BUNDES WEIT INFORMATIK NACHWUCHS FÖRDERN

www.bwinf.de

Informatik-Biber

AUFGABEN 2018

Der Wettbewerb zum digitalen Denken.

bwinf.de/biber

Herausgeber: Wolfgang Pohl, BWINF

Der Aufgabenausschuss Informatik-Biber 2018

Hans-Werner Hein, BWINF Bonn
Ulrich Kiesmüller, Simon-Marius-Gymnasium Gunzenhausen
Wolfgang Pohl, BWINF Bonn
Kirsten Schlüter, Bayerisches Staatsministerium für Unterricht und Kultus
Michael Weigend, Holzkamp-Gesamtschule Witten

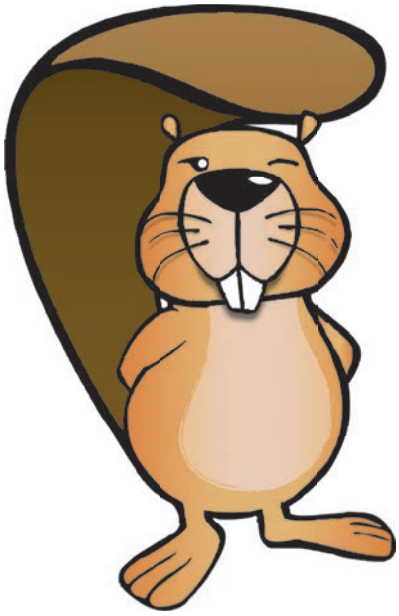
Die deutschsprachigen Fassungen der Aufgaben wurden auch in Österreich und der Schweiz verwendet. An ihrer Erstellung haben mitgewirkt:

Andrea Adamoli, Università della Svizzera italiana
Daniel Brüning, BWINF Bonn
Wilfried Baumann, Österreichische Computer Gesellschaft
Robert Czechowski, BWINF Bonn
Christian Datzko, Wirtschaftsgymnasium und Wirtschaftsmittelschule Basel
Susanne Datzko, freischaffende Graphikerin, ETH Zürich
Olivier Ens, Freis Schulen, Schweiz. Verein für Informatik in der Ausbildung (SVIA)
Hanspeter Erni, Pädagogische Hochschule Luzern, SVIA
Gerald Futschek, Technische Universität Wien
Martin Guggisberg, Pädagogische Hochschule FHNW, SVIA
Urs Hauser, ETH Zürich / Pädagogische Hochschule Luzern, SVIA
Juraj Hromkovic, ETH Zürich, SVIA
Ivana Kosirová, ETH Zürich, SVIA
Regula Lacher, ETH Zürich, SVIA
Jean-Philippe Pellet, Haute École Pédagogique Vaud, SVIA
Katharina Resch-Schobel, Österreichische Computer Gesellschaft
Martin Stangl, Student Pädagogische Hochschule FHNW

Der Informatik-Biber

ist ein Projekt der Bundesweiten Informatikwettbewerbe (BWINF).
BWINF ist eine Initiative der Gesellschaft für Informatik (GI),
des Fraunhofer-Verbunds IUK-Technologie und
des Max-Planck-Instituts für Informatik.
BWINF wird vom Bundesministerium für Bildung und Forschung (BMBF)
gefördert. Die Bundesweiten Informatikwettbewerbe gehören zu den
von den Kultusministerien empfohlenen Schülerwettbewerben und stehen
unter der Schirmherrschaft des Bundespräsidenten.

Einleitung



Der Informatik-Biber ist ein Online-Test mit Aufgaben zur Informatik. Er erfordert Köpfchen, aber keine Vorkenntnisse.

Der Informatik-Biber will das allgemeine Interesse für das Fach Informatik wecken und gleichzeitig die Motivation für eine Teilnahme an Informatikwettbewerben stärken. Schülerinnen und Schüler, die mehr wollen, sind herzlich eingeladen, sich anschließend am Jugendwettbewerb Informatik und auch am Bundeswettbewerb Informatik zu versuchen (siehe Seite 5).

Der Informatik-Biber findet jährlich im November statt. An der 12. Austragung im Jahr 2018 beteiligten sich 2.101 Schulen und andere Bildungseinrichtungen mit 373.406 Schülerinnen und Schülern. Die Möglichkeit, auch in Zweierteams zu arbeiten, wurde gern genutzt.

Die Online-Teilnahme am Informatik-Biber 2018 war mit Desktops, Laptops und Tablets möglich. Weniger als die Hälfte der Antworteingaben waren multiple-choice. Verschiedene andere Interaktionsformen machten die Bearbeitung abwechslungsreich. In diesem Biberheft ist die Dynamik der Aufgabenbearbeitung nicht vorführbar. Darum geben Handlungstipps in den Aufgabenstellungen und Bilder von Lösungssituationen davon eine Vorstellung. Der Umgang mit dem Wettbewerbssystem selbst konnte in den Wochen vor der Austragung online geübt werden.

Der Informatik-Biber 2018 wurde in fünf Altersgruppen durchgeführt. In den Klassenstufen 3 bis 4 waren innerhalb von 30 Minuten 9 Aufgaben zu lösen, jeweils drei in den Schwierigkeitsstufen leicht, mittel und schwer. In den Klassenstufen 5 bis 6 waren innerhalb von 35 Minuten 12 Aufgaben zu lösen, jeweils vier in den Schwierigkeitsstufen leicht, mittel und schwer. In den Klassenstufen 7 bis 8, 9 bis 10 und 11 bis 13 waren innerhalb von 40 Minuten 15 Aufgaben zu lösen, jeweils fünf in den Schwierigkeitsstufen leicht, mittel und schwer.

Die 38 Aufgaben des Informatik-Biber 2018 sind auf Seite 6 gelistet, nach ungefähr steigender Schwierigkeit und mit einer informatischen Klassifikation ihres Aufgabenthemas. Ab Seite 7 folgen die Aufgaben nach ihrem Titel alphabetisch sortiert. Im Kopf sind die zugeordneten Altersgruppen und Schwierigkeitsgrade vermerkt. Eine kleine Flagge gibt an, aus welchem Bebras-Land die Idee zu dieser Aufgabe stammt. Der Kasten am Aufgabenende enthält Erläuterungen zu den Lösungen und Lösungswegen sowie eine kurze Darstellung des Aufgabenthemas hinsichtlich seiner Relevanz in der Informatik.

Die Veranstalter bedanken sich bei allen Lehrkräften, die mit großem Engagement ihren Klassen und Kursen ermöglicht haben, den Informatik-Biber zu erleben.

Wir laden die Schülerinnen und Schüler ein, auch 2019 wieder beim Informatik-Biber mitzumachen, und zwar in der Zeit vom 4. bis 15. November. Weitere Informationen werden über die Website bwinf.de und per E-Mail an die Koordinatorinnen und Koordinatoren bekannt gegeben.

Bebras: International Challenge on Informatics and Computational Thinking

Der zypriotische Biber

Der deutsche Informatik-Biber ist Partner der internationalen Initiative Bebras. 2004 fand in Litauen der erste Bebras Challenge statt. 2006 traten Estland, die Niederlande und Polen der Initiative bei, und auch Deutschland veranstaltete im Jahr der Informatik als „EI: Spiel blitz!“ einen ersten Biber-Testlauf. Seitdem kamen viele Bebras-Länder hinzu. Zum Drucktermin sind es weltweit 66, und weitere Länderteilnahmen sind in Planung. Insgesamt hatte der Bebras Challenge 2018 international annähernd drei Millionen Teilnehmerinnen und Teilnehmer.

Die Bebras-Community erarbeitet jedes Jahr auf einem internationalen Workshop anhand von Vorschlägen der Länder eine größere Auswahl möglicher Aufgabenideen. Die Ideen zu den 38 Aufgaben des Informatik-Biber 2018 stammen aus 18 Ländern: Belgien, China, Deutschland, Irland, Italien, Kanada, Kroatien, Litauen, Pakistan, Schweiz, Slowakei, Taiwan, Tschechien, Türkei, Ungarn, USA, Vereinigtes Königreich und Vietnam.

Der lettische Biber

Der indonesische Biber

Deutschland nutzt zusammen mit einer Vielzahl anderer Länder zur Durchführung des Informatik-Biber ein gemeinsames Online-System.
Dieses „International Bebras Challenge System“ wird von der niederländischen Firma Eljakim IT betrieben und fortentwickelt.

Informationen über die Aktivitäten aller Bebras-Länder finden sich auf der Website bebras.org.

Bundesweite Informatikwettbewerbe

Bundesweite Informatikwettbewerbe

Informatik-Biber

Jugendwettbewerb Informatik

Bundeswettbewerb Informatik

Informatik-Olympiade

Bei jungen Menschen das Interesse für Informatik wecken, Begabungen entdecken und fördern: das ist das Ziel der Bundesweiten Informatikwettbewerbe (BWINF), an denen im Jahr 2018 über 390.000 junge Menschen teilnahmen. Der Informatik-Biber ist das BWINF-Einstiegsformat; außerdem werden noch drei weitere Wettbewerbe angeboten:

Jugendwettbewerb Informatik
Der Jugendwettbewerb Informatik (JwInf) wurde 2017 zum ersten Mal ausgerichtet. Er richtet sich an Kinder und Jugendliche, die erste Programmiererfahrungen sammeln und vertiefen möchten. Er ist in den ersten Runden ein reiner Online-Wettbewerb, genauso wie der Informatik-Biber. Empfohlen wird eine Teilnahme ab der Jahrgangsstufe 5; die dafür nötigen Kenntnisse können auf der JwInf-Plattform erworben werden (wettbewerb.jwinf.de).

Bundeswettbewerb Informatik
Der Bundeswettbewerb Informatik (BwInf) wurde 1980 von der Gesellschaft für Informatik e.V. (GI) auf Initiative von Prof. Dr. Volker Claus ins Leben gerufen. Dieser traditionsreichste BWINF-Wettbewerb beginnt jedes Jahr im September. Die Aufgaben der ersten und zweiten Runde werden zu Hause selbstständig bearbeitet, einzeln oder in einer Gruppe. In der zweiten Runde ist dann eigenständiges Arbeiten gefordert. Die ca. dreißig bundesweit Besten werden zur dritten Runde, einem Kolloquium, eingeladen. Allen Teilnehmenden stehen weitergehende Fördermaßnahmen offen. Die Siegerinnen und Sieger werden ohne weiteres Verfahren in die Studienstiftung des deutschen Volkes aufgenommen.

Internationale Informatik-Olympiade
Die Jüngeren unter den BwInf-Finalisten und einige ausgewählte Teilnehmende der zweiten Runde können sich in mehreren Trainingsrunden sowie bei Vorbereitungswettbewerben im europäischen Ausland für das vierköpfige deutsche Team qualifizieren, das im Folgejahr an der Internationalen Informatik-Olympiade (IOI) teilnimmt.

Austausch
Die Teilnahme an BWINF-Wettbewerben eröffnet Möglichkeiten zum Austausch mit Gleichgesinnten. Erste Anknüpfungspunkte bieten „BWINF – Informatik erleben" bei Facebook, die BWINF-Accounts bei Twitter und Instagram, das Informatik-Jugendportal Einstieg Informatik mit seiner Community und die BWINF-Website. Die mehr als 35 Jahrgänge von BwInf-Teilnehmenden bilden ein wachsendes Netzwerk, vor allem im BwInf Alumni und Freunde e.V. Nach der ersten BwInf-Runde lernen sich viele Teilnehmende bei Informatik-Workshops von Hochschulen und Unternehmen kennen.

Träger und Förderer
BWINF ist eine Initiative der Gesellschaft für Informatik (GI), des Fraunhofer-Verbunds IUK-Technologie und des Max-Planck-Instituts für Informatik. BWINF wird vom Bundesministerium für Bildung und Forschung (BMBF) gefördert. Die Bundesweiten Informatikwettbewerbe gehören zu den von der Kultusministerkonferenz empfohlenen Schülerwettbewerben und stehen unter der Schirmherrschaft des Bundespräsidenten.

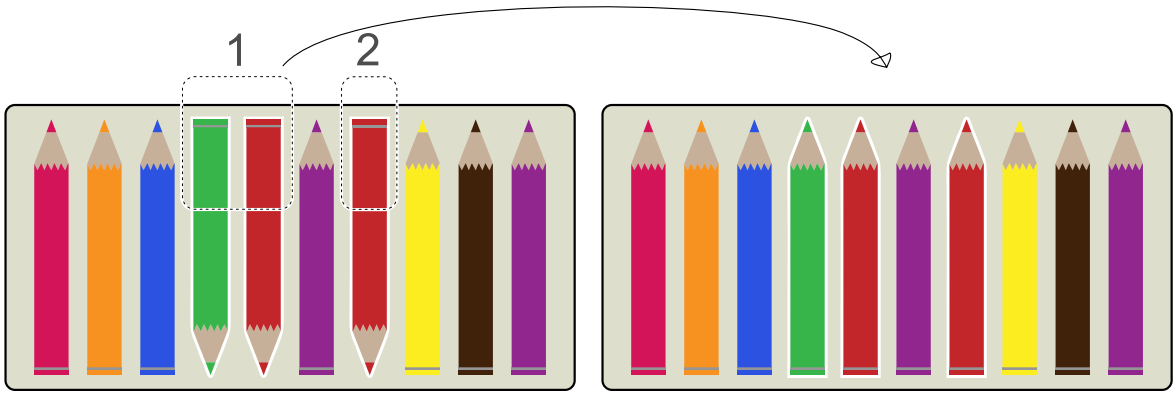
Aufgabenliste

Dies sind die 38 Aufgaben des Informatik-Biber 2018, geordnet nach ungefähr steigender Schwierigkeit und gelistet mit einer Klassifikation ihres informatischen Inhalts.

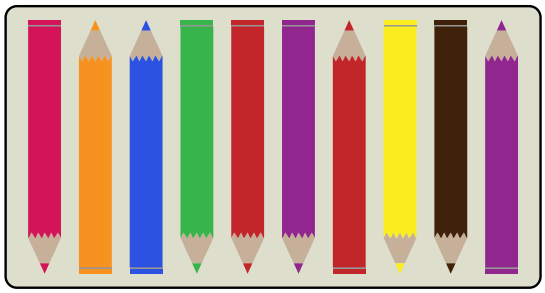
Adas Stifte

Ada hat eine Schachtel mit 10 Stiften. Einige zeigen nach oben, einige zeigen nach unten. Ada möchte, dass alle Stifte nach oben zeigen.

Ada kann Stifte, die nebeneinander liegen, auf einmal umdrehen.
Wenn die Stifte so liegen, muss sie nur zweimal Stifte umdrehen:

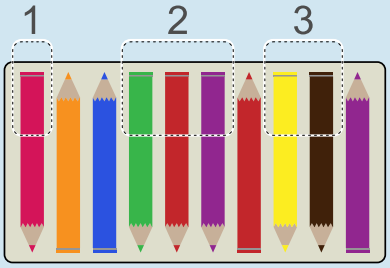


Nun liegen die Stifte so:



Wie oft muss Ada mindestens Stifte umdrehen, damit alle Stifte nach oben zeigen?

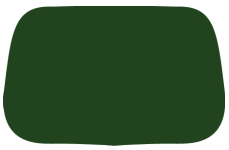
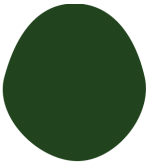
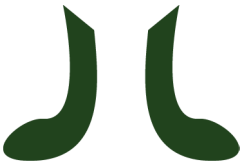
3 ist die richtige Antwort.
Ada muss dreimal Stifte umdrehen. Dreimal zeigen Stifte nebeneinander nach unten:



Das ist Informatik!
Ada hätte wahrscheinlich genug Zeit, jeden Stift, der nicht nach oben zeigt, einzeln umzudrehen. Für sie ist es nicht so wichtig, die Stifte in „Blöcken“ umzudrehen, wie sie das in dieser Biberaufgabe macht. Computer haben es da nicht so einfach. Sie haben es nicht mit einigen wenigen Stiften zu tun, sondern müssen meist sehr viele Daten bearbeiten. Da ist es gut, wenn sie möglichst viel auf einmal tun können und deshalb mit ihrer Arbeit schnell fertig werden.
Schnelligkeit ist zum Beispiel beim Lesen und Schreiben von Daten auf Datenspeicher erwünscht. Solche Speicher sind deshalb in Blöcken organisiert. Der Computer kann Daten in einem Block auf einen Schlag lesen oder auch schreiben – so wie Ada Blöcke von Stiften, die nach unten zeigen, auf einen Schlag umdrehen kann. Das spart Zeit.
Übrigens: Je größer die Blöcke sind, desto mehr Zeit wird gespart – aber auch mehr Platz verschwendet. Stell dir vor, Ada legt ihre 10 Stifte in 4er-Schachteln. Dann braucht sie drei Schachteln und lässt Platz für zwei Stifte leer. Benutzt sie 8er-Schachteln, braucht sie zwar nur zwei Schachteln, lässt aber Platz für sechs Stifte leer. Das ist oft so in der Informatik: Man kann entweder Zeit oder Platz sparen, aber nicht beides gleichzeitig.

Aliens basteln

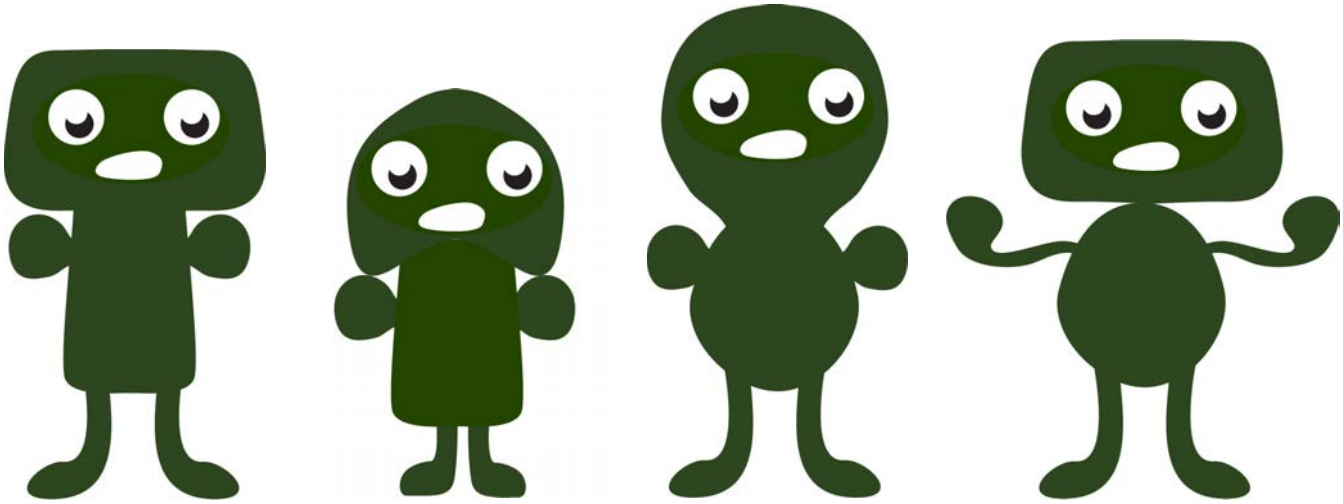
In einem Computerspiel kann man sich Aliens basteln.
Mit diesen Befehlen erstellt man die einzelnen Körperteile:

Kopf	**K(r)** runder Kopf 	**K(3)** eckiger Kopf 	**K(4)** eckiger Kopf 
Rumpf	**R(r)** runder Rumpf 	**R(3)** eckiger Rumpf 	**R(4)** eckiger Rumpf 
Arme	**A(+)** lange Arme 	**A(–)** kurze Arme 	
Beine	**B(+)** lange Beine 	**B(–)** kurze Beine 	

Werden mehrere Befehle für einen Körperteil verwendet, gilt der letzte.

Ein Beispiel: Mit den Befehlen
K(r), R(4), K(4), A(–), B(–)
bastelt man sich dieses Alien:

Welches Alien bastelt man sich mit diesen Befehlen:
K(3), B(+), R(3), A(+), K(r), A(–), R(r)?



A) **B)** **C)** **D)**

Antwort C ist richtig:
Für jeden Körperteil gilt nur der letzte Befehl. Von den abgegebenen Befehlen haben also nur die fett gedruckten eine Wirkung: K(3), **B(+),** R(3), A(+), **K(r), A(–), R(r)**

Mit diesen Befehlen bastelt man sich also ein Alien mit einem runden Kopf, einem runden Rumpf, kurzen Armen und langen Beinen. Das ist genau das Alien von Antwort C. Bei den Aliens der anderen Antworten sind jeweils mindestens zwei Körperteile anders:

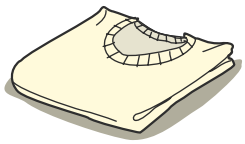
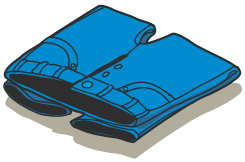
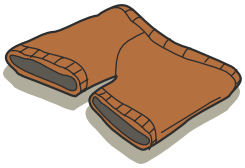
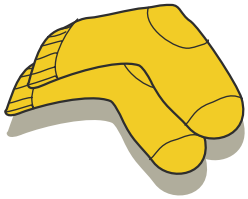
Antwort A: Dieses Alien hat einen 4-eckigen Kopf und einen 4-eckigen Rumpf.
Antwort B: Dieses Alien hat einen 3-eckigen Kopf, einen 4-eckigen Rumpf und kurze Beine.
Antwort D: Dieses Alien hat einen 4-eckigen Kopf und lange Arme.

