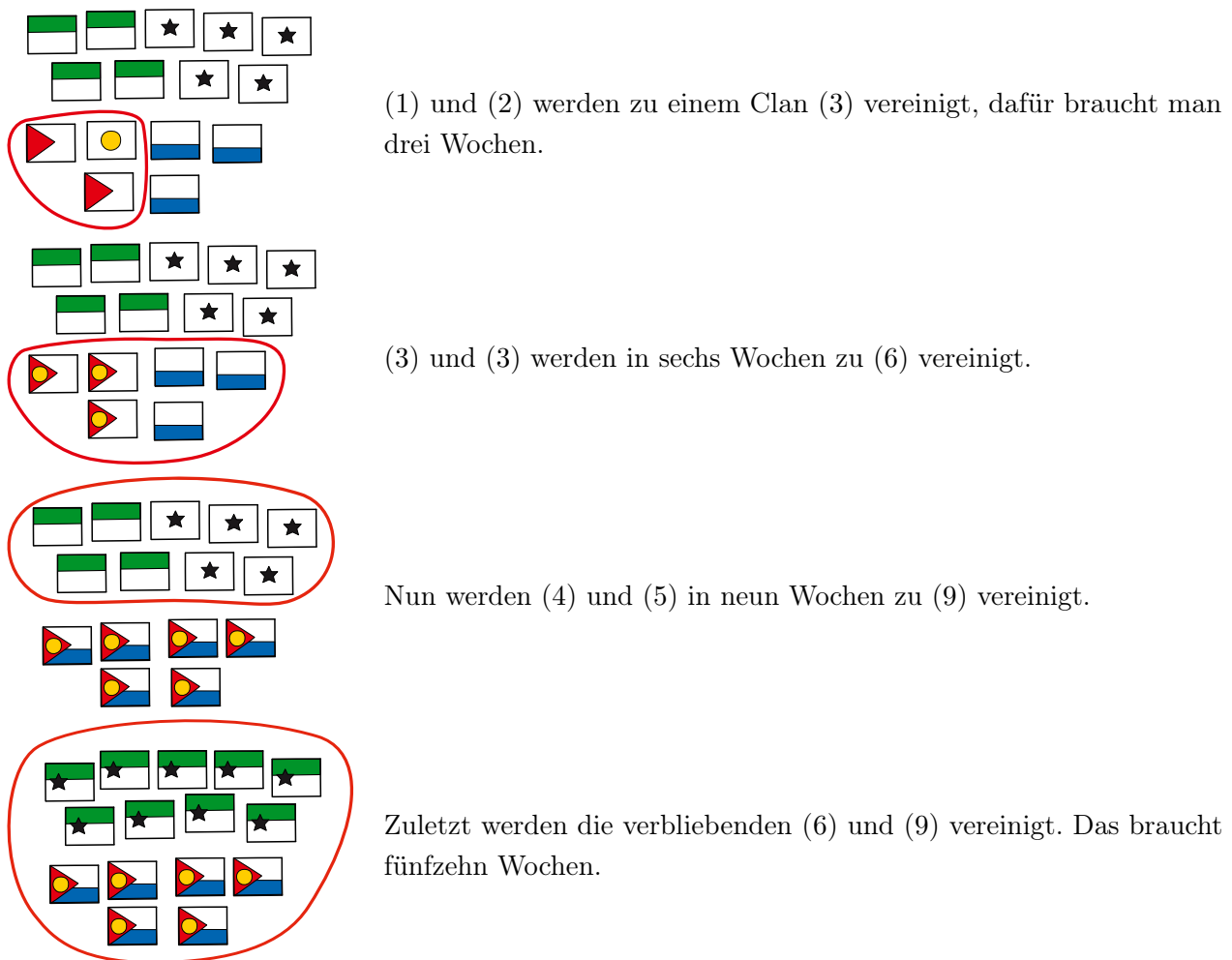


## Lösung

Die richtige Antwort ist B) 33 Wochen.

Die optimale Strategie zur Minimierung der Gesamtzahl der Wochen, die für die Vereinigung aller Clans benötigt werden, besteht darin, die Anzahl der jeweils beteiligten Häuser zu minimieren. Die Häuser der zuerst vereinigten Clans werden dann ja auch bei den weiteren Vereinigungen berücksichtigt. Daher ist es sinnvoll, zunächst jeweils die kleinsten Clans zu vereinigen, so dass die grossen nicht zu oft an der Zeremonie teilnehmen müssen. Um dies zu erreichen, sollten in jedem Schritt die beiden Clans mit den wenigsten Häusern vereinigt werden.

Dieser schrittweise Prozess ist in folgender Tabelle dargestellt, die Clannamen lassen wir der Übersicht halber weg und nennen nur in Klammern die Clangrössen:



Insgesamt ist der Prozess also nach  $15 + 9 + 6 + 3 = 33$  Wochen abgeschlossen, das ist Antwort B).





## Dies ist Informatik!

Dies ist ein *Optimierungsproblem* — eine Aufgabe, einen Wert (in diesem Fall die Dauer der Vereinigung) zu minimieren oder zu maximieren. Hierfür werden oft *Algorithmen* verwendet, die zum Beispiel für ressourcenschonende, umweltfreundliche Produktionsprozesse heute nicht mehr wegzudenken sind.