Das ist Informatik!
Auch unter den Anweisungen im Quellcode eines Computerprogramms kann es überflüssige Anweisungen geben, die keine Auswirkungen auf das Endergebnis des Programms haben. Das können Anweisungen sein, deren Wirkung – wie in dieser Biberaufgabe – von einer späteren Anweisung komplett überschrieben wird. Das können aber auch Anweisungen sein, deren Ergebnis an keiner Stelle im Programm verwendet wird. Ein Compiler, der den Quellcode eines Programms in die vom Computer ausführbaren Befehle übersetzt, wird versuchen, solch „toten Code“ zu ignorieren. Damit wird das übersetzte Programm kleiner, und der Computer muss keine unnötigen Befehle ausführen.

https://de.wikipedia.org/wiki/Toter_Code

Anzieh-Stapel

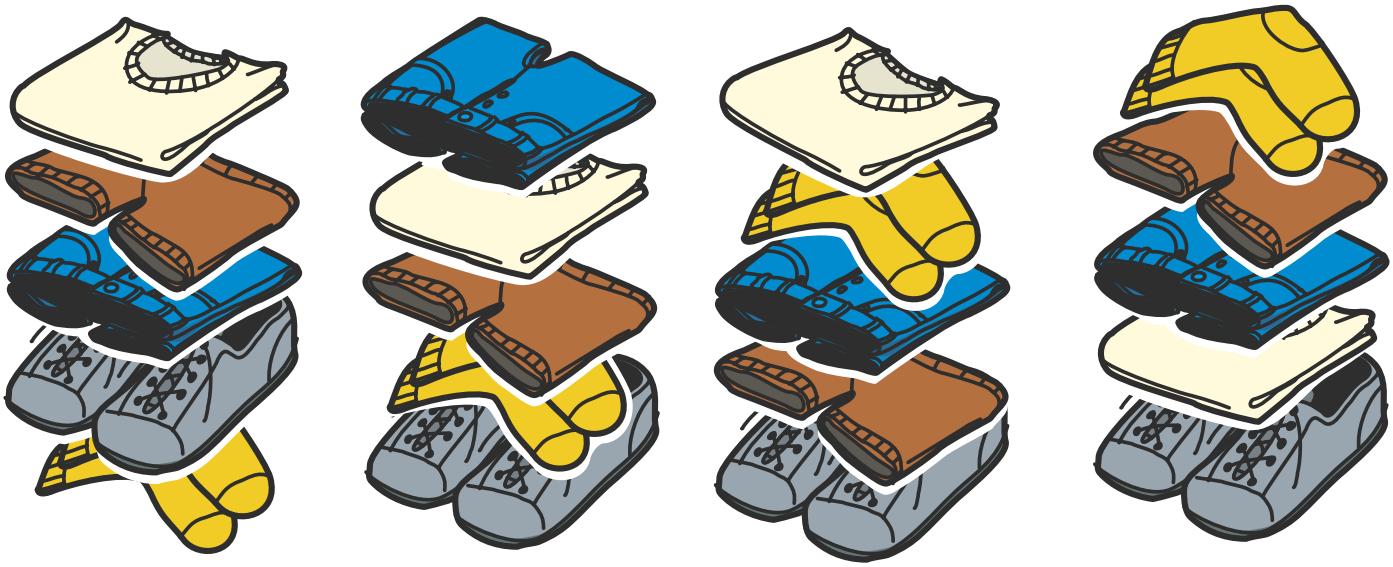
Das sind Brunos Anziehsachen:

Hemd	Hose	Unterhose	Socken	Schuhe
				

Abends legt Bruno seine Sachen auf einen Stapel.
Dann geht es morgens schnell:
Er zieht zuerst die oberste Sache an, dann die nächste – schön der Reihe nach.

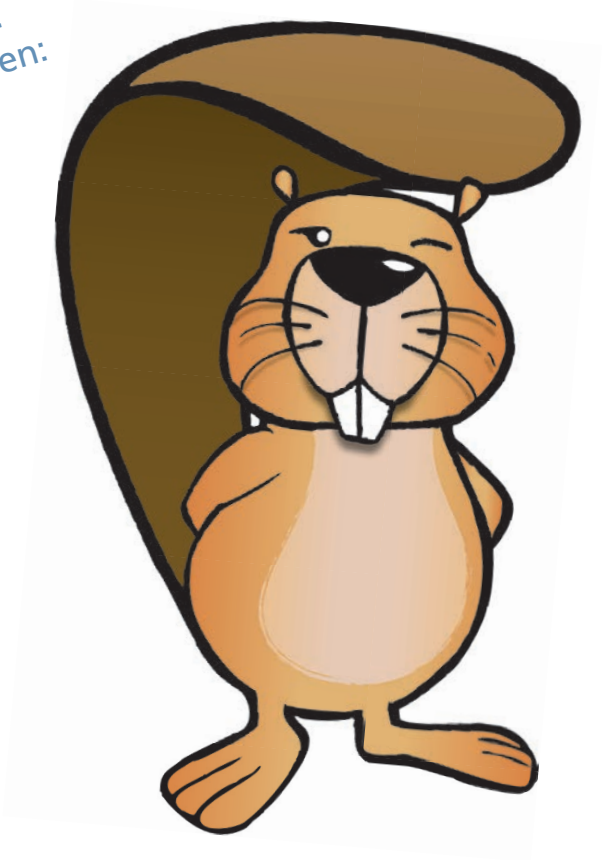
Aber manchmal ist der Stapel falsch:
Dann sind am Ende die Socken über den Schuhen oder die Unterhose über der Hose.
Oh je!

Diese Stapel sind fast alle falsch. Nur einer ist richtig.
Welcher?



A) **B)** **C)** **D)**

Auch mit Biberheften kannst du Stapel bilden.
Es gibt sie schon seit dem Jahr 2007.
Alle Biberhefte kannst du hier finden:
bwinf.de/biber/downloads



Antwort D ist richtig:
Bei Stapel A zieht Bruno die Socken über die Schuhe an.
Bei den Stapeln B und C zieht Bruno die Unterhose über die Hose an.
Bei Stapel D sind zwar merkwürdigerweise die Socken zuerst an der Reihe, aber am Ende ist alles richtig: Die Socken sind in den Schuhen, und die Unterhose ist unter der Hose.

Das ist Informatik!
Das ist klar: Die Unterhose muss vor der Hose und die Socken müssen vor den Schuhen angezogen werden. Aber ob das Hemd vor den Socken angezogen wird oder umgekehrt, ist egal. Nur gleichzeitig geht es nicht, und deshalb wird eine Reihenfolge benötigt, bei der eine Sache nach der anderen kommt. Eine Reihenfolge, bei der alle „vor-Sachen“ (z. B. die Socken) auch wirklich vor ihren „danach-Sachen“ (z. B. die Schuhe) und alle anderen Sachen irgendwo stehen, wird in der Mathematik als topologische Sortierung bezeichnet. Die Informatik kennt Verfahren, mit denen man eine topologische Sortierung berechnen kann – oder herausfinden kann, dass es keine gibt: Wenn Brunos Eltern auf den merkwürdigen Gedanken kämen, dass er das Hemd vor der Unterhose, die Unterhose vor der Hose und die Hose vor dem Hemd anziehen müsste, könnte Bruno niemals einen richtigen Stapel legen.

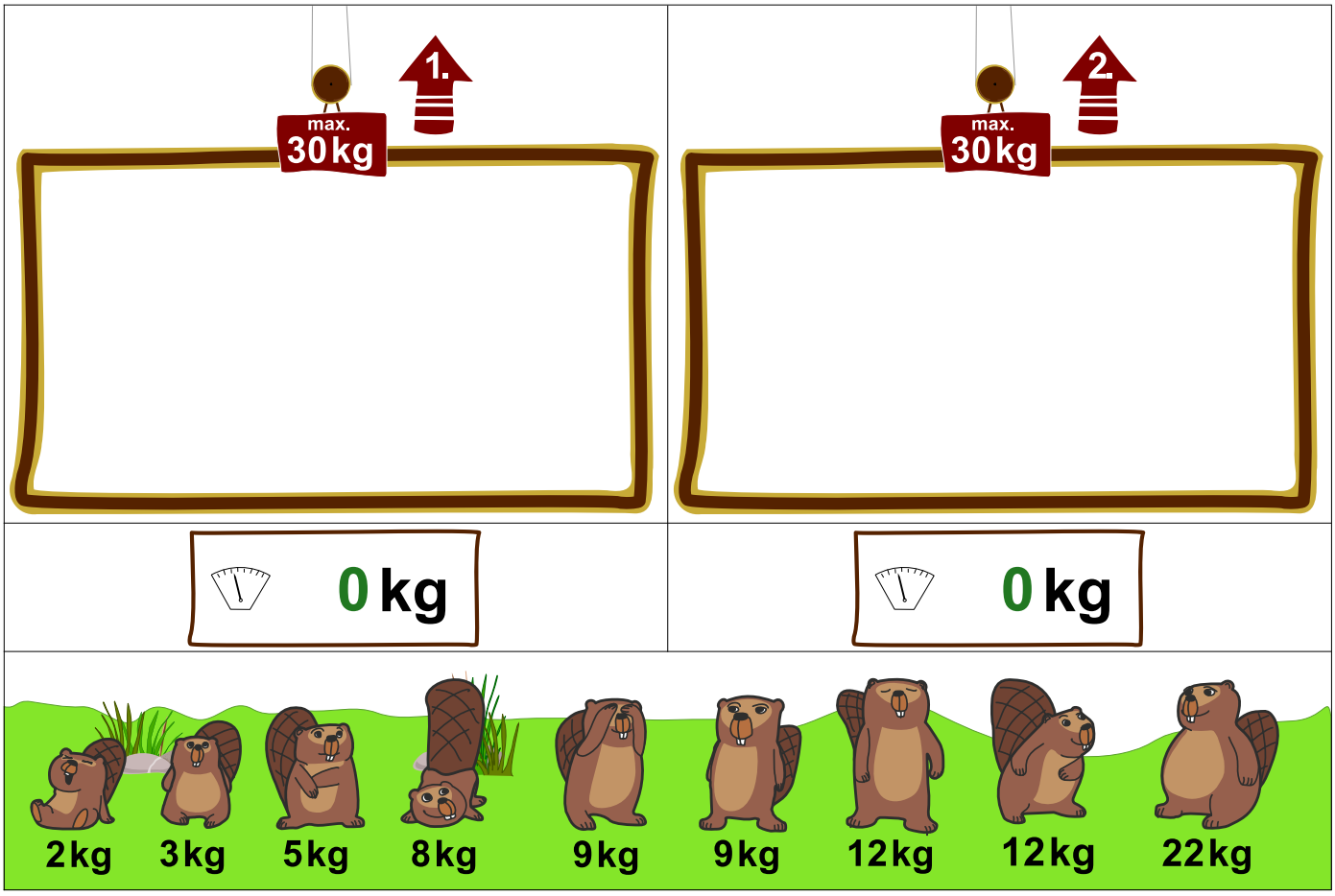
Aufzüge

Auf einer Reise sind die Biber an einem Aussichtsturm angekommen.
Der Turm hat zwei Aufzüge. Jeder Aufzug kann höchstens 30 kg transportieren.

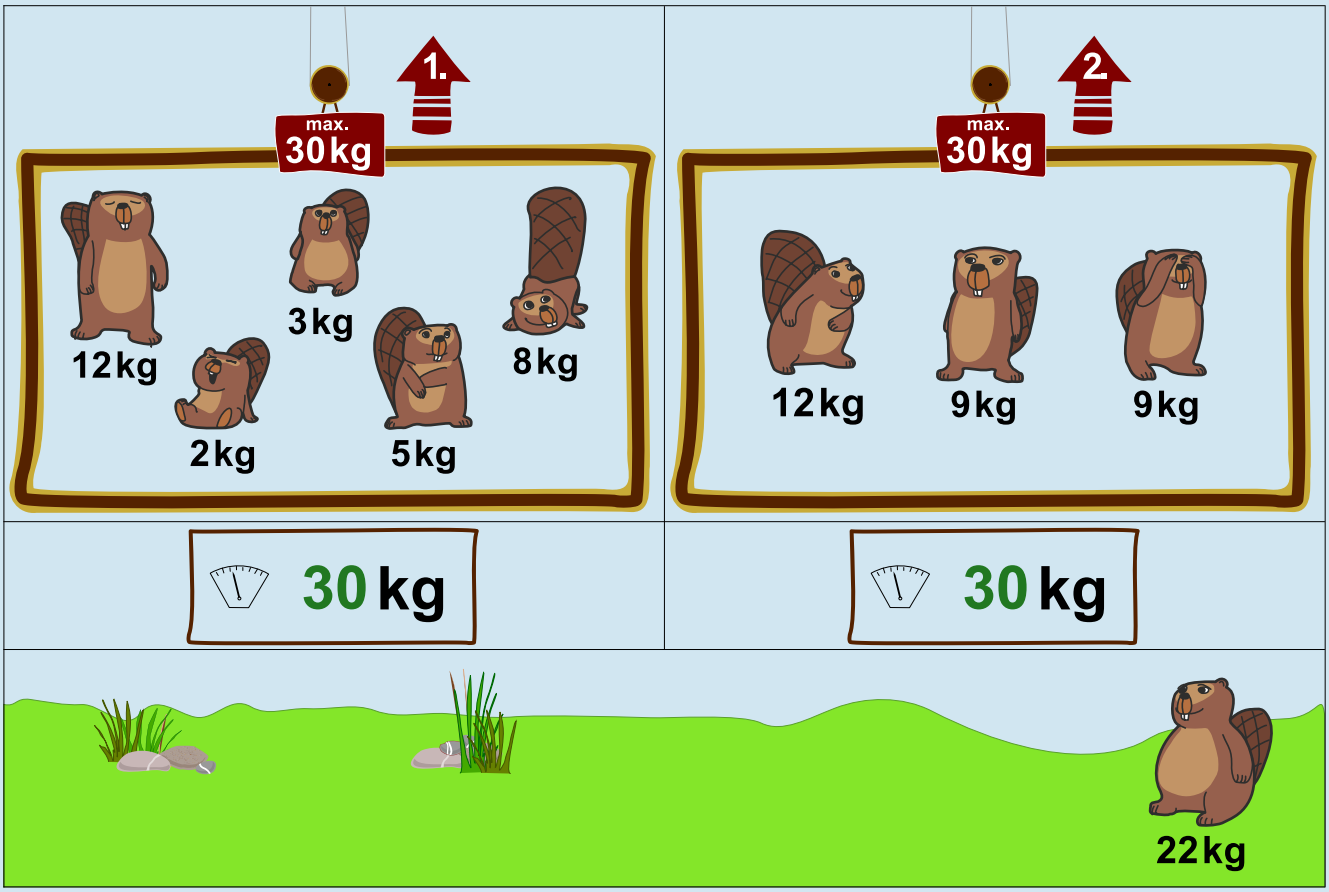
Die Biber wollen schnell auf den Turm.

**Verteile die Biber so auf die Aufzüge,
dass möglichst viele Biber gleichzeitig hineinpassen.**

Ziehe die Biber in die Aufzüge.



So ist es richtig:
So können acht Biber auf die Aufzüge verteilt werden:

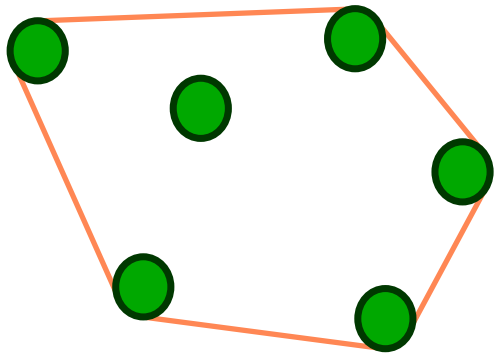
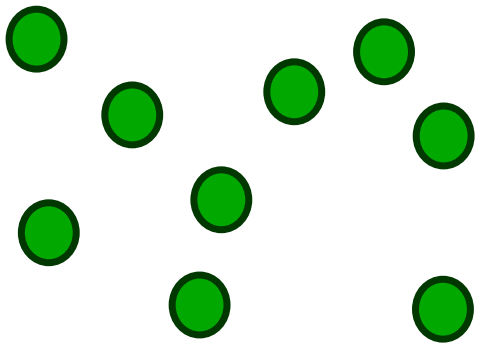


Da beide Aufzüge gleich viel Gewicht transportieren können, können die Biber aus Aufzug 1 auch mit Aufzug 2 fahren – und umgekehrt. Außerdem können die beiden 12 kg schweren Biber ausgetauscht werden. Bei allen anderen Verteilungen von Bibern auf die Aufzüge, bei denen das maximale Transportgewicht nicht überschritten wird, passen weniger Biber in die Aufzüge hinein.

Das ist Informatik!
Viele Menschen sind fasziniert davon, Dinge zu optimieren – häufig übrigens, um Kosten zu sparen und so Profit zu maximieren. Auch ein wichtiger Bereich der Informatik befasst sich mit Verfahren zur Optimierung. Computerprogramme helfen dann dabei, vorhandene Ressourcen (wie die Aufzüge in dieser Biberaufgabe) optimal, also möglichst sparsam zu nutzen.
Manche Optimierungsprobleme lassen sich mit einem „gierigen" (engl.: greedy) Algorithmus lösen. Dabei wird jeder Schritt zur Lösung (hier: die Platzierung eines Bibers in einen Aufzug) so gewählt, dass er möglichst viel Gewinn bzgl. des Findens einer besten Lösung bringt – das ist gierig. In dieser Biberaufgabe würde man z. B. „gierig" vorgehen, wenn man zuerst einen Aufzug füllt und den nächsten Biber so platziert, dass der Aufzug danach noch möglichst viel Gewicht aufnehmen kann: Leichte Biber zuerst! Doch mit diesem Vorgehen würde man die beste Verteilung der Biber nicht finden. Auch bei vielen anderen Optimierungsproblemen hilft ein gieriger Algorithmus nicht weiter, und komplexere Algorithmen werden benötigt, um optimale Lösungen zu finden.
Gier ist nicht die Lösung, und Ressourcen sollen sparsam genutzt werden: Wer sagt da noch, dass man von der Informatik nichts Allgemeines lernen kann!

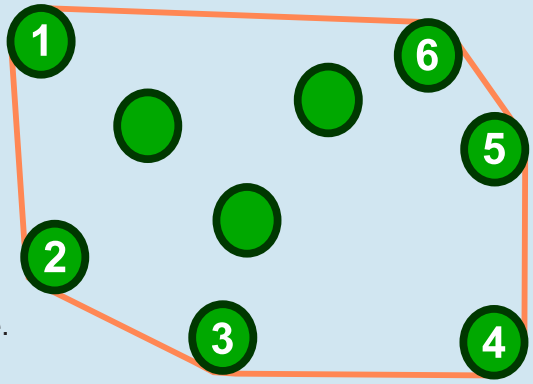
Bäume-Band

Die Biber spannen immer ein langes Band um Bäume, die sie fällen wollen.

Gestern wollten sie sechs Bäume fällen. Das Band hat aber nur fünf Bäume berührt. Aus der Luft sah das so aus:	Heute wollen die Biber diese Bäume fällen:
	

Wie viele Bäume berührt das gespannte Band diesmal?
A) 3 Bäume **B)** 5 Bäume **C)** 6 Bäume **D)** 9 Bäume

Antwort C ist richtig:
Die Biber spannen das Band so um die Bäume:



Das Band berührt die sechs nummerierten Bäume.

Das ist Informatik!
Wenn die Bäume im Lösungsbild winzige Punkte wären, hätte das Biber-Bäume-Band die Form eines Sechsecks. Dieses Sechseck ist das kleinste Vieleck, auf dem alle zu fällenden Bäume stehen.
Ein solches kleinstes Vieleck, das alle Punkte aus einer vorgegebenen Menge enthält, heißt in der Mathematik auch „konvexe Hülle“ dieser Punktmenge. Dabei bedeutet „konvex“, dass sich etwas nach außen dehnt, wie z. B. die konvexe Linse einer Lupe. Und eine „Hülle“ ist etwas, das etwas anderes umschließt, so wie die Haut den Körper, dabei aber nicht größer als nötig ist. Das Biber-Bäume-Band ist also wie eine solche konvexe Hülle: Es umschließt alle Bäume und geht zwischen zwei Bäumen nicht nach innen – und „nicht nach innen“ ist in der Mathematik schon so viel wie „nach außen“.
In der Informatik ist es häufig wichtig, die konvexe Hülle einer Menge von Punkten zu berechnen:

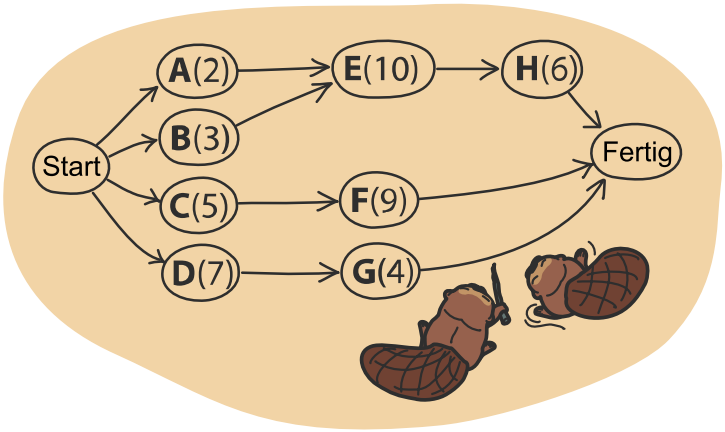
- Mustererkennung: Ist in einem Bild ein Gesicht zu sehen?
- Handschrifterkennung: Ist ein handgeschriebenes Zeichen der Buchstabe B?
- Geographische Informationssysteme: Wie groß ist ein Überschwemmungsgebiet oder ein Flusssystem?
- Verpackungen: Was ist die kleinste Menge an Material, die genügt, einen bestimmten Gegenstand zu verpacken?

Die Informatik kennt Verfahren, welche die konvexe Hülle einer Punktmenge effizient berechnen können. Sie funktionieren also auch dann noch gut, wenn die Punktmenge sehr groß ist.

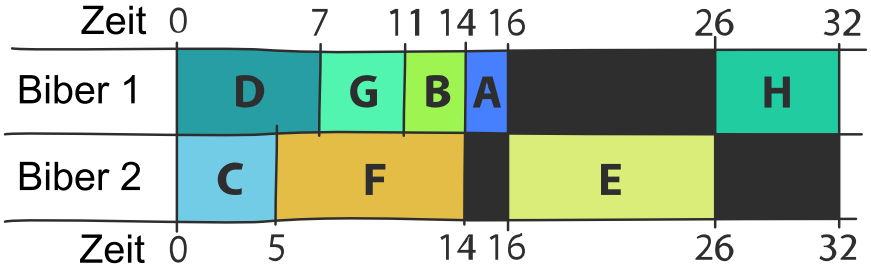
Biber-Arbeit

Zwei Biber bauen einen Damm. Dazu müssen sie acht Aufgaben erledigen:
Bäume fällen, Äste entfernen, Stämme ins Wasser bringen usw.

Die Biber machen sich erstmal ein Bild.
Für jede Aufgabe gibt es einen Buchstaben. In Klammern steht, wie viele Stunden es dauert, die Aufgabe zu erledigen. Ein Pfeil sagt, dass eine Aufgabe vor einer anderen erledigt werden muss. Zum Beispiel kann E erst begonnen werden, wenn A und B beide erledigt sind.



Die Biber können gleichzeitig arbeiten, aber an unterschiedlichen Aufgaben.
Hier ist ihr Arbeitsplan. Damit wird der Damm in 32 Stunden fertig.
Es geht aber schneller!

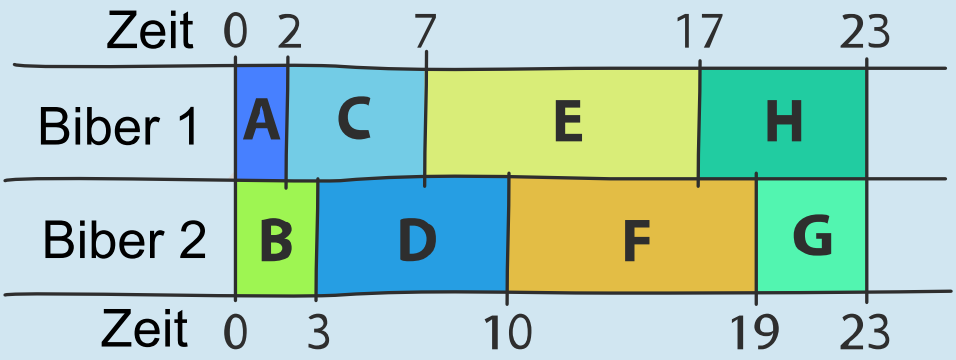


Erstelle einen Arbeitsplan, mit dem der Damm so schnell wie möglich fertig wird!

Ziehe dazu die Aufgaben nach unten in den Plan.
Aufgaben, die noch nicht begonnen werden können, sind grau gefärbt.
Wenn du mit dem Plan noch nicht zufrieden bist, ziehe Aufgaben zurück nach oben.

Biber 1	
Biber 2	

So ist es richtig:
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 Arbeitsplan, der mit 23 Stunden auskommt. Schneller geht es nicht, denn die beiden Biber arbeiten ohne Pause.



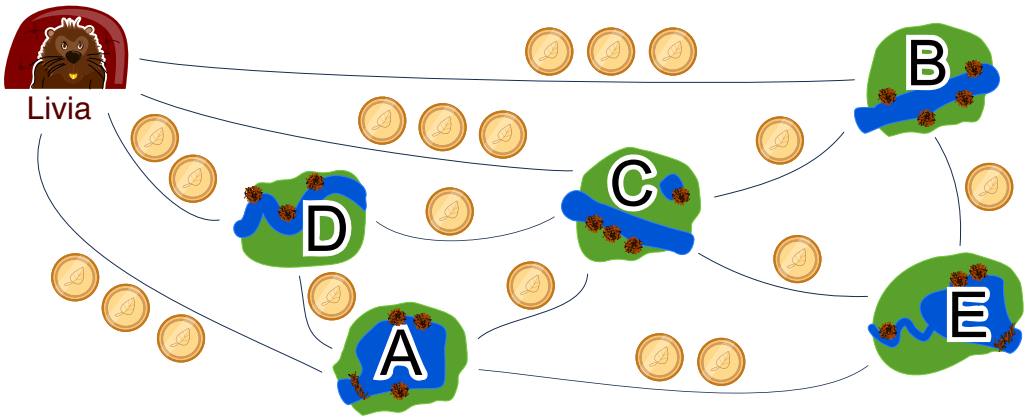
Das ist Informatik!
Menschen sind ungeduldig, und deshalb wird oft verlangt, dass Arbeiten möglichst schnell erledigt werden. Wenn mehrere „Betriebsmittel“ zur Verfügung stehen, seien es Menschen, Maschinen oder Biber, spielt die Verteilung der „Jobs“ auf diese Betriebsmittel für die Schnelligkeit eine wichtige Rolle.
Auch in Computern sollen die Jobs, die durch die vielen, gleichzeitig darauf ablaufenden Prozesse zu erledigen sind, gut auf die Betriebsmittel, etwa die Prozessorkerne verteilt werden. Für dieses „Scheduling“ gibt es viele verschiedene, von der Informatik gut untersuchte Strategien. Der Arbeitsplan in dieser Biberaufgabe wurde so erstellt, dass unter den anstehenden Jobs der mit der längsten Dauer einem gerade „arbeitslosen“ Biber zugeteilt wurde – in diesem Fall eine schlechte Strategie. Im Allgemeinen funktioniert es besser, wenn kurze Jobs zuerst erledigt werden: Die Strategie „shortest-job-next“ (kürzester Job zuerst) führt zu einer im Durchschnitt minimalen Wartezeit der Jobs.

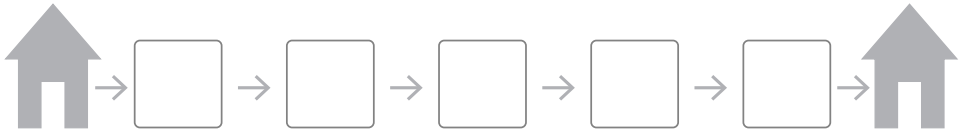
Bibertour

Biber Livia will ihre Freunde mit dem Bus besuchen. Die Freunde leben in den Dörfern A, B, C, D und E. Zwischen den Dörfern gibt es Busverbindungen. Das Bild zeigt, wie viel die einfache Fahrt jeweils kostet. Livia plant eine Bibertour: Sie startet an ihrem Bau, besucht jedes Dorf genau einmal und kehrt dann zu ihrem Bau zurück.
Zum Beispiel ist das eine Bibertour: Bau ⟶ B ⟶ E ⟶ A ⟶ D ⟶ C ⟶ Bau
Diese Bibertour kostet 11 Bibertaler.

Finde eine Bibertour, die so wenig wie möglich kostet.

Ziehe die Dorf-Buchstaben A bis E in die Felder unten, um die Bibertour zu beschreiben.
Gibt es mehrere Bibertouren, die gleich wenig kosten, kannst du irgendeine davon beschreiben.





So ist es richtig:
Es gibt zwei Bibertouren, die möglichst wenig kosten:
Bau ⟶ B ⟶ E ⟶ C ⟶ A ⟶ D ⟶ Bau
Bau ⟶ D ⟶ A ⟶ C ⟶ E ⟶ B ⟶ Bau
Die zweite Bibertour ist wie die erste Tour, nur rückwärts gefahren.
Beide Bibertouren kosten jeweils neun Bibertaler.
Eine Bibertour, die noch weniger kostet, gibt es nicht: Die Fahrten vom Bau zum ersten Dorf und vom letzten Dorf (das nicht das gleiche sein kann wie das erste Dorf) zum Bau zurück kosten mindestens fünf Bibertaler. Die vier Fahrten vom ersten Dorf über die anderen drei Dörfer zum letzten Dorf kosten jeweils mindestens einen Bibertaler. Und 5 + 4 = 9.

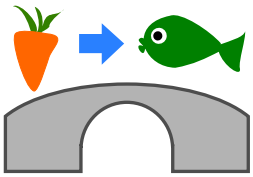
Das ist Informatik!
Kommt uns dieses Problem nicht irgendwie bekannt vor? Es geht um eine Rundtour, bei der alle gegebenen Stationen genau einmal besucht werden müssen und die möglichst kurz ist (wenn man die in Münzen gegebenen Kosten eines Weges als Weglänge versteht). Wenn man das Problem so allgemein beschreibt, wird offensichtlich: Diese Biberaufgabe ist ein Beispiel für das Problem des Handlungsreisenden, im Englischen auch als „Traveling Salesman Problem“ bzw. kurz TSP bekannt.
Das TSP ist eines der bekanntermaßen sehr schwierigen Probleme der Informatik. Im schlimmsten Fall muss man alle möglichen Wege bestimmen und daraus den kürzesten Weg auswählen. Die Anzahl der möglichen Wege liegt für n Stationen aber in der Größenordnung von $n!$ – das ist schon für $n = 20$ eine Zahl mit 19 Ziffern. Die Informatik kennt aber Verfahren, die mit sehr viel weniger Aufwand zu optimalen oder zumindest sehr guten Ergebnissen für größere Werte kommen. Das ist gut, denn das TSP hat viele wichtige Anwendungen, unter anderem beim Design von Mikrochips.

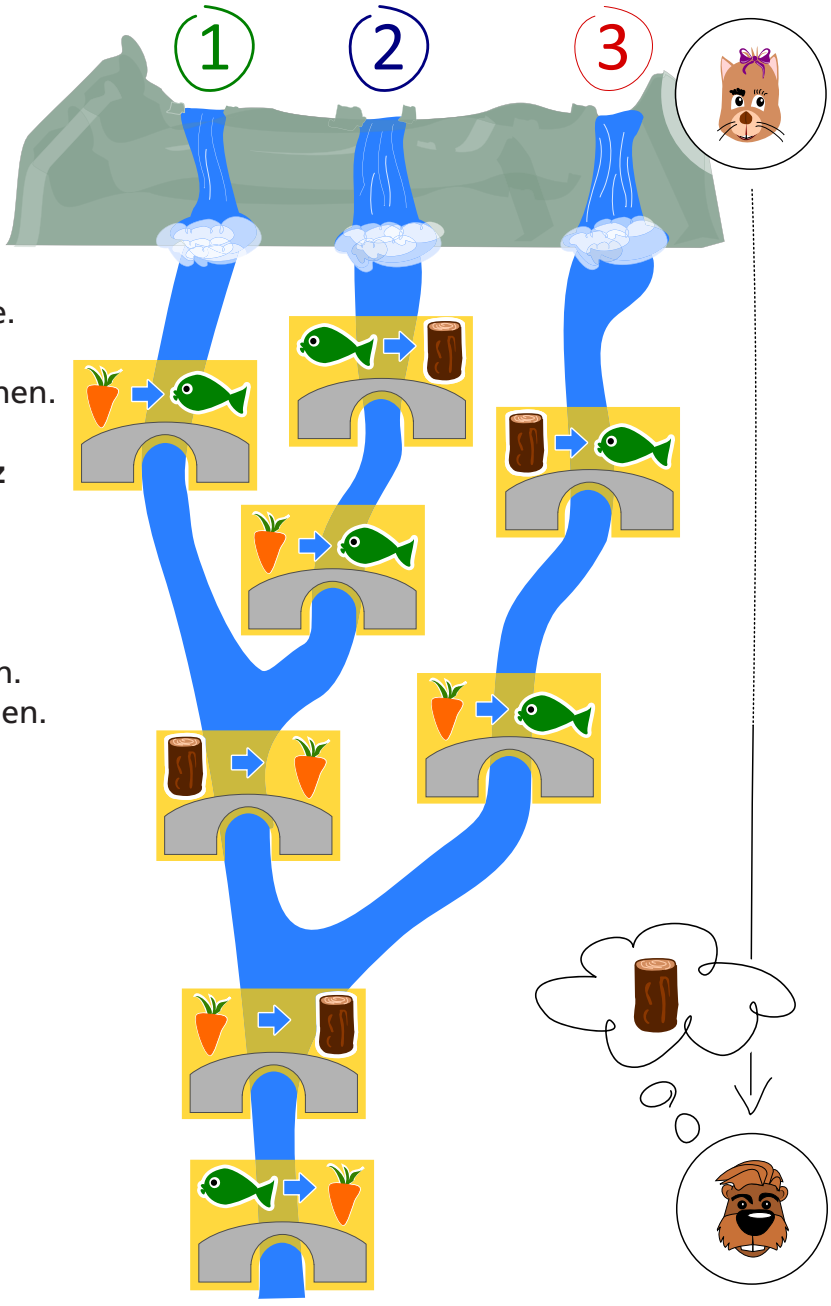
Brückentrolle

Von einem Berg kommen drei Flüsse.
Sie fließen unter Brücken hindurch.

Katja kann einen Gegenstand in einen der drei Flüsse fallen lassen:
ein Stück Holz, einen Fisch oder eine Karotte.

Unter allen Brücken sitzen Trolle.
Sie ersetzen Gegenstände, die unter den Brücken hindurch fließen.
Der Troll unter der Brücke oben links
ersetzt zum Beispiel jede Karotte durch einen Fisch:





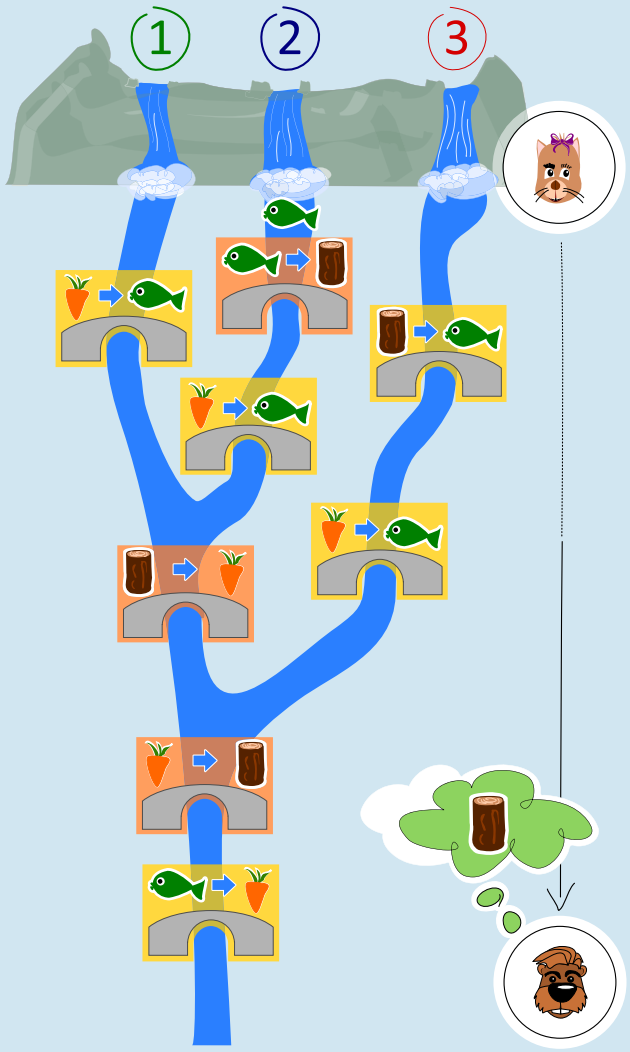
Justus wartet hinter der letzten Brücke.
Er möchte ein Stück Holz bekommen.
Katja weiß, was die Brückentrolle machen.

Was muss Katja tun, damit Justus Holz bekommt?

A) Sie lässt einen Fisch in Fluss 1 fallen.
B) Sie lässt einen Fisch in Fluss 2 fallen.
C) Sie lässt eine Karotte in Fluss 3 fallen.
D) Sie lässt ein Stück Holz in Fluss 3 fallen.

Antwort B ist richtig:
Wenn Katja einen Fisch in Fluss 2 fallen lässt, werden drei Trolle aktiv – siehe Bild. Der erste Troll ersetzt den Fisch durch ein Stück Holz. Der Troll an der nächsten Brücke hat dann nichts zu tun. Eine Brücke weiter ersetzt der Troll das Stück Holz durch eine Karotte. Der Troll von der nächsten Brücke ersetzt die Karotte wieder durch ein Stück Holz. Der Troll an der letzten Brücke hat dann ebenfalls nichts zu tun. Justus kann das Stück Holz aus dem Fluss fischen.

Bei den anderen Antworten bekommt Justus kein Holz.
Antwort A: Wenn Katja einen Fisch in Fluss 1 fallen lässt, wird der an der letzten Brücke durch eine Karotte ersetzt.
Antwort C: Wenn Katja eine Karotte in Fluss 3 fallen lässt, wird diese zwischendurch durch einen Fisch ersetzt. Der Fisch wird dann an der letzten Brücke durch eine Karotte ersetzt.
Antwort D: Wenn Katja ein Stück Holz in Fluss 3 fallen lässt, wird dieses an der ersten Brücke durch einen Fisch ersetzt. Der Fisch wird, wie bei Antwort C, an der letzten Brücke durch einen Karotte ersetzt. Wieder kein Holz für Justus!



Das ist Informatik!
Die Brückentrolle arbeiten nach dem EVA-Prinzip der Datenverarbeitung: Katjas **E**ingabe wird von ihnen **v**erarbeitet, und Justus erhält die **A**usgabe. Ein System, das rein nach diesem Prinzip arbeitet, muss sich nichts merken: Die Ausgabe hängt nur von der Eingabe ab, nicht aber davon, was bei früheren Verarbeitungen passiert ist. Das ist wie bei einer Funktion in der Mathematik, wo der Funktionswert nur vom Argumentwert abhängt.
Gemeinsam arbeiten die Brückentrolle in dieser Biberaufgabe also wie eine Funktion. Aber auch jeder einzelne tut das. Die große Brückentroll-Funktion ist so aus vielen kleinen Brückentroll-Funktionen zusammengesetzt. Es gibt Programmiersprachen, die ganz ähnlich nur die Programmierung von Funktionen und deren Zusammensetzung zu komplexeren Funktionen erlauben. Die funktionalen Programme, die man in diesen Sprachen schreiben kann, lassen sich gut analysieren und auf ihre Korrektheit prüfen – dank des EVA-Prinzips.