In diesem Fall kann das Problem sogar mit einem einfachen *Greedy-Algorithmus* optimal gelöst werden. Hierbei tut man in jedem Schritt das, was nach einer einfachen Idee gerade am besten zu sein scheint. In unserem Fall ist die Idee, dass wir zuerst kleine Clans vereinigen, weil dies am wenigsten Zeit kostet.

## Stichwörter und Webseiten

- Optimierungsproblem - <https://de.wikipedia.org/wiki/Optimierungsproblem>
- Greedy-Algorithmus - <https://de.wikipedia.org/wiki/Greedy-Algorithmus>




# A. Aufgabenautoren

- |                              |                                   |
|------------------------------|-----------------------------------|
| Daumilas Ardickas            | Ezgi Arzu Güneş                   |
| Sarah Atkins                 | Mathias Hiron                     |
| Michael Barot                | Benjamin Hirsch                   |
| Liam Baumann                 | Juraj Hromkovič                   |
| Wilfried Baumann             | Andrea Hrušecká                   |
| Linda Björk Bergsveinsdóttir | Alisher Ikramov                   |
| Javier Bilbao                | Tiberiu Iorgulescu                |
| Lucia Budinská               | Svetlana Jaksic                   |
| Carmel Carroll               | YongJu Jeon                       |
| Sarah Chan                   | Soojin Jun                        |
| Anton Chukhnov               | Ungyeol Jung                      |
| Kris Coolsaet                | Filiz Kalelioğlu                  |
| Valentina Dagiéné            | Martin Kandlhofer                 |
| Christian Datzko             | Ulrich Kiesmüller                 |
| Susanne Datzko               | Dong Yoon Kim                     |
| Janez Demšar                 | Jihye Kim                         |
| Nora A. Escherle             | Vaidotas Kinčius                  |
| Lidia Feklistova             | Víctor Koleszar                   |
| Margarita Flores-Sicich      | Regula Lacher                     |
| Fabian Frei                  | Taina Lehtimäki                   |
| Gerald Futschek              | Marielle Léonard                  |
| Jens Gallenbacher            | Angélica Herrera Loyo             |
| Thomas Galler                | Tom Naughton                      |
| Mark Edward M. Gonzales      | Mochammad Irfan Noviana           |
| Martin Guggisberg            | Graciela Oyhenard                 |
| Yasemin Gülbahar             | Gabriela Lourdes Rodríguez Parada |



 Jean-Philippe Pellet

 Hannah Piper

 Jonatan Pipping


 Zsuzsa Pluhár

 Wolfgang Pohl


 Peter Rossmann

 Rodrigo Santamaría

 Eljakim Schrijvers

 Rosario Schunk

 Tomas Šiaulys

 Timur Sitdikov

 Bernadette Spieler

 Cuttle.org Team

 Ezra Templonuevo

 Ahto Truu

 Troy Vasiga


 Florentina Voboril

 Eslam Wageed

 Michael Weigend

 Kyra Willekes

 Hongjin Yeh

 Mija Zaļūksne



## B. Sponsoring: Wettbewerb 2021

### HASLERSTIFTUNG

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

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



<http://www.baerli-biber.ch/>

Schon in der vierten Generation stellt die Familie Bischofberger ihre Appenzeller Köstlichkeiten her. Und die Devise der Bischofbergers ist dabei stets dieselbe geblieben: «Hausgemacht schmeckt's am besten». Es werden nur hochwertige Rohstoffe verwendet: reiner Bienenhonig und Mandeln allererster Güte. Darum ist der Informatik-Biber ein «echtes Biberli».



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



Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich



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://www.ubs.com/>

Wealth Management IT and UBS Switzerland IT



**OXOCARD**

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

OXOcard: Spielend programmieren lernen  
OXON

**educaTEC**

<https://educatec.ch/>

educaTEC

Wir sind MINT-Experten. Seit unserer Gründung 2004 verfolgen wir das Ziel, Technik und ingenieurwissenschaftliches Denken in öffentlichen und privaten Schulen der Schweiz zu fördern. In Kombination mit kompetenter Beratung und Unterstützung offerieren wir Lehrkräften innovative Lehrmaterialien von weltweit führenden Herstellern sowie Lernkonzepte für den MINT-Bereich und verwandte Fächer.

**senarclens**  
**leu+partner**  
strategische kommunikation

<http://senarclens.com/>

Senarclens Leu & 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

**PH LUZERN**  
**PÄDAGOGISCHE**  
**HOCHSCHULE**

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

Pädagogische Hochschule Luzern

**n|w** Fachhochschule  
Nordwestschweiz

<https://www.fhnw.ch/de/die-fhnw/hochschulen/ph>

Pädagogische Hochschule FHNW

Scuola universitaria professionale  
della Svizzera italiana

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

La Scuola universitaria professionale della Svizzera italiana  
(SUPSI)

**SUPSI**



PÄDAGOGISCHE  
HOCHSCHULE  
ZÜRICH



<https://www.phzh.ch/>  
Pädagogische Hochschule Zürich



## C. Weiterführende Angebote

### 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.



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//sociétésuissepourl'infor  
matique dans l'enseignement//societàsviz  
zera per l'informatica nell'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.