Büchertausch

In einer Bibliothek sitzen drei weise Personen nebeneinander, jede an einem Tisch mit zwei Büchern. Mit einem Spiel wollen sie die Bücher sortieren: Büchertausch.
Das Spiel läuft in Runden ab. Es gibt zwei Arten von Runden:

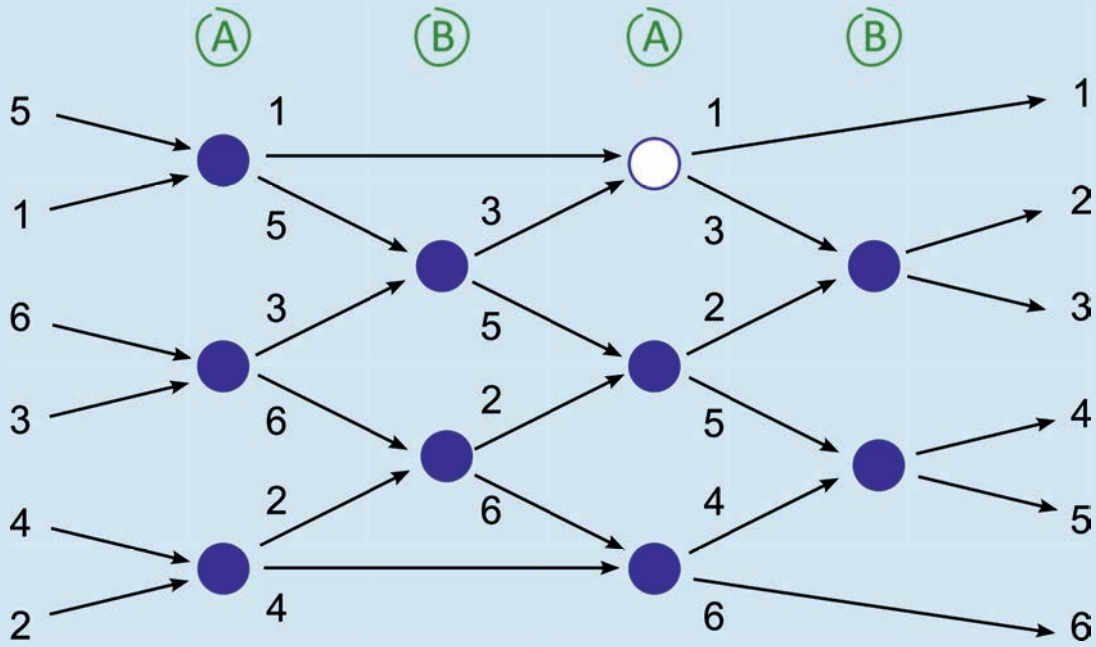
A) Die beiden Bücher auf jedem Tisch dürfen (müssen aber nicht) getauscht werden.
B) Jedes Buch darf (aber muss nicht) mit einem benachbarten Buch von einem Nachbartisch getauscht werden.

A- und B-Runden wechseln sich ab. Das Spiel beginnt mit einer A-Runde.
In jeder Runde darf jedes Buch höchstens einmal getauscht werden.

Am Anfang liegen die Bücher wie im Bild.
Wie viele Runden sind insgesamt mindestens notwendig um die Bücher zu sortieren, also in diese Reihenfolge zu bringen: 1, 2, 3, 4, 5, 6 ?

A) drei Runden **B)** vier Runden **C)** fünf Runden **D)** sechs Runden

Antwort B ist richtig:
Das Bild zeigt, wie die Bücher beim Büchertausch sortiert werden. In jeder Runde werden die benachbarten Bücher, die gerade getauscht werden dürfen, miteinander verglichen. Wenn diese beiden Bücher schon sortiert sind (das linke Buch hat eine kleinere Nummer als das rechte Buch), dann passiert nichts. Ansonsten werden die Bücher getauscht.
In der ersten Runde (Art A) werden auf jedem Tisch die beiden Bücher getauscht. In der zweiten Runde (Art B) werden benachbarte Bücher von Nachbartischen getauscht. In der dritten Runde (wieder Art A) werden nur auf den beiden rechten Tischen die Bücher getauscht, und in der vierten Runde (wieder Typ B) werden alle benachbarten Bücher von Nachbartischen getauscht: fertig. Schneller geht es nicht. Denn z. B. Buch 5 muss um vier Positionen nach rechts wandern. Dazu muss es mindestens vier Mal getauscht werden, denn bei jedem Tausch kann ein Buch sich nur um eine Position bewegen.

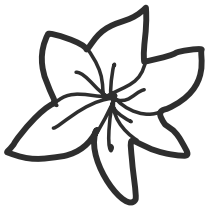
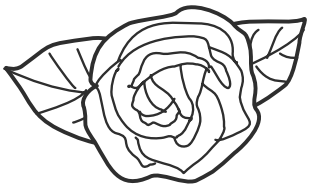


Das ist Informatik!
Das Sortieren in dieser Biberaufgabe liefert ein Beispiel für einen parallelen Algorithmus: ein Sortiernetz. Ein Sortiernetz kann, wie oben, durch Pfeile und Kreise dargestellt werden. Die Kreise stellen die Vergleichseinheiten dar. Die Pfeile zeigen, wie sich die zu sortierenden Objekte zwischen den Einheiten bewegen. Zu jeder Einheit führen zwei Pfeile, so dass auf einem Knoten zwei Objekte miteinander verglichen werden können. Wieder zwei Pfeile führen von jeder Einheit weg; auf dem oberen bewegt sich das kleinere Objekt weiter, auf dem unteren das größere. Wenn man den Pfeilen von links nach rechts folgt, kann man erkennen, wie ein Objekt nach einigen Vergleichen allmählich seine richtige Position in der angestrebten Reihenfolge einnimmt.
Die Einheiten, die im Bild übereinander stehen, können ihre Vergleiche gleichzeitig, also parallel ausführen. Durch die Parallelität sind Sortiernetze oft schneller als nicht-parallele Sortierverfahren.

https://en.wikipedia.org/wiki/Sorting_network

Claras Blumen

Clara liebt Blumen! Im Blumenladen gibt es diese Sorten:

			
Gladiolen	**Lilien**	**Tulpen**	**Rosen**

Jede Sorte ist in diesen Farben erhältlich: Weiß, Blau und Gelb.

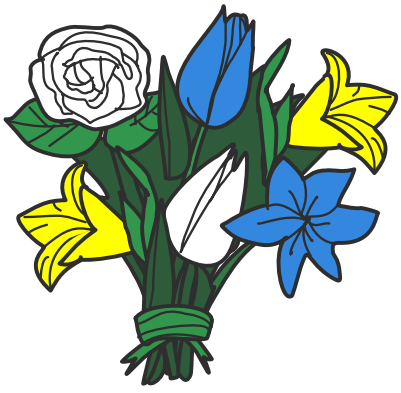
Clara möchte einen Blumenstrauß kaufen. Aber sie hat besondere Wünsche:

1. Jede Farbe soll genau zweimal vorkommen,
2. Blumen der gleichen Sorte sollen nicht die gleiche Farbe haben und
3. jede Sorte darf höchstens zweimal vorkommen.

Welchen Blumenstrauß wird Clara kaufen?

A)

B)



C)

D)

Antwort D ist richtig:
Im Blumenstrauß A gibt es drei weiße Blüten: Claras erster Wunsch ist nicht erfüllt.
Im Blumenstrauß B gibt es drei Rosen: Claras dritter Wunsch ist nicht erfüllt.
Im Blumenstrauß C haben die beiden Gladiolen die gleiche Farbe: Claras zweiter Wunsch ist nicht erfüllt.
Blumenstrauß D erfüllt alle von Claras Wünschen:
1. Jede Farbe kommt genau zweimal vor.
2. Die Blumen der gleichen Sorte haben unterschiedliche Farben:
   Die beiden Tulpen sind blau und weiß, die beiden Gladiolen sind gelb und blau.
3. Jede Sorte kommt höchstens zweimal vor:
   Im Blumenstrauß sind zwei Gladiolen, eine Lilie, zwei Tulpen und eine Rose.

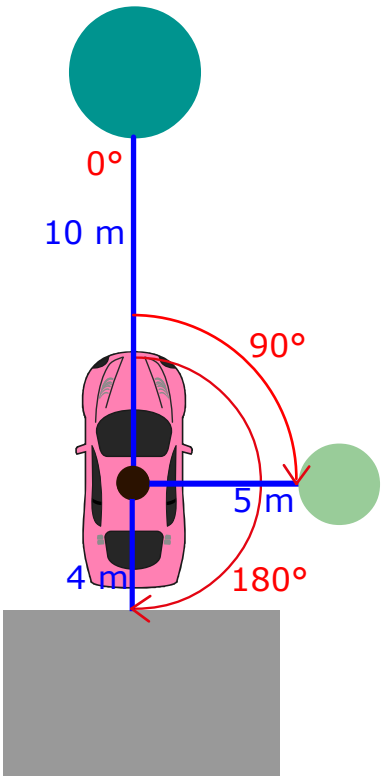
Das ist Informatik!
In dieser Biberaufgabe solltest du aus einer Menge von Dingen das Ding heraussuchen, das bestimmte Bedingungen erfüllt. Das ist wie bei der Suche nach Daten in einer Datenbank, z. B. in Online-Shops: alle roten Blusen in Größe M, alle braunen oder schwarzen Sneaker für Herren der Marke „Bibersports“ in Größe 40, einen maximal fünf Jahre alten Film der Kategorie Komödie mit Emma Stone, und so weiter.
In all diesen Beispielen werden, wie in dieser Biberaufgabe, mehrere Teilbedingungen zu einer Gesamtbedingung verknüpft. Dazu kann man unter anderem logische Operatoren verwenden. Wenn alle Teilbedingungen gleichzeitig erfüllt sein müssen, verwendet man den Operator UND:
Art = Bluse UND Farbe = rot UND Größe = M.
Wenn nur mindestens eine von mehreren Teilbedingungen erfüllt sein muss, kann man den Operator ODER verwenden:
… (Farbe = Braun ODER Farbe = Schwarz).
Für Suchen in Datenbanken kann man mit Hilfe von Abfragesprachen sehr komplexe Bedingungen formulieren.

Das vermisste Auto

Ein selbstfahrendes Auto wird vermisst. Es blieb mit leerem Akku irgendwo in der Stadt stehen. So ein Pech!

Kurz vorher konnte das vermisste Auto noch ein Modell seiner Umgebung senden. In diesem Modell wird jedes Objekt, das der 360°-Sensor erfasst hat, durch zwei Werte beschrieben:

1. Der Winkel der Blickrichtung des Sensors zu dem Objekt (siehe Bild). Geradeaus ist 0°.
2. Die Entfernung vom Sensor zu dem Objekt.

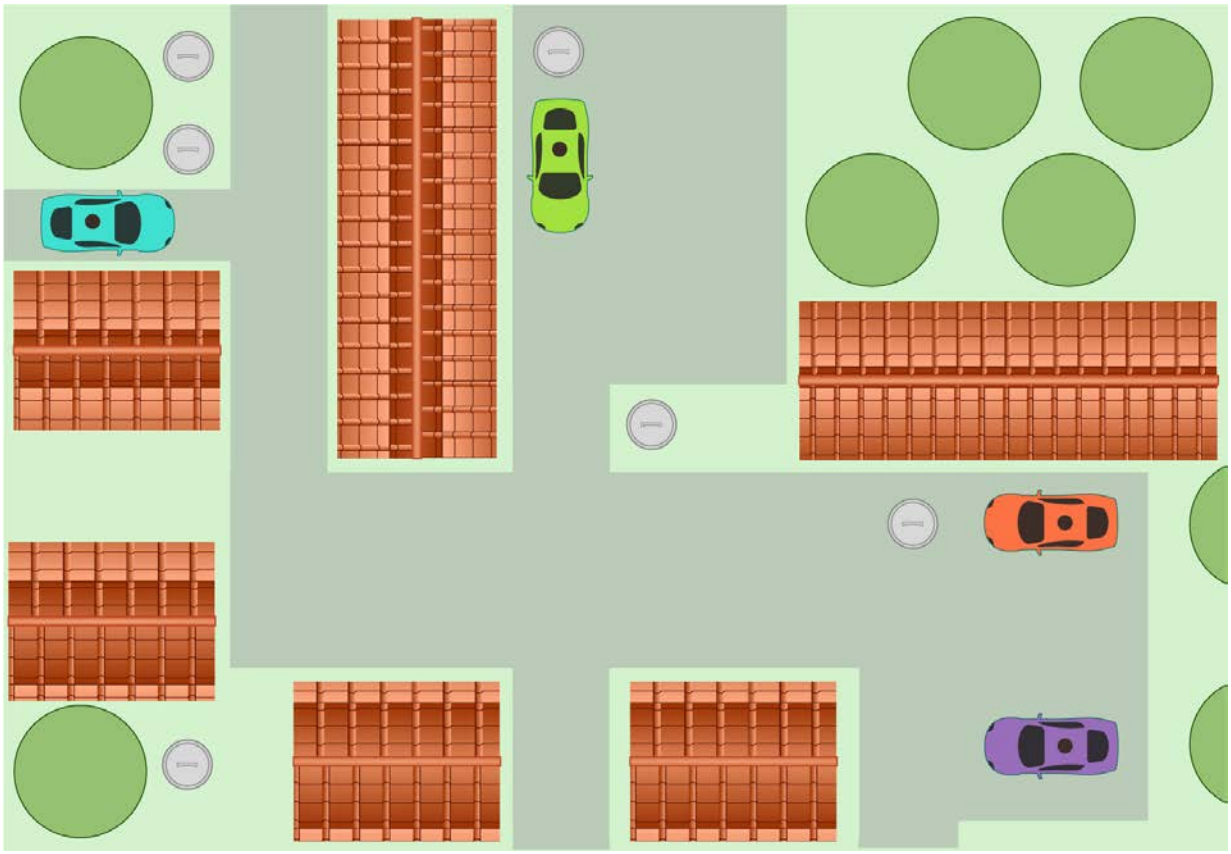


In dem Beispiel ist das Modell [(0, 10), (90, 5), (180, 4)].

Das vermisste Auto sendete dieses Modell:
[(0, 5), (90, 3), (180, 5), (270, 8)].

Finde das vermisste Auto auf der Karte!

Wähle das vermisste Auto durch einen Mausklick aus.



So ist es richtig:
Das orange Auto (im Bild mitte rechts) ist das vermisste Auto. Das vermisste Auto ist von folgenden Objekten umgeben:

Ein Objekt in einer Entfernung von 5m vor den Auto,
ein Objekt in einer Entfernung von 3m an seiner rechten Seite,
ein Objekt in einer Entfernung von 5m hinter dem Auto und
ein Objekt in einer Entfernung von 8m links neben dem Auto.

Das blaue Auto ist nicht das vermisste Auto, weil die Objekte an seiner rechten und linken Seite gleich weit entfernt sind.
Auch das grüne Auto kann nicht das vermisste Auto sein; denn das Objekt hinter ihm (graue Kreisfläche) ist viel näher als jedes mögliche Objekt vor ihm, das man im Bild gar nicht sieht.
Das violette Auto ist nicht das vermisste Auto, weil das Objekt hinter ihm (grüne Kreisfläche) viel näher als das Objekt vor ihm ist (der Häuserblock).
Das orange Auto ist das vermisste Auto; denn die Objekte vor und hinter ihm sind gleich weit entfernt und das Objekt auf seiner rechten Seite (Häuserblock) ist viel näher als das Objekt auf seiner linken Seite (violettes Auto).

Das ist Informatik!
Unter einem Modell versteht man – auch in der Informatik – ein vereinfachtes Abbild der Wirklichkeit, das nur solche Aspekte berücksichtigt, die für einen bestimmten Zweck wichtig sind. Das selbstfahrende, autonome Fahrzeug in dieser Biberaufgabe verwendet ein sehr einfaches Modell seiner Umgebung, das nur die nächstgelegenen Objekte umfasst.
Reale autonome Fahrzeuge verwenden die LIDAR-Techologie (light detection and ranging) um ihre Umgebung zu erfassen. Die Software des Fahrzeugs erstellt aus den so erzeugten Daten ein komplexes 3D-Modell aller Objekte in einem Abstand von bis zu mehreren hundert Metern. Aber auch in einem komplexen Modell müssen nicht alle Aspekte der Umgebung erfasst werden: Wichtig sind insbesondere Position und Ausmaße der Objekte, während ihre Farbe ignoriert werden kann.

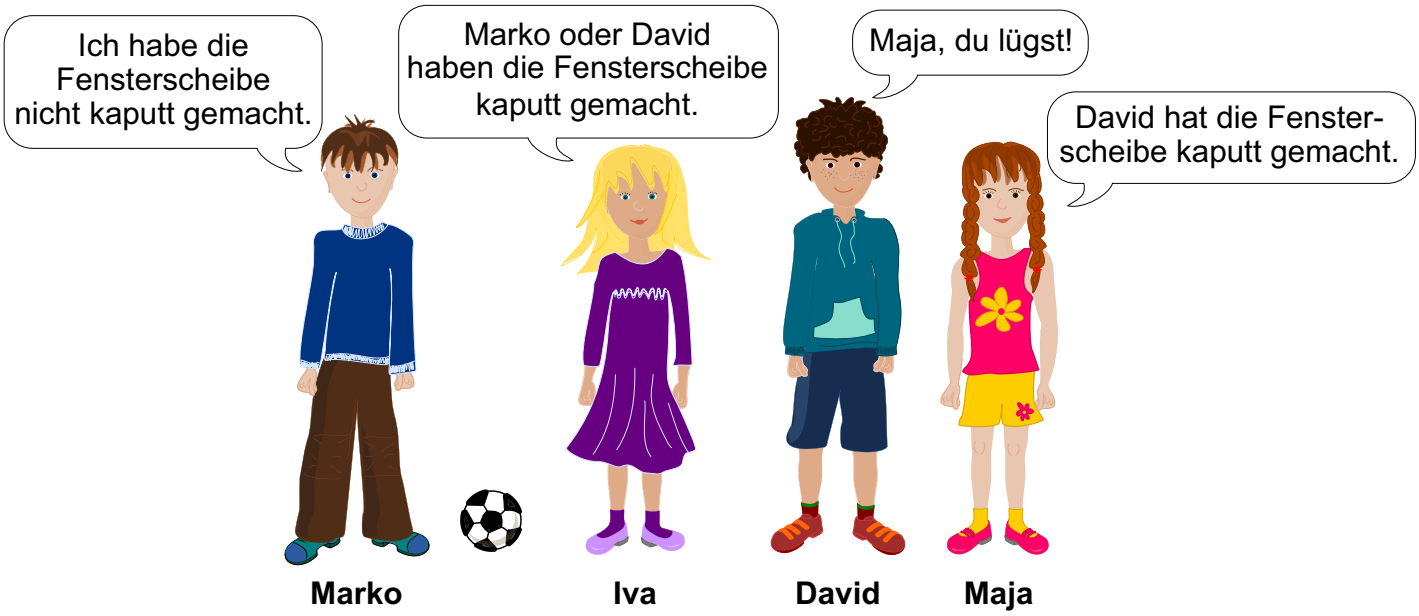
https://de.wikipedia.org/wiki/Lidar

Die Fensterscheibe

Maja, David, Iva und Marko spielen Fußball bei Annas Haus.
Auf einmal geht eine Fensterscheibe kaputt.
Anna fragt die vier Kinder, wer es war.
Anna weiß, dass drei von ihnen immer die Wahrheit sagen.
Bei dem vierten Kind weiß sie es nicht.

Wer hat die Fensterscheibe kaputt gemacht?

Lies genau durch, was die Kinder sagen. Klicke dann auf ein Kind, um es auszuwählen.

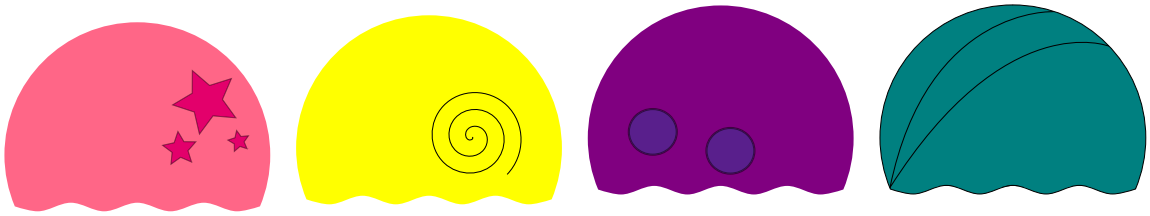


So ist es richtig:
David hat die Fensterscheibe kaputt gemacht. Die Aussagen von Maja und David können nicht gleichzeitig wahr sein. Einer der beiden sagt also die Unwahrheit. Damit weiß Anna, dass Iva und Marko sicher die Wahrheit sagen. Durch Iva weiß sie, dass es Marko oder David gewesen sein können, doch Marko war es laut eigener (wahrer) Aussage nicht. Damit bleibt nur David übrig.

Das ist Informatik!
Die Erklärung der Lösung oben ist doch logisch, oder? „Logisch“ ist im Allgemeinen, wenn man aus Beobachtungen oder bekannten Tatsachen Schlussfolgerungen ziehen und neue Erkenntnisse herleiten kann. Schon in der Antike haben kluge Menschen wie Aristoteles das logische Schließen näher untersucht. Es war eben immer schon wichtig, gründlich beurteilen zu können, ob eine Behauptung wahr ist oder falsch.
Die Kinder in dieser Biberaufgabe machen einfache Aussagen. Jede einzelne Aussage kann nur entweder wahr oder falsch sein. George Boole hat 1847 als erster beschrieben, wie mit solchen Aussagen gerechnet werden kann, und damit die Aussagenlogik formalisiert. Ein aussagenlogischer Kalkül (so nennt man eine Menge von Rechenvorschriften in der Logik) beschreibt, wie aus wahren Aussagen andere wahre Aussagen konstruiert werden können, ohne dass man über den Sinn der Aussagen nachdenken muss.
Auch Computer arbeiten mit Aussagenlogik: Zum einen kennt die kleinste Informationseinheit, das Bit, auch nur zwei Werte. Zum anderen lassen sich einfache aussagenlogische Kalküle sehr kostengünstig in Computerschaltungen umsetzen, die beliebige Berechnungen mit Bits anstellen können.

Eishörnchen

Die Eisdiele „Al Goritmo“ bietet diese vier Eissorten an:

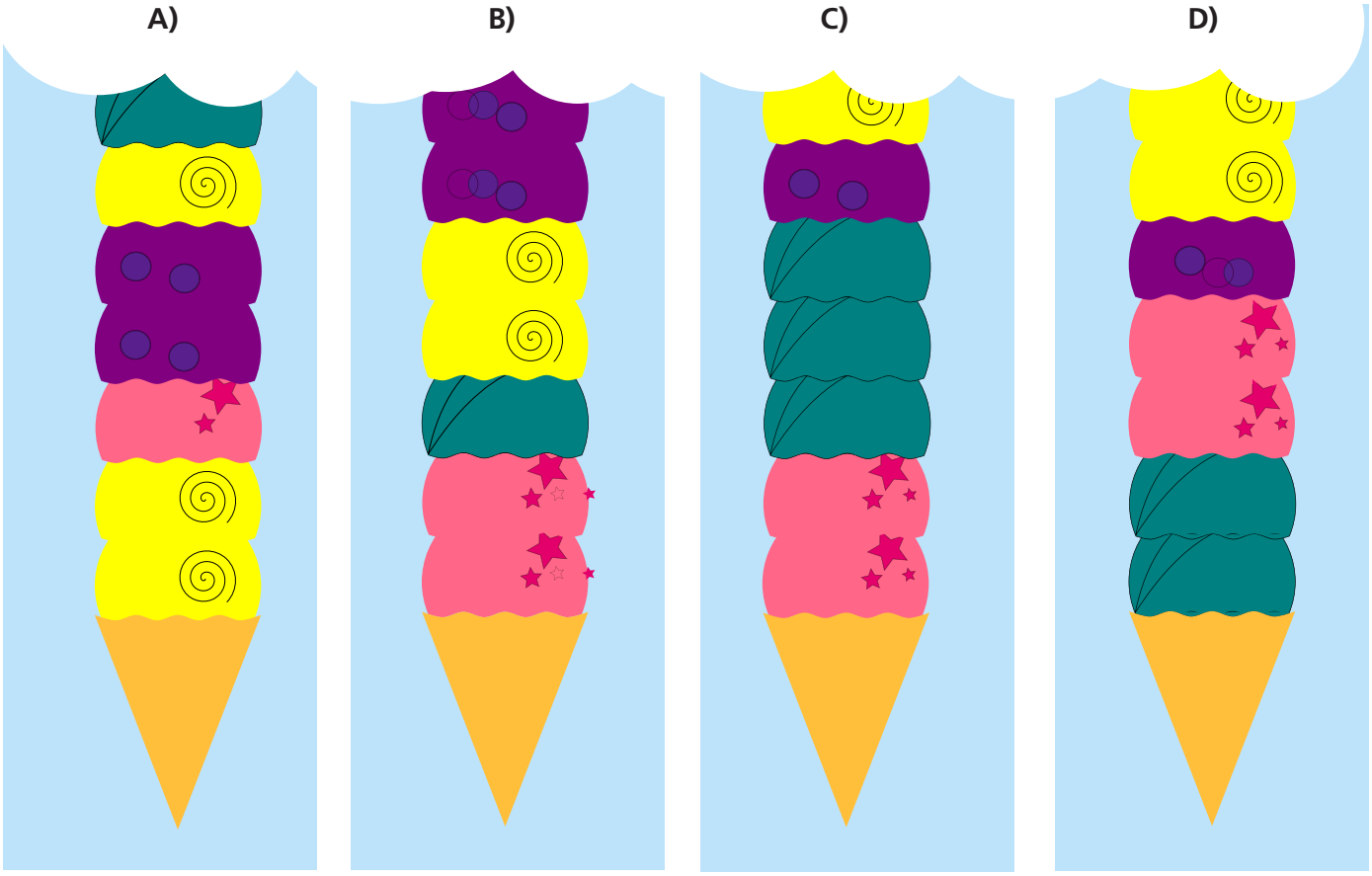


Bei Al Goritmo werden die Eishörnchen streng nach dieser Vorschrift hergestellt:

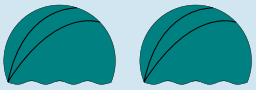
Schritt 0: Nimm ein leeres Hörnchen.
Schritt 1: Wähle zufällig eine Eissorte und gib zwei Kugeln dieser Eissorte hinzu.
Schritt 2: Gib eine Kugel mit irgendeiner anderen Eissorte als in Schritt 1 hinzu.
Schritt 3: Wenn die gewünschte Kugelzahl erreicht ist, höre auf. Ansonsten mache mit Schritt 1 weiter.

Unten siehst du einige Eishörnchen mit ihren unteren Kugeln.

Welches Eishörnchen stammt mit Sicherheit NICHT von Al Goritmo?



Antwort D ist richtig:
Das ist das einzige Eishörnchen, das ganz eindeutig nicht nach den Anweisungen hergestellt worden ist. Es beginnt korrekt mit zwei Kugeln der gleichen Eissorte

 (Schritt 1),

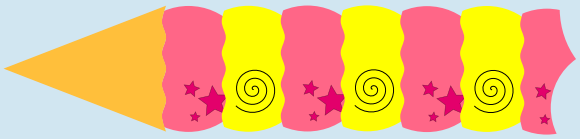
gefolgt von einer Kugel einer anderen Eissorte:

 (Schritt 2).

Dann jedoch kommen zwei Kugeln unterschiedlicher Sorten, obwohl nun wieder Schritt 1 der Vorschrift an der Reihe ist und damit zwei Kugeln der gleichen Sorte kommen müssten.

Die Antworten A, B und C sind nicht richtig. Diese Eishörnchen wurden nach der Vorschrift von Al Goritmo hergestellt, zumindest soweit man es erkennen kann: Es folgen wiederholt Gruppen von zwei gleichen Kugeln und einer anderen Kugel aufeinander.

Das ist Informatik!
Muster in Eishörnchen, Texten oder Bildern können durch Anweisungsfolgen erzeugt werden. Informatikerinnen und Informatiker entwickeln Computerprogramme, mit denen Muster und Abweichungen von Mustern erkannt werden.
Manchmal entstehen Muster durch Wiederholung von Anordnungen. Ein Beispiel:

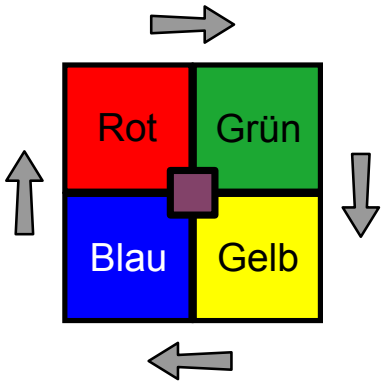


ist ein einfaches Muster, das durch Wiederholung von  entstanden ist. Dieses Muster ist leicht zu erkennen. In der Aufgabe ist die Sache schwieriger, weil die Vorschrift der Eisdiele auch Zufallsentscheidungen enthält.
Im Grunde genommen kann man niemals ganz sicher sein, ob eine Reihenfolge durch Zufall oder durch eine Folge von Anweisungen erzeugt worden ist. Bei den Beispielen dieser Aufgabe konnten wir nur bei einem Eis sagen, dass es sicher *nicht* den Anweisungen entsprach und deshalb nicht von Al Goritmo stammen konnte. Man kann aber aufgrund der Zusammensetzung des Eishörchens niemals sicher entscheiden, ob es von Al Goritmo stammt: Auch bei einer anderen Eisdiele könnte ein Hörnchen erstellt werden, das zufällig der Vorschrift von Al Goritmo entspricht.

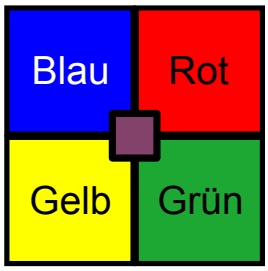
Farbenspiel

Simon hat ein Spiel mit vier Farben.

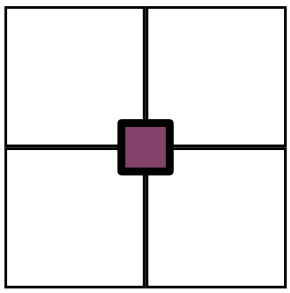
Wenn Simon den Knopf drückt,
bewegen sich die Quadrate um eins weiter,
wie die Pfeile zeigen.

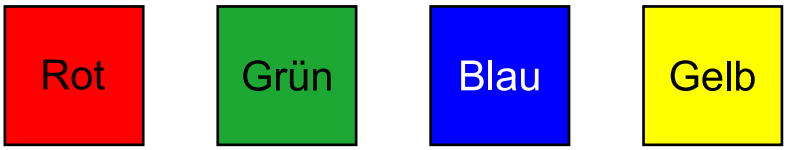


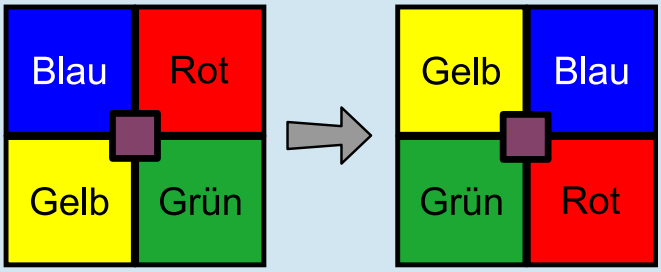
Simon hat den Knopf einmal gedrückt.
Jetzt liegen die Farben so:



Simon drückt den Knopf noch einmal.
Wo liegen danach die Farben?
Ziehe die Farben auf den richtigen Platz.







So ist es richtig:
Das Bild zeigt, wohin sich die Quadrate bewegen, wenn Simon den Knopf noch einmal drückt. Danach liegen die Farben so wie im Bild rechts.

Das ist Informatik!
Wer Computer programmiert, möchte genau wissen, welche Auswirkungen das fertige Programm am Ende hat. Dazu muss insbesondere geklärt sein, welche Auswirkungen jede einzelne Anweisung des Programms hat. Es ist aber gar nicht so leicht, diese Auswirkungen geschickt zu beschreiben. Bei dieser Biberaufgabe kann zum Beispiel für jede Farbanordnung beschrieben werden, in welche Anordnung sie übergeht, wenn Simon den Knopf drückt. Für das Beispiel in der Aufgabenstellung könnte das so lauten: „Blau/Rot/Grün/Gelb“ (beginnend unten links; im Uhrzeigersinn) geht über in „Gelb/Blau/Rot/Grün“. Das klingt kompliziert, und es ist geschickter, nach Gesetzmäßigkeiten zu suchen und diese zu beschreiben: Durch Knopfdruck rückt jede jede Farbe im Uhrzeigersinn um genau ein Feld weiter. Daraus lässt sich leichter die Auswirkung einer Reihe von Knopfdrücken festlegen: Zweimaliges Drücken verschiebt jede Farbe in die jeweils gegenüberliegende Ecke, viermaliges Drücken verändert die Farbanordnung nicht.

Ferienhaus 29

Milo arbeitet in einer Ferienhaus-Siedlung.
Einige Ferienhäuser haben noch keine Nummern.
Eines davon soll die neue Nummer 29 bekommen. Das soll Milo erledigen.

Milo startet bei Haus 50. Von einem Haus mit Nummer geht er

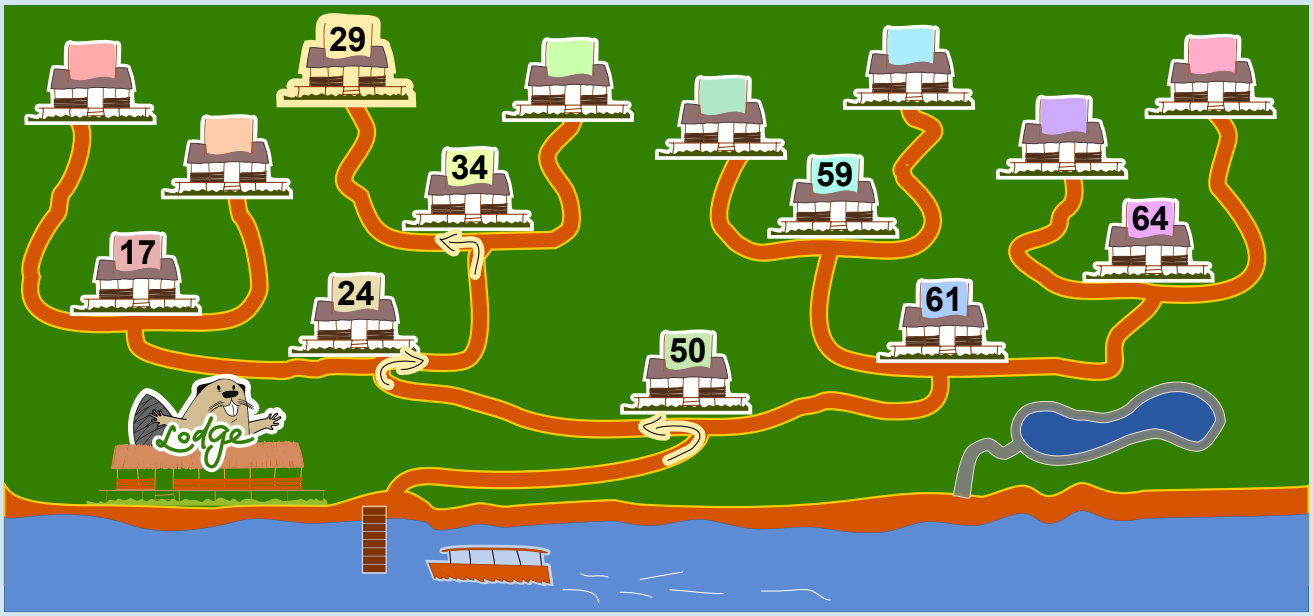
- nach links, wenn dessen Nummer größer ist als die neue Nummer.
- nach rechts, wenn dessen Nummer kleiner ist als die neue Nummer.

Sobald er zu einem Haus ohne Nummer gelangt, bekommt dieses die neue Nummer 29.

Welches Ferienhaus bekommt die Nummer 29?

Klicke auf das richtige Haus.

So ist es richtig:



Die Nummer 50 des ersten Ferienhauses ist größer als die neue Ferienhausnummer 29; also muss Milo als erstes nach links gehen. Als nächstes erreicht er das Haus mit der Nummer 24. Da 24 kleiner 29, geht Milo nun nach rechts. Die Nummer 34 des nächsten Hauses ist wieder größer als die 29; deshalb geht er von dort nach links. Nun gelangt er zu einem Haus ohne Nummer. Dieses Ferienhaus bekommt die Nummer 29.

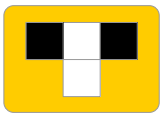
Das ist Informatik!
Die Ferienhäuser in dieser Biberaufgabe sind wie in einem binären Suchbaum angeordnet und nummeriert. Diese Datenstruktur wird in der Informatik häufig genutzt. Mit einem binären Suchbaum kann man gespeicherte Daten schnell wiederfinden.
Ein binärer Suchbaum ist so aufgebaut: Er beginnt mit einer Verzweigung. An jeder Verzweigung („Knoten“) ist ein „Element“ gespeichert. Von jedem Knoten führen maximal zwei Wege („Kanten“) zu weiteren Knoten. Beim Speichern neuer Elemente wird bei jedem Knoten stets der linke Weg eingeschlagen, wenn das neue Element einen kleineren Wert hat als das am Knoten selbst gespeicherte Element, sonst der rechte Weg. Das neue Element wird als Nachfolger des ersten Knotens gespeichert, von dem aus noch weniger als zwei Kanten ausgehen.
Bei der Suche nach einem Element kann man dann an jedem Knoten leicht entscheiden, welchen Weg man einschlagen muss. Wenn der binäre Suchbaum „balanciert“ ist, muss man nach einem Schritt nur noch jeweils ungefähr die Hälfte der Kreuzungen durchsuchen. Das führt dazu, dass 1000 Elemente in nur 10 Schritten durchsucht werden können, 1.000.000 Elemente in 20 Schritten oder 1.000.000.000 Elemente in 30 Schritten (also bei n Elementen in $\log_2 n$ Schritten).

Fliesenmuster

Tina möchte schwarze und weiße Fliesen legen, auf einer Fläche von 31 mal 16 Fliesen.

Tina liebt es, Fliesen nach festen Regeln zu verlegen.
Für jedes mögliche Muster von drei Fliesen in einer Reihe legt sie fest,
ob die Fliese in der Mitte darunter weiß oder schwarz sein soll.

Ein Beispiel:



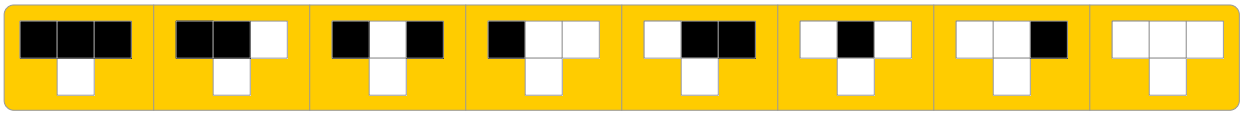
Wenn eine schwarze, eine weiße und eine schwarze Fliese nebeneinander liegen, soll die Fliese in der Mitte darunter weiß sein.

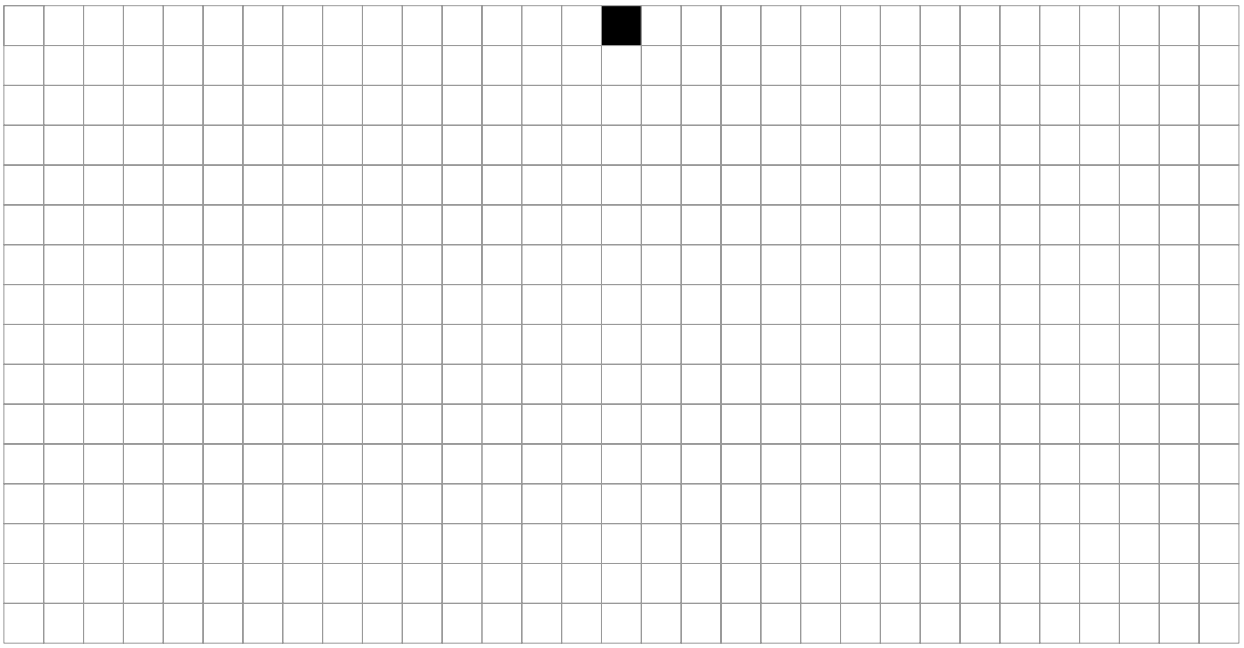
Damit sie ihre Regeln auch am Rand der Fläche anwenden kann,
tut Tina so, als lägen außen um die Fläche weiße Fliesen.

In die oberste Reihe legt Tina genau eine schwarze Fliese in die Mitte.
In der untersten Reihe sollen sich schwarze und weiße Fliesen schön abwechseln:

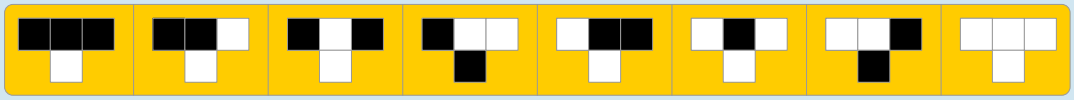
Wie müssen Tinas Regeln aussehen, damit das klappt?

Klicke in die unteren Fliesen der acht Regeln, um sie auf weiß oder schwarz zu setzen.
Es wird sofort angezeigt, wie eine Regeländerung sich auswirkt.

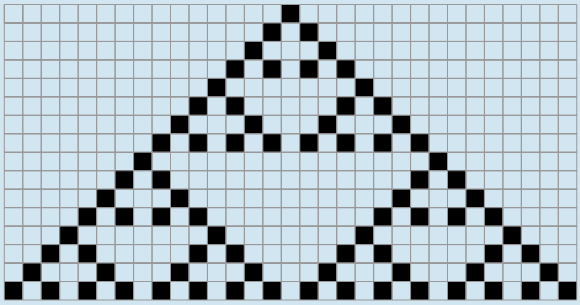




So ist es richtig:
Hier ist die einfachste Regelmenge (bei der die wenigsten unteren Fliesen auf schwarz gesetzt sind), die in der untersten Reihe das gewünschte Muster erzeugt:



Mit dieser Regelmenge wird die Fläche so belegt:



Bei dieser Biberaufgabe gibt es mehrere richtige Antworten. Wir beschreiben eine Regelmenge durch die Farbwahl der unteren Fliesen und kodieren weiß mit 0 und schwarz mit 1. Die oben genannte Regelmenge wird dann mit 00010010 bezeichnet.

Hier sind alle 17 Regelmengen, die zu dem gewünschten Muster führen:
00010010, 00011010, 00110010, 00111010, 01010010, 01011010, 01110010, 01111010, 10010010, 10011010, 10110010, 10110011, 10111010, 11010010, 11011010, 11110010, 11111010
Weil es so viele richtige Antworten gibt, hat man gute Chancen, eine davon durch Ausprobieren zu finden. Wer genauer überlegt, wird sich zuerst die Regeln 4, 6 und 7 (von links gezählt) anschauen. Nur diese Regeln sind auf die einzelne schwarze Fliese in der ersten Reihe anwendbar. Bei mindestens einer dieser Regeln muss eine untere Fliese auf schwarz gesetzt werden, sonst sind in der nächsten Reihe alle Fliesen weiß, und das gewünschte Muster lässt sich nicht mehr erreichen. Bei Regel 6 wiederum darf die untere Fliese nicht auf schwarz gesetzt werden: Sonst wäre in jeder Reihe die mittlere Fliese schwarz, aber das ist im gewünschten Muster nicht der Fall. So kommt man schnell darauf, dass es genügt, in den Regeln 4 und 7 die untere Fliese auf schwarz zu setzen.

Das ist Informatik!
Für das Verständnis der Möglichkeiten und Grenzen der Informatik spielen Automaten eine entscheidende Rolle – aber nicht etwa Kaffeeautomaten, wie Spötter behaupten mögen. In der Informatik sind Automaten formale Modelle von Berechnungsprozessen. Es gibt sehr viele verschiedene Arten von Automaten, alle mit unterschiedlichen Eigenschaften. Einer der berühmtesten Automaten ist die Turingmaschine, die ein umfassendes Modell aller berechenbaren Funktionen darstellt. Wie alle Automaten, liest sie nach und nach die „Buchstaben“ (allgemeiner sagt man: Zeichen) eines Wortes ein und verändert in jedem Schritt ihren Zustand. Die Turingmaschine kann dabei selbst wieder Zeichen auf ein endloses Band schreiben.
Etwas anders funktionieren die sogenannten zellulären Automaten. In dieser Biberaufgabe geht es um elementare zelluläre Automaten, die nur aus einer Reihe von Zellen bestehen, mit je zwei möglichen Zuständen. In jedem Schritt verändert sich jede Zelle abhängig von ihrem Zustand und dem Zustand ihrer Nachbarzellen. Tinas Regeln sind also zusammen die Zustandsveränderungsregel eines elementaren zellulären Automaten. Da für alle acht möglichen Zustände einer Zelle und ihrer beiden Nachbarn je zwei Nachfolgezustände möglich sind, gibt es insgesamt $2^8 = 256$ solcher Regeln. Sie können, wie oben gezeigt, als Binärzahl beschrieben werden und sind unter dem dezimalen Wert dieser Binärzahl bekannt. Die oben beschriebene Lösung dieser Biberaufgabe, 00010010, kennt man also als Regel 18. Elementare zelluläre Automaten erscheinen sehr einfach. Dennoch konnte bewiesen werden, dass Regel 110 eine Turingmaschine simulieren kann und damit letztlich auch ein Modell für beliebige Berechnungen darstellt.

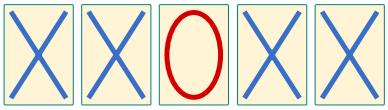
Flugzeug finden

Jana und Robin spielen mit ihrem Modellflugzeug. Sie lassen es von einem Hügel aus starten und weit weg im Gras landen. Dann geht Robin das Flugzeug holen.
Aber das Gras ist hoch, und nur vom Hügel aus kann man das Flugzeug sehen.

Jana zeigt vom Hügel aus, wohin Robin laufen muss, um das Flugzeug zu finden.
Dazu haben sie zwei Schilder mitgebracht und diesen Code vereinbart:

links	rechts	vor	zurück
X	X O	X O X	O X O

Leider gibt es ein Problem mit diesem Code. Ein Beispiel:

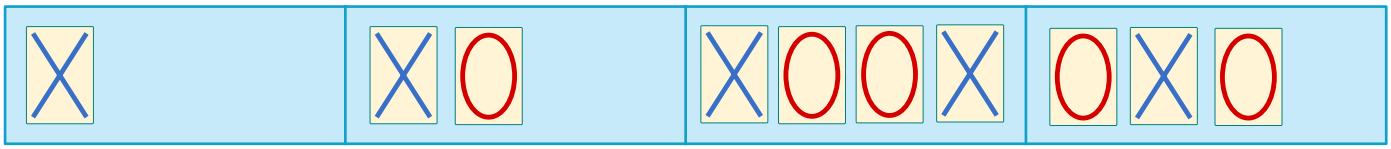


Diese Schilderfolge kann bedeuten: links, vor, links
Sie kann aber auch bedeuten: links, rechts, links, links

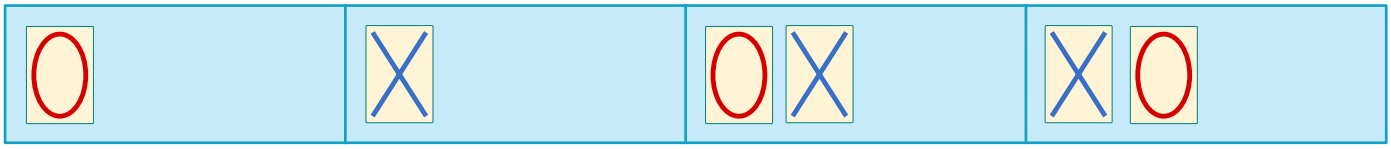
Jana und Robin überlegen sich deshalb einen neuen Code.
Damit kann es keine Schilderfolge geben, die mehrere Bedeutungen hat.

Welcher ist der neue Code?

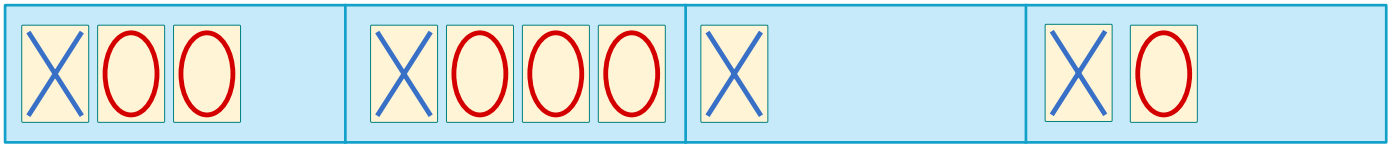
A)



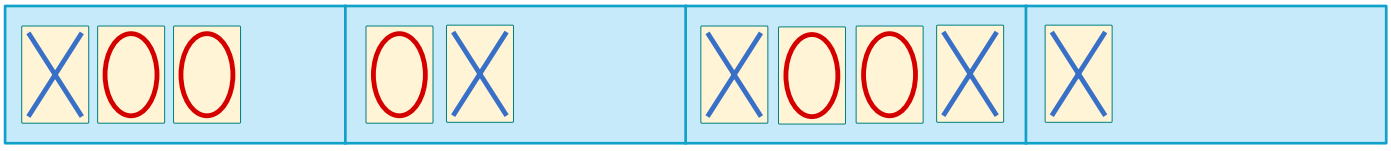
B)



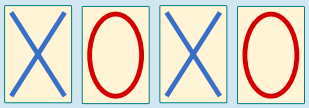
C)



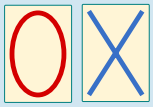
D)



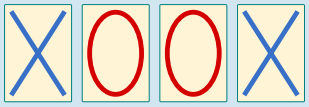
Die richtige Antwort ist C.
Die Antworten A, B und D sind falsch. Für diese Codes kann man Schilderfolgen angeben, die mehrere Bedeutungen haben:



Antwort A:
Diese Schilderfolge kann „rechts, rechts“, aber auch „links, zurück“ bedeuten.



Antwort B:
Diese Schilderfolge kann „vor“, aber auch „links, rechts“ bedeuten.

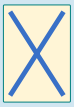
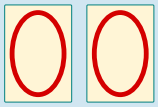
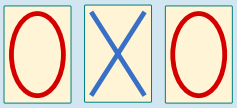
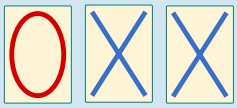


Antwort D:
Diese Schilderfolge kann „vor“, aber auch „links, zurück“ bedeuten.

Der Code von Antwort C hat eine besondere Eigenschaft: Jedes Codewort, also jede Schilderfolge, die eine der vier Richtungen bedeutet, beginnt mit einem einzigen Schild X, auf das 0, 1, 2 oder 3 Schilder mit einem O folgen. Allein die Anzahl der Schilder mit O bestimmt, um welches Codewort es sich handelt. Deshalb kann jedes Codewort eindeutig durch eine der Ziffern 0, 1, 2 oder 3 ersetzt werden. Jede Schilderfolge entspricht also einer Folge dieser Ziffern. Jede Ziffer bedeutet genau eine Richtung; mehrere Bedeutungen kann eine solche Ziffernfolge – und damit die entsprechende Schilderfolge – nicht haben.

Das ist Informatik!
Wenn Computer sich gegenseitig Nachrichten senden, dann werden diese in eine Folge von Signalen oder Zeichen übersetzt. Die spezielle Art und Weise, wie diese Übersetzung erfolgt, wird als Code bezeichnet. Von Binär-Code spricht man, wenn nur zwei verschiedene Zeichen vorkommen können.
Die einzelnen übersetzten Nachrichten bezeichnet man als Codewörter.
Vom Empfänger müssen diese Codewörter wieder richtig decodiert werden. Dabei ergibt sich das Problem, zwei aufeinanderfolgende Codewörter voneinander zu trennen. Eine Möglichkeit wäre, einen Blockcode zu verwenden, dessen Wörter alle gleich lang sind. Dann ergeben sich die Grenzen zwischen Codewörtern automatisch. Blockcodes führen allerdings zu im Schnitt unnötig langen Nachrichten.
In dieser Biberaufgabe kommt ein Code mit Wörtern unterschiedlicher Länge zum Einsatz.
Im Lösungscode oben wurde der einfache Trick angewendet, dass jedes Codewort mit einem X beginnt und dann nur noch aus weiteren O. Damit wirkt das X wie ein Trennzeichen zwischen den Codewörtern. Das ist eine gute Idee für Codes mit wenigen Codewörtern, hat aber den Nachteil, dass bei dieser Lösung die Länge der Codewörter mit jedem weiteren benötigten Codewort wächst.
Eine elegantere Lösung ist ein sogenannter Präfixcode oder präfixfreier Code. In einem Präfixcode beginnt kein Codewort mit der vollständigen Folge von Zeichen eines anderen Codewortes. Deshalb kann man auch ohne Trennzeichen unterschiedlich lange Codewörter erkennen und dekodieren. Und als Codes mit variabler Wortlänge eignen sich Präfixcodes gut zur Verdichtung (Komprimierung) von Information.

Dies ist ein Präfixcode für Jana und Robin:

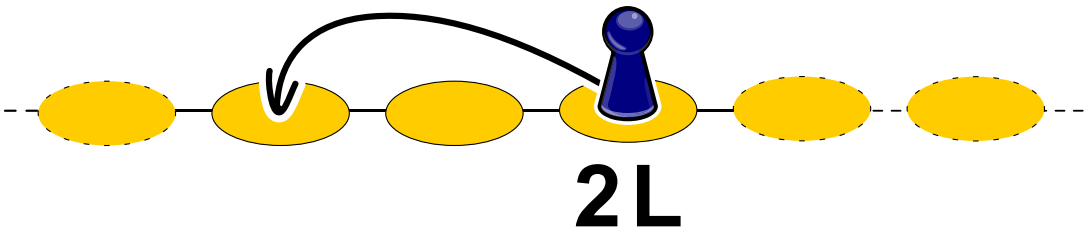
links	rechts	vor	zurück
			

Genau einmal

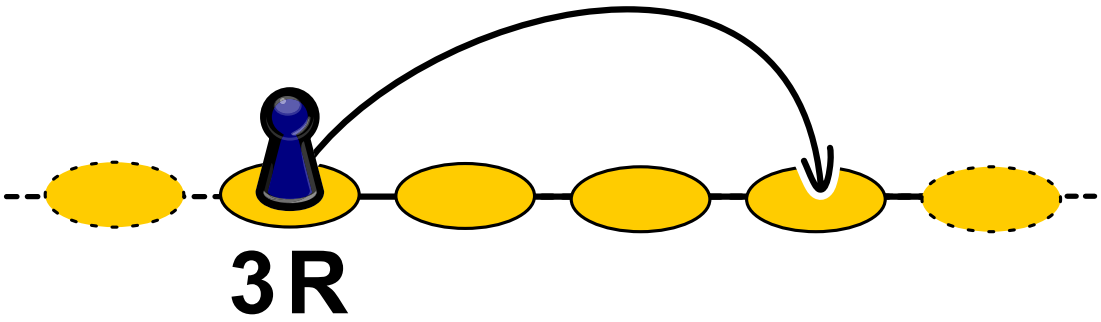
In einem Spiel zieht die Spielfigur über eine Reihe von Feldern.
Bei jedem Feld steht eine Anweisung.
Sie sagt, wohin die Figur von dem Feld aus ziehen soll.

Es gibt drei Arten von Anweisungen:

• *n* L bedeutet: Ziehe *n* Felder nach links. Ein Beispiel:



• *n* R bedeutet: Ziehe *n* Felder nach rechts. Ein Beispiel:

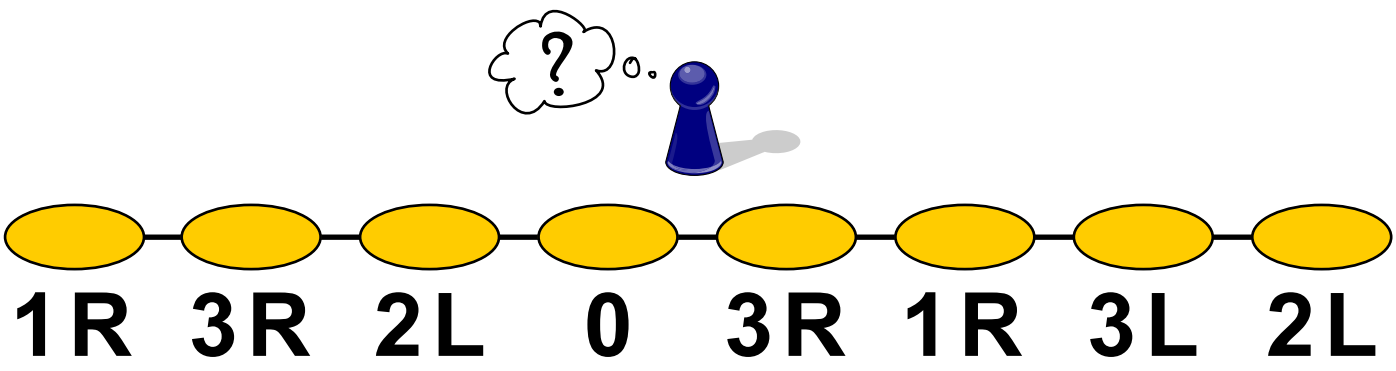


• 0 bedeutet: Bleib stehen.

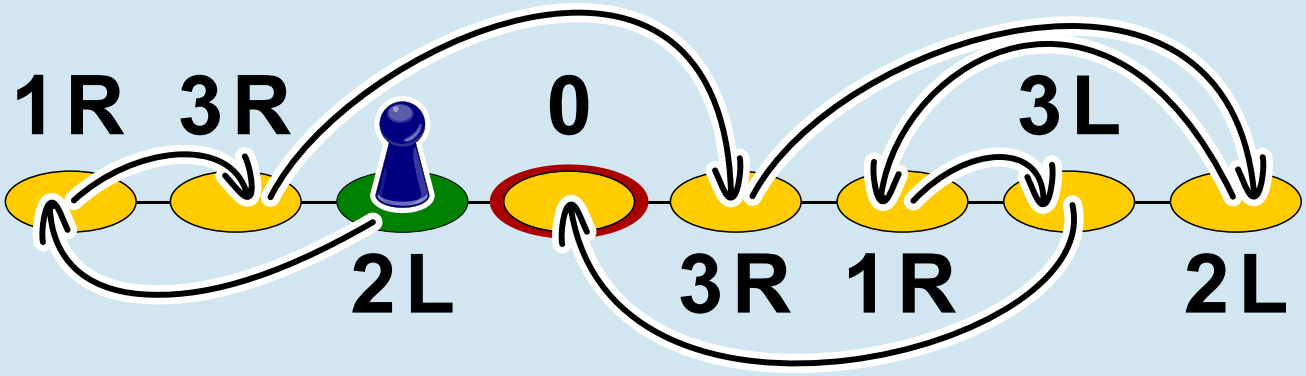
Unten siehst du das gesamte Spiel mit allen Anweisungen.
Die Spielfigur soll so ziehen, dass sie am Ende auf jedem Feld genau einmal gewesen ist.

Auf welchem Feld muss die Spielfigur starten?

Klicke auf ein Feld, um es als Startfeld auszuwählen.



So ist es richtig:
Die Spielfigur muss auf dem dritten Feld von links („2L“) starten. Von dort aus zieht sie über alle Felder bis zum Feld mit der Anweisung „0“ und bleibt stehen.



Um das richtige Startfeld zu bestimmen, kann man beim letzten Feld „0“ beginnen und von dort aus rückwärts den Weg über die Felder suchen. Ausgehend vom Feld „0“ wird für jedes Feld dessen „Herkunftsfeld“ bestimmt, von dem aus die Spielfigur auf das Feld zieht, und dann wird von diesem Herkunftsfeld aus weiter gesucht. Das Herkunftsfeld von Feld „0“ ist das zweite Feld von rechts („3L“). Dessen Herkunftsfeld wiederum ist das dritte Feld von rechts („1R“); von dort aus geht es weiter zum Feld ganz rechts („2L“) usw. bis zum dritten Feld von links („2L“). Dieses Feld hat kein Herkunftsfeld, und glücklicherweise hat die Rückwärtssuche damit auch alle Felder erreicht.

Dass ein Weg über alle Felder möglich ist, liegt an den passend gewählten Anweisungen des Spiels. Stünde z. B. auf dem zweiten Feld von links die Anweisung „1L“ statt „3R“, würde die Spielfigur endlos zwischen den beiden linken Feldern hin und her springen und weder stehen bleiben noch auf ihrem Weg über alle Felder ziehen können.

Das ist Informatik!
In der Biberaufgabe „Licht an!“ auf Seite 44 geht es um das Problem, einen Weg von einem Startzustand zu einem bestimmten Zielzustand zu suchen. Auch bei anderen Problemen wird gesucht, etwa wenn sozusagen aus dem Nichts die (beste) Lösung eines Problems konstruiert werden muss – wie der Arbeitsplan in der Biberaufgabe „Biber-Arbeit“ auf Seite 15. Der leere Arbeitsplan ist dann der Startzustand der Suche, und Zielzustand ist jeder Arbeitsplan, bei dem alle Aufgaben verteilt wurden.

Suchen spielt eine große Rolle in der Informatik. Auch in dieser Biberaufgabe sucht die Spielfigur einen Weg vom Start zum Ziel. Allerdings ist nur das Ziel bekannt. Zum Glück kann jeder Schritt von einem Zustand zum nächsten umgedreht werden: wir können für jedes Feld das Herkunftsfeld bestimmen, wie oben beschrieben. Deshalb kann man die Suche umdrehen und vom Ziel aus den Start suchen. In dieser Biberaufgabe kann von jedem Zustand aus nur ein anderer Zustand erreicht werden. Deshalb ist die Suche über die acht Felder schon nach sieben Schritten zu Ende. Bei den meisten Problemen, die in der Informatik mit Suchmethoden angegangen werden, gibt es von jedem Zustand aus mehrere Möglichkeiten, einen anderen Zustand zu erreichen oder zu konstruieren. Dann ist der „Suchraum“ keine Linie wie in dieser Aufgabe, sondern ein stark verzweigter Baum mit möglicherweise sehr vielen Zuständen. Um auch große Suchräume einigermaßen schnell durchsuchen zu können, werden gute algorithmische Ideen benötigt.

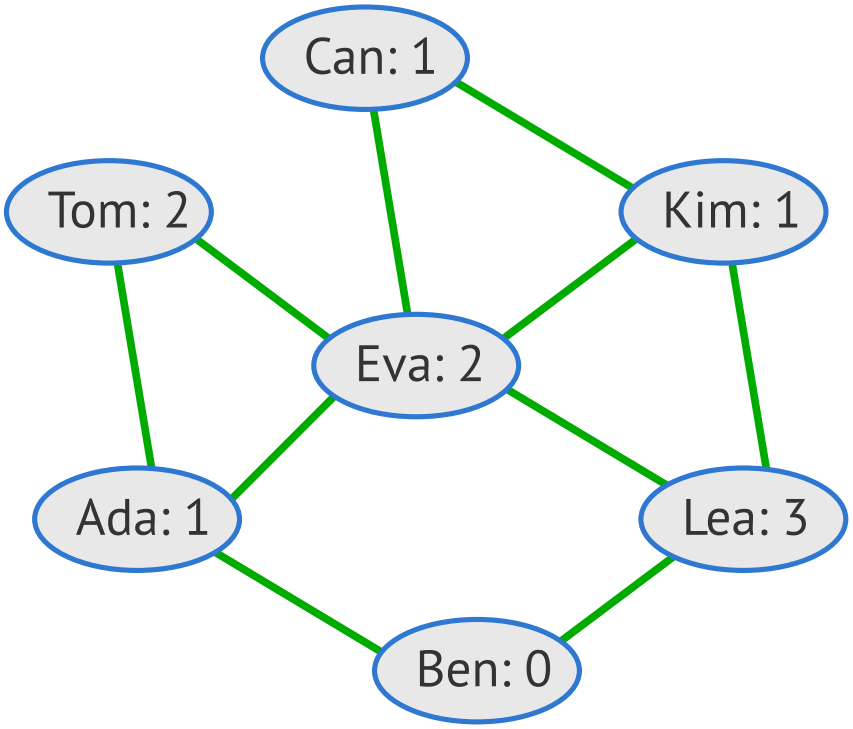
Geschenke

Das Bild zeigt die Freundschaften zwischen den Kindern in einem Haus.
Eine Linie zwischen zwei Namen bedeutet: Diese Kinder sind ein Freundespaar.

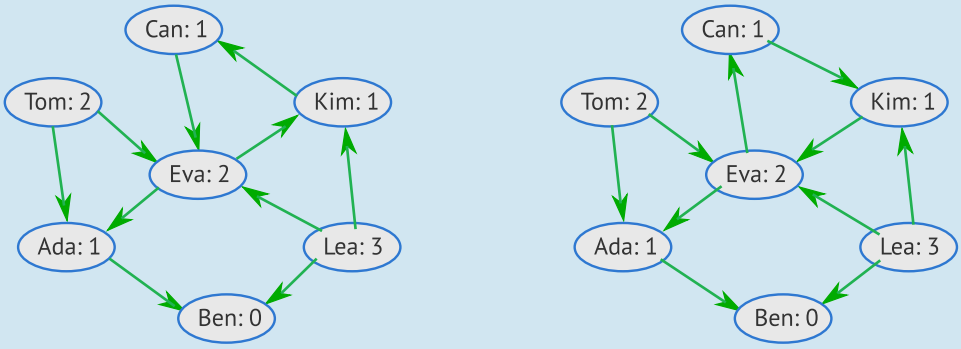
Die Hausbewohner planen ein Kinderfest mit Geschenken.
Bei allen Freundespaaren soll ein Kind dem anderen Kind ein Geschenk besorgen.
Im Bild steht hinter jedem Namen, wie viele Geschenke das Kind besorgen kann.

Entscheide für jedes Freundespaar, wer das Geschenk besorgt.
Kein Kind soll mehr Geschenke besorgen müssen, als es besorgen kann.

Klicke dazu auf die Linie zwischen den beiden Namen. Dann erscheint ein Pfeil.
Klicke noch einmal, um die Richtung des Pfeils zu ändern.
Der Pfeil zeigt die „Schenkrichtung“ an:
Zeigt er z. B. von Lea zu Eva, bedeutet das, dass Lea ein Geschenk für Eva besorgt.



So ist es richtig:
Es gibt zwei Möglichkeiten, wie man für alle Freundespaare die „Schenkrichtung“ angeben kann, ohne dass ein Kind mehr Geschenke besorgen muss, als es besorgen kann.



Wir beginnen mit Ben: Er kann kein Geschenk besorgen und erhält deshalb je ein Geschenk von seinen Freundinnen Ada und Lea. Damit hat Ada ihr Geschenk vergeben und erhält selbst je ein Geschenk von Eva und Tom. Für die Freundespaare Ben&Ada, Ben&Lea, Ada&Eva und Tom&Eva sind die Schenkrichtungen also klar.
Auf den ersten Blick scheint es nun viele weitere Möglichkeiten zu geben. Wer genau hinsieht, erkennt aber, dass die Freunde Can und Kim jeder nur ein Geschenk besorgen können. Deshalb gibt es bei diesem Freundespaar nur zwei Möglichkeiten:

1. Kim besorgt ein Geschenk für Can: Dann hat Kim sein Geschenk vergeben und erhält ein Geschenk von seiner Freundin Eva. Damit hat auch Eva ihre Geschenke vergeben und erhält Geschenke von ihren Freunden Can (der ja sein Geschenk noch frei hat), Tom und Lea. Lea, die auch schon für Ben ein Geschenk besorgen muss, hat aber immer noch ein Geschenk frei und besorgt es für Kim. Damit sind die Schenkrichtungen für alle Freundespaare festgelegt, und zwar so wie im Bild links.
2. Can besorgt ein Geschenk für Kim: Dann hat Can sein Geschenk vergeben und erhält ein Geschenk von seiner Freundin Eva. In diesem Fall erhält Eva Geschenke von Kim, Tom und Lea; Lea besorgt auch hier ihr letztes Geschenk für Kim. Damit sind die Schenkrichtungen so festgelegt wie im Bild rechts.

Egal welche der beiden möglichen Schenkrichtungen für das Freundespaar Can&Kim gewählt wird, ergeben sich alle weiteren Schenkrichtungen daraus eindeutig. Es gibt also insgesamt nur zwei verschiedene Möglichkeiten, alle Schenkrichtungen festzulegen.

Das ist Informatik!
Die Freundschaften zwischen den Kindern bilden ein Netzwerk aus Knoten (die Kinder) und Verbindungen (die Freundschaftsbeziehungen selbst). Das erinnert an die so genannten „sozialen Netzwerke“ mit ihren Millionen von Benutzern. Es gibt aber einen wichtigen Punkt, in dem sich diese Systeme unterscheiden können: In manchen gibt es wechselseitige „Freundschaften“, in denen die Verbindungen keine Richtung haben, so wie in dieser Biberaufgabe. In anderen gibt es „Anhänger“ (Englisch: „follower“), und dann haben die Verbindungen eine Richtung: Wenn du einer berühmten anderen Nutzerin „folgst“, muss sie nicht auch dir folgen.
In dieser Biberaufgabe sollen aus allen Freundschafts-Verbindungen (ohne Richtung) nun „Schenk-Verbindungen“ (mit Richtung) werden. Entlang dieser Richtung soll über jede Verbindung dann ein Geschenk „fließen“. Aber die „Schenk-Kapazitäten“ der Kinder setzen der Wahl der Richtungen Grenzen. Das ähnelt einem in der Informatik bekannten Problem: In einem Netzwerk (etwa aus Wasserkanälen oder Kommunikationsverbindungen) soll so viel (Wasser oder Daten) über die Verbindungen fließen wie möglich, aber die Kapazitäten der Verbindungen sind begrenzt. Nur sind bei einem solchen „Netzwerk-Fluss“ Grenzen für die Verbindungen angegeben, bei dieser Biberaufgabe hingegen für die Knoten. Es ist aber leicht möglich, das Problem der Schenkrichtungen in ein „Fluss-Problem“ umzuwandeln. Und dann lässt sich das „Schenkrichtungs-Problem“ mit den effizienten Algorithmen lösen, welche die Informatik für Fluss-Probleme kennt. Dieser Trick ist übrigens in der Informatik sehr beliebt: Ein Problem wird in ein anderes Problem umgewandelt (die Informatik sagt auch: reduziert), für das Lösungen gut bekannt sind.

Karten drehen

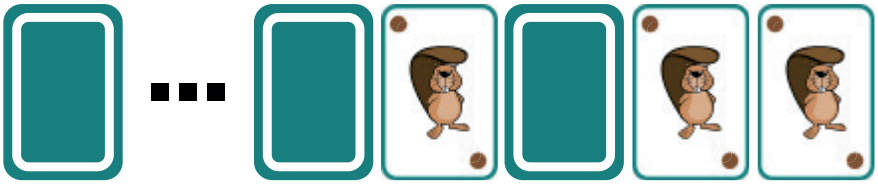
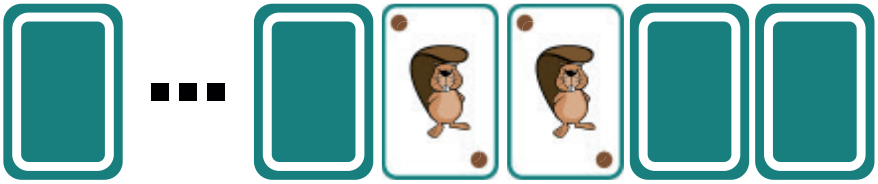
Jemand schenkt dir einen Satz gleicher Karten. Die Karten sehen so aus:

Aufgedeckt:  Verdeckt: 

Mit diesen Karten kannst du „Drehen“ spielen.
Eine Reihe von Karten liegt vor dir.
In einem Spielzug gehst du diese Karten von rechts nach links so durch:

- Ist die aktuelle Karte verdeckt, decke sie auf.
  Damit ist der Spielzug beendet, die übrigen Karten bleiben unverändert.
- Ist die aktuelle Karte aufgedeckt, drehe sie um.

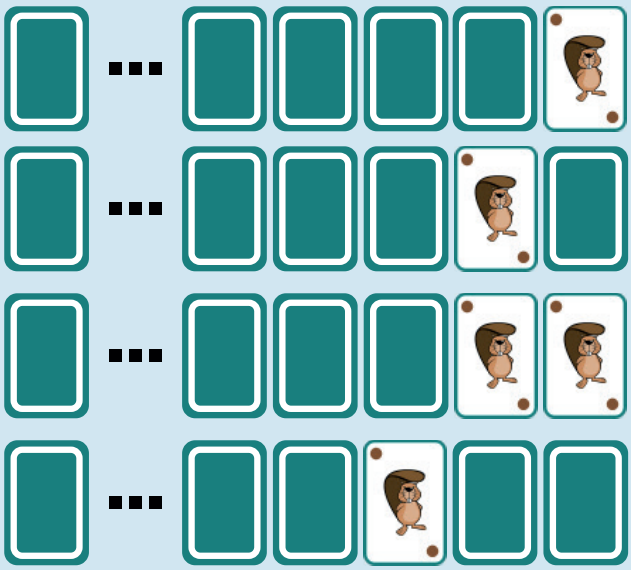
Das Bild zeigt, wie sich die Karten in einem Spielzug verändern können.

Vorher:	
Nachher:	

Diesmal beginnt das Spiel mit 16 verdeckten Karten.

Wie viele Karten sind nach 16 Spielzügen aufgedeckt?

1 ist die richtige Antwort.
Woran erinnern uns die Karten? Eine Reihe von gleichen Karten, die entweder verdeckt oder aufgedeckt sein können, stellt eine Binärzahl dar. Binärzahlen bestehen nur aus den Ziffern 0 und 1. Eine verdeckte Karte stellt die Ziffer 0 dar und eine aufgedeckte Karte die Ziffer 1.
Analog zum üblichen Zehnersystem gibt jede Stelle einer Binärzahl an, ob die passende Zweierpotenz in den Wert der Zahl einzurechnen ist oder nicht. Ist z. B. die dritte Stelle (von rechts) einer Binärzahl mit einer 1 besetzt, ist die dritte Zweierpotenz zum Wert der Zahl zu addieren – also 2^2, denn wir fangen bei 2^0 an.
Binärzahlen werden auf diese Weise um 1 hochgezählt. Man beginnt mit der Ziffer ganz rechts:
Ist die aktuelle Ziffer eine 0, mache daraus eine 1.
Damit ist die gesamte Zahl um 1 hochgezählt.
Ist die aktuelle Ziffer eine 1, mache daraus eine 0 und gehe zur nächsten Ziffer nach links (Übertrag).
Das entspricht genau einem Spielzug im Spiel „Drehen“. Ein Spielzug erhöht also den Wert der Binärzahl, die durch die Kartenreihe dargestellt wird, um 1. Die verdeckten Karten zu Beginn stellen die Binärzahl dar, die nur aus 0en besteht, also den Wert 0 hat.
Das Bild zeigt die ersten vier Spielzüge, die also von 1 bis 4 zählen. Man kann sehen, dass bei den Zahlen 1, 2 und 4 (also den Zweierpotenzen 2^0, 2^1 und 2^2) genau eine Karte aufgedeckt ist: Bei einer Binärzahl, die eine Zweierpotenz als Wert hat, ist nur an der Stelle, die dieser Zweierpotenz entspricht, eine 1.



Nach 16 Spielzügen erhalten wir also die Darstellung einer Binärzahl mit Wert 16. Da $16 = 2^4$, hat diese Binärzahl genau an der fünften Stelle von rechts eine 1 und sonst nur 0en: 0…010000. In der Darstellung mit den Karten ist also genau eine Karte aufgedeckt.

Das ist Informatik!
Die kleinste Speichereinheit eines Computers kann nur zwei Werte unterscheiden: AN und AUS, WAHR oder FALSCH, 0 oder 1. Alle Daten, die in einem Computer gespeichert und verarbeitet werden, können wir also als Reihen binärer Ziffern, letztlich also als Binärzahlen sehen. Deshalb haben Operationen auf Binärzahlen für die Informatik eine große Bedeutung.
Seitdem es Computer gibt, werden darin solche Operationen möglichst effizient realisiert. Es gibt Operationen, die zwei Binärzahlen miteinander verknüpfen, wie etwa die Rechenoperationen Addition oder Multiplikation. Es gibt aber auch Operationen, die eine einzelne Binärzahl verändern, etwa das Verschieben aller Ziffern um eine Position nach links (was einer Multiplikation mit 2 entspricht) – oder eben das Hochzählen, also die Addition um 1, die in dieser Biberaufgabe die zentrale Rolle gespielt hat.

Klang-Code

Margaret möchte Namen so kodieren, dass ähnliche klingende Namen gleich kodiert werden. Sie hat dazu dieses Verfahren entwickelt:

1. Behalte den ersten Buchstaben, streiche aus den restlichen Buchstaben alle H und W.
2. Ersetze die verbleibenden Buchstaben (bis auf den ersten) so durch Ziffern:

Ersetze	durch
A, E, I, O, U oder Y	0
B, F, P oder V	1
C, G, J, K, Q, S, X oder Z	2
D oder T	3
L	4
M oder N	5
R	6

3. Wenn nun mehrere gleiche Ziffern hintereinander stehen, behalte nur eine davon.
4. Entferne alle Nullen (0).
5. Hat das bisherige Ergebnis noch nicht die Form Buchstabe, Ziffer, Ziffer, Ziffer, streiche überschüssige Ziffern oder füge so viele Nullen (0) an wie nötig.

Hier sind einige Beispiele dafür, wie das Verfahren Namen kodiert:
ROBERT ⟶ R163, RUPERTUS ⟶ R163, KNUTH ⟶ K530, GAUSS ⟶ G200

Wie kodiert das Verfahren den Namen HILBERT ?
A) I416 **B)** B540 **C)** H041 **D)** H416

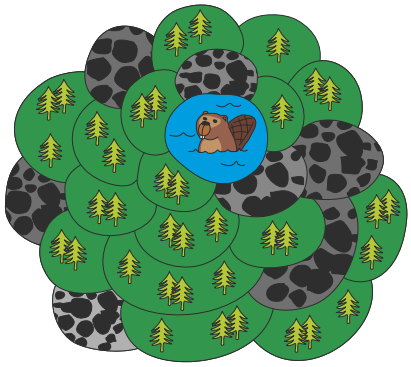
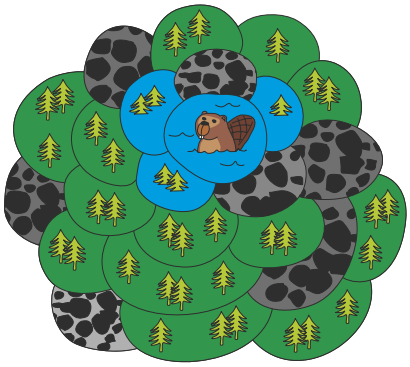
Antwort D ist richtig:
Wir gehen das Verfahren schrittweise durch, mit der Eingabe „HILBERT“, und zeigen die Ergebnisse der einzelnen Schritte.
Schritt 1: HILBERT, Schritt 2: H041063, Schritt 3: H041063, Schritt 4: H4163, Schritt 5: H416
Da das Verfahren ein eindeutiges Ergebnis liefert, sind die anderen Antworten nicht richtig.

Das ist Informatik!
Das in dieser Biberaufgabe beschriebene Soundex-Verfahren (genauer gesagt: der amerikanische Soundex) wurde bereits vor 100 Jahren von Robert C. Russell und Margaret King Odell entwickelt und patentiert. Es wurde dazu verwendet, ähnlich klingende Wörter und insbesondere ähnlich klingende Namen zu finden. Das funktioniert, weil die Gruppen von Buchstaben, die durch die gleiche Ziffer kodiert werden, ähnlich klingen: B, F, P und V sind Lippenlaute, C, G, J, K, Q, S, X und Z sind Gaumenlaute und Zischlaute, D und T sind Zahnlaute, L ist ein langer Fließlaut, M und N sind Nasenlaute und R ist ein kurzer Fließlaut.
Das Verfahren ist sehr einfach und liefert relativ gute Resultate. Deshalb wird es häufig zur phonetischen Suche, also zur Suche nach ähnlich klingenden Wörtern verwendet. Es ist auch als Standard in wichtigen Datenbanksystemen eingebaut. Einige der obigen Beispiele stammen von Donald Knuth, einem der bedeutendsten Informatiker des 20. Jahrhunderts. Er arbeitet schon seit Jahrzehnten an seinem Buch „The Art of Computer Programming“. Im Band 3 „Sorting And Searching“ beschreibt er das Soundex-Verfahren. In dieser Biberaufgabe wird allerdings eine Variante vorgestellt, die in Schritt 3 den Rückgriff auf das Eingabewort vermeidet.

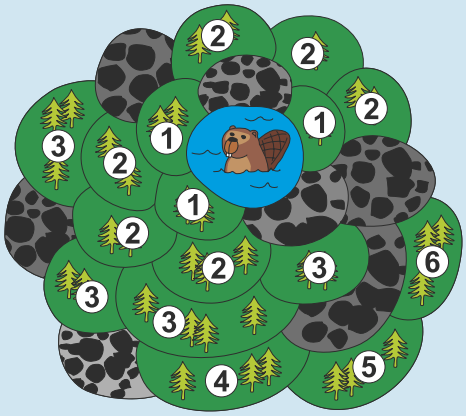
Kleiner Teich

In einem Tal liegt ein kleiner Teich. Er ist umgeben von Landstücken mit Wald oder mit Felsen. Im Teich leben einige Biber.
Eines Tages wird den Bibern der Teich zu klein, und nun fluten sie den Wald.
An jedem Tag fluten sie alle Waldstücke, die an bereits geflutete Waldstücke angrenzen.
Nach einem Tag sind drei Waldstücke geflutet.

Nach wie vielen Tagen sind alle Waldstücke geflutet?

Teich im Tal	Nach einem Tag
	

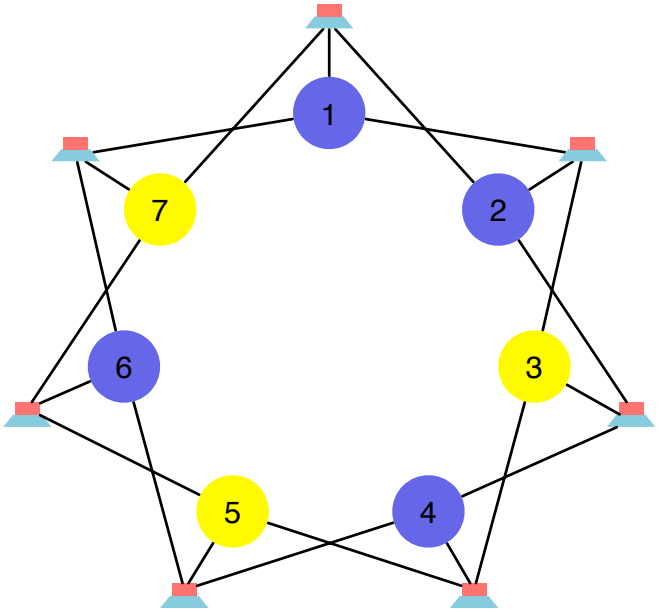
6 ist die richtige Antwort.
Das Bild zeigt für jedes Waldstück, nach wie vielen Tagen es geflutet ist: Die Waldstücke, die an den See angrenzen, sind nach einem Tag geflutet und deshalb mit der Zahl 1 markiert. Die Waldstücke, die an diese Felder angrenzen, sind mit der Zahl 2 markiert; sie sind nach zwei Tagen geflutet. Und so weiter. 6 ist die größte Zahl, mit der ein Waldstück auf diese Weise markiert wird. Nach sechs Tagen ist also dieses Waldstück geflutet – und damit alle Waldstücke im Tal.



Das ist Informatik!
In dieser Biberaufgabe fluten die Biber ein zusammenhängendes Waldgebiet, das aus einzelnen, benachbarten Waldstücken besteht – und dem ursprünglichen Teich. Das Gebiet ist zusammenhängend, weil jedes Waldstück in diesem Gebiet zu mindestens einem anderen Waldstück benachbart ist. So kann man von jedem Stück des Gebietes jedes andere Stück über einen Weg durch das Gebiet selbst erreichen.
Gibt es zusammenhängende Gebiete auch außerhalb des Biber-Teich-Tals? Aber natürlich! Ein einfarbiger Bereich in einem (Computer-)Bild ist ein zusammenhängendes Gebiet gleichfarbiger Pixel. Eine Gruppe von Jugendlichen, in denen jeder mindestens mit einem anderen Jugendlichen der Gruppe befreundet ist, ist auch ein zusammenhängendes Gebiet, wenn man die Freundschaftsbeziehung als Nachbarschaft betrachtet.
Die Informatik kennt Methoden, zusammenhängende Gebiete zu ermitteln, etwa die Breitensuche oder die Tiefensuche. Mit Hilfe dieser Methoden können z. B. Bereiche in Bildern umgefärbt oder Gruppierungen in sozialen Netzwerken ermittelt werden.

https://de.wikipedia.org/wiki/Zusammenhang_(Graphentheorie)
https://de.wikipedia.org/wiki/Breitensuche

Licht an!

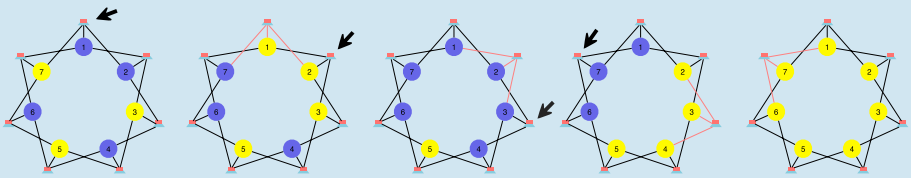


Es gibt sieben Leuchten und sieben Schalter. Jeder Schalter ist mit jeweils drei Leuchten verbunden. Wenn du einen Schalter drückst, werden die mit ihm verbundenen Leuchten umgeschaltet: von aus nach an bzw. von an nach aus.

Schalte alle Leuchten an!

Klicke dazu auf die Schalter.

So ist es richtig:
Die Schalter bei den Leuchten 1, 2, 3 und 7 können nacheinander gedrückt werden, um alle Leuchten anzuschalten.



Bei der Suche nach der richtigen Schalterreihenfolge hilft es, vom Ziel her zu denken. Damit beim letzten Schalterdruck die mit dem Schalter verbundenen Leuchten alle angehen, müssen sie vorher aus sein. Deshalb ist es wichtig, zunächst Lampen auszuschalten. Mit Hilfe der Schalter bei den Leuchten 1 und 2 kann man die Leuchten 7 sowie 1, 2 und 3 ausschalten. Dann ist nur noch Leuchte 5 an, und sechs nebeneinander liegende Leuchten sind aus. Die kann man in zwei Dreiergruppen einteilen, die man mit dem Schalter jeweils in der Mitte (also mit den Schaltern 3 und 7) anschalten kann.
Für das Leuchten-Schalter-Netz dieser Biberaufgabe kann man beobachten:
1. Die Reihenfolge, in der man zwei Schalter drückt, ist egal; das Resultat ist gleich.
2. Wenn man einen Schalter zweimal drückt, ändert sich nichts; das zweite Drücken macht das Resultat des ersten Drückens rückgängig.

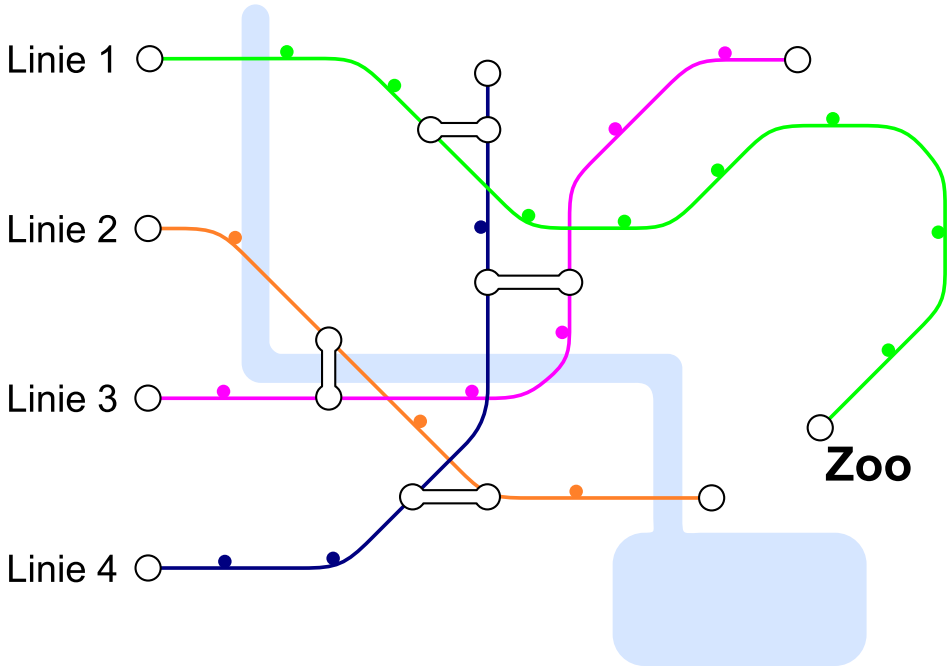
Damit stellt sich letztlich nur die Frage, welcher Schalter gedrückt werden muss und welcher nicht. Bei sieben Schaltern gibt es immerhin 2^7 (= 128) verschiedene Möglichkeiten. Eine davon kann man aber von vornherein ausschließen: Keinen Schalter zu drücken, bringt nichts – es sei denn, alle Leuchten sind ohnehin schon an.

Das ist Informatik!
In dieser Biberaufgabe ist ein Weg gesucht, nämlich von einem Start-Zustand (Leuchten 3, 5 und 7 sind angeschaltet) zu einem Ziel-Zustand (alle Leuchten sind angeschaltet). Ein „Schritt“ auf diesem Weg besteht darin, die gegebene Operation (das Drücken eines Schalters, mit den beschriebenen Auswirkungen) anzuwenden und damit zu einem nächsten Zustand zu kommen. Der gesuchte Weg führt also durch die Menge (die Informatik sagt auch: Raum) der möglichen Zustände vom Start zum Ziel. Die Suche eines Wegs durch einen Zustandsraum zu einem Ziel-Zustand ist in der Informatik ein bekanntes Problem und wurde besonders intensiv im Bereich der „Künstlichen Intelligenz“ untersucht. Wenn zum Beispiel ein Staubsauger-Roboter plant, welche Operationen er ausführen muss, um eine Wohnung zu säubern und am Ende wieder an seiner Ladestation zu stehen, kann er eine Zustandsraum-Suche durchführen. Ist der Ziel-Zustand klar beschrieben, kann die Suche deutlich beschleunigt werden, indem nicht nur ein Weg vom Start zum Ziel, sondern auch ein Weg vom Ziel zum Start gesucht wird. Diese bidirektionale Suche ist erfolgreich, wenn beide Suchen den gleichen Zustand erreichen.

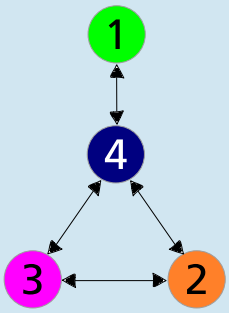
Liniennetz

In Biberstadt gibt es vier Bahnlinien. Eine Linie fährt zum Zoo.
An diesen Stationen ⊂⊃ kann man von einer Linie zu einer anderen wechseln.
Johannes fährt zum Zoo. Er muss nur einmal die Linie wechseln.

Mit welcher Linie fährt er los? Klicke links die richtige Linie an.



So ist es richtig:



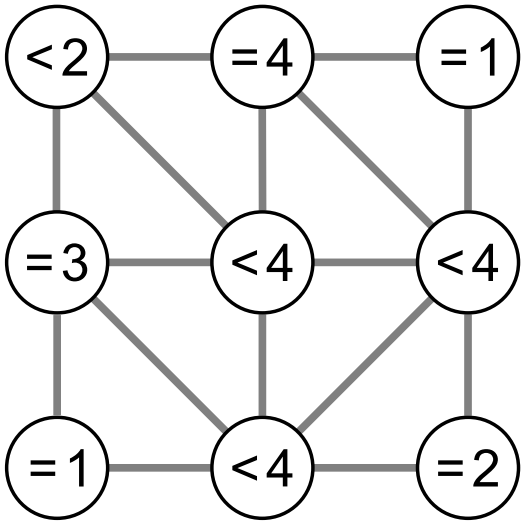
Johannes fährt mit Linie 4 los. Nur Linie 4 hat eine Station, an der man direkt zu Linie 1 wechseln kann. An dieser Station wechselt Johannes zu Linie 1 und fährt damit weiter zum Zoo. Das Bild stellt übersichtlich dar, von welcher Linie aus man zu welcher anderen Linie wechseln kann. Die Kreise enthalten die Liniennummern. Ein Doppelpfeil zwischen zwei „Linien-Kreisen" bedeutet, dass man zwischen diesen Linien wechseln kann. So ist sofort sichtbar, dass Johannes nur von Linie 4 aus direkt zu Linie 1 wechseln kann.

Das ist Informatik!
Während man beim Bahnliniennetz von Biberstadt noch recht genau hinschauen muss, kann man die Antwort auf die Frage dieser Biberaufgabe aus dem obigen Bild unmittelbar ablesen. Aber warum geht es damit so viel einfacher? Weil beim Erstellen der „einfachen" Abbildung gleich zwei wichtige Methoden der Informatik angewandt wurden. Zum einen die *Abstraktion:* Das neue Bild enthält nur noch die für die Beantwortung der Frage wichtige Information, von welcher Linie man zu welcher anderen wechseln kann. Zum anderen die *Reduktion,* also die Umwandlung eines Problems in ein anderes, vielleicht einfacheres Problem: Die Frage nach dem einmaligen Umstieg in die Linie zum Zoo wird zur Frage, welcher Kreis im Bild mit dem Kreis 1 direkt per Doppelpfeil verbunden ist. Wenn ein neues Problem in ein bekanntes Problem umgewandelt, also reduziert werden kann, lässt sich die Schwierigkeit des neuen Problems mit der des bekannten Problems vergleichen. Zum Beispiel ist das Problem, eine Zahl auf Teilbarkeit durch 5 zu prüfen, recht leicht zu lösen. Es lässt sich nämlich auf die Frage reduzieren, ob die letzte Ziffer der Zahl eine 0 oder eine 5 ist.

Nachbarn

Das Bild unten zeigt neun Kreise. Wenn zwei Kreise durch eine Linie verbunden sind, sind sie Nachbarn.

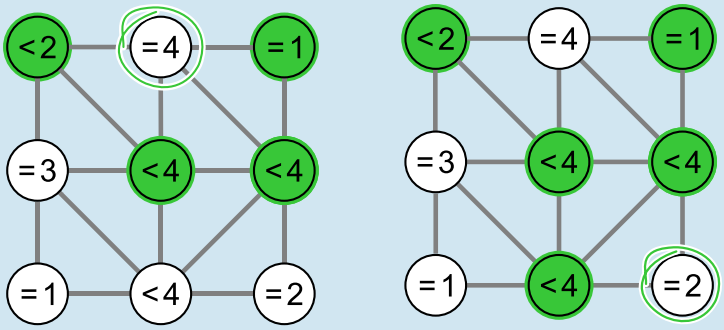
Du sollst einige Kreise auswählen. Dazu steht in jedem Kreis eine Bedingung. Sie sagt, wie viele Nachbarn des Kreises ausgewählt werden sollen.
Zum Beispiel bedeutet „= 3“, dass genau drei Nachbarn ausgewählt werden sollen.
„< 4“ bedeutet, dass weniger als vier Nachbarn ausgewählt werden sollen.



Wähle nun Kreise so aus, dass alle Bedingungen erfüllt sind.

Klicke einen Kreis an, um ihn auszuwählen; er wird dann grün gefärbt.

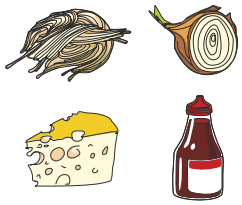
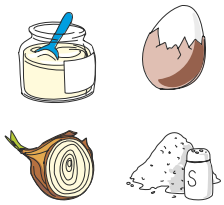
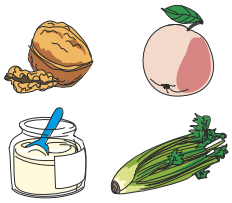
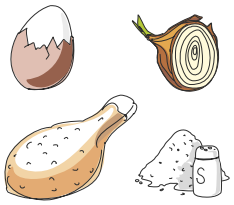
So ist es richtig:
Um die richtigen Kreise auszuwählen, kann man mit den Bedingungen anfangen, die keine Wahl zulassen. Die Bedingung im Kreis oben mitte lautet „=4“. Sie legt zwingend fest, dass alle vier Nachbarn ausgewählt werden müssen; siehe Bild links. In gleicher Weise legt die Bedingung „=2“ im Kreis rechts unten fest, dass alle zwei Nachbarn ausgewählt werden müssen; siehe Bild rechts.
Damit sind bereits die Bedingungen in allen neun Kreisen erfüllt. Und es ist nicht möglich, einen weiteren Kreis auszuwählen, ohne eine Bedingung zu verletzen.



Das ist Informatik!
Damit Computer tun, was Menschen wollen, müssen sie mit klaren Anweisungen versorgt werden. Deshalb ist in der Informatik Klarheit schon beim Nachdenken über diese Anweisungen erforderlich. Auch in dieser Biberaufgabe müssen einige Begriffe klar unterschieden werden: Eine Lösung der Aufgabe ist eine Auswahl der Kreise, die alle Bedingungen erfüllt. Nicht jede mögliche Auswahl ist auch eine Lösung, kann aber als Kandidat für eine Lösung bezeichnet werden.
Bei den meisten Problemen gibt es viele Wege, Lösungen zu finden. Fast immer ist es möglich, alle Kandidaten aufzuzählen und zu prüfen, ob sie eine Lösung darstellen. Diese Vorgehensweise wird in der Informatik als „brute force“ bezeichnet, was so viel heißt wie „rohe Kraft“. In dieser Biberaufgabe gibt es $2^9 = 512$ Kandidaten, denn bei der Auswahl von Kreisen gibt es für jeden der neun Kreise zwei Möglichkeiten: auswählen oder nicht auswählen. Für die Prüfung von 512 Kandidaten reicht die Zeit beim Informatik-Biber aber nicht; rohe Kräfte walten eben häufig sinnlos, wie schon Friedrich Schiller wusste. Ein systematisches Vorgehen, das die angegebenen Bedingungen (engl. auch „constraints“) ausnutzt, um den Kandidatenkreis möglichst schnell zu verringern, ist viel effizienter.

Passende Gerichte

Bela kann fünf verschiedene Gerichte zubereiten.
Das Bild zeigt die Gerichte mit ihren Zutaten.

Pasta	Eiersalat	Nusssalat	Hühnersuppe	Torte
				

Für eine Party will Bela zwei Gerichte zubereiten.
Diese beiden Gerichte sollen zusammen passen.
Bela meint: Zwei Gerichte passen zusammen, wenn mindestens zwei Zutaten gleich sind.

Welche zwei Gerichte passen zusammen?

A) Hühnersuppe und Pasta

B) Hühnersuppe und Nusssalat

C) Hühnersuppe und Eiersalat

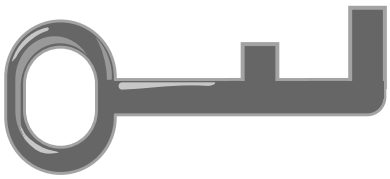
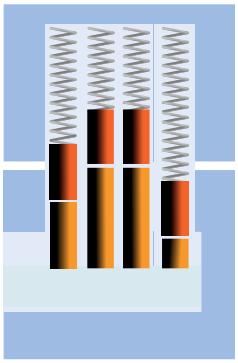
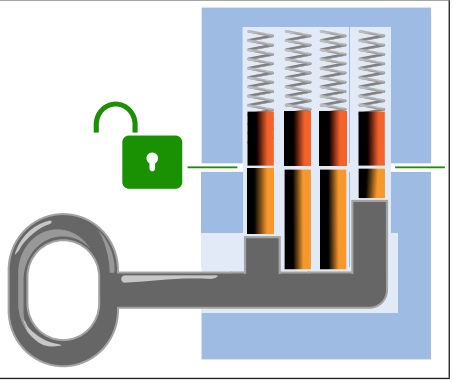
D) Nusssalat und Torte

Antwort C ist richtig:
Bei Hühnersuppe und Eiersalat sind drei Zutaten gleich: Ei, Zwiebel und Salz. Diese beiden Gerichte passen für Bela also zusammen.

Die anderen Antworten sind nicht richtig.
Antwort A: Bei Hühnersuppe und Pasta ist nur eine Zutat gleich: die Zwiebel.
Antwort B: Bei Hühnersuppe und Nusssalat gibt es gar keine gemeinsame Zutat.
Antwort D: Bei Nusssalat und Torte gibt es ebenfalls keine gemeinsame Zutat.

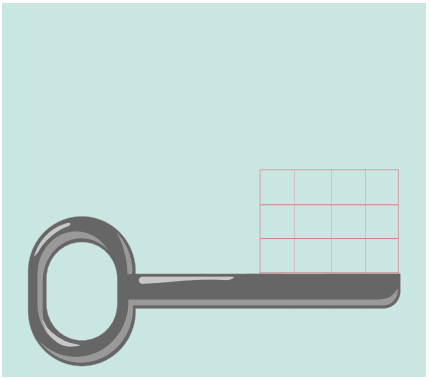
Das ist Informatik!
Um zu entscheiden, welche Gerichte zueinander passen, misst Bela, wie weit zwei Gerichte voneinander entfernt sind. Eine Entfernung bzw. einen Abstand kann man nämlich nicht nur in Metern, Yards, Seemeilen oder Li messen. Das Maß für den Abstand zwischen den Gerichten in dieser Biberaufgabe ist die Anzahl der gleichen Zutaten: Sind alle Zutaten gleich, haben die Gerichte keinen Abstand voneinander und sind gleich. Und je weniger gemeinsame Zutaten zwei Gerichte haben, desto weiter sind sie voneinander entfernt. Abstandsmaße – oder ihre Umkehrung, Ähnlichkeitsmaße – finden in der Informatik breite Verwendung. Ein Beispiel ist der Mustervergleich; dazu ist bei der nächsten Aufgabe („Passt der Schlüssel“, S. 48) mehr zu finden. Ein anderes Beispiel ist die Suche nach der besten Lösung eines Optimierungsproblems: Hier kann bei der Verbesserung von (noch nicht optimalen) Zwischenlösungen überlegt werden, möglichst große Verbesserungsschritte zu unternehmen; zur Bestimmung der Schrittweite wird dann ein Abstandsmaß benötigt.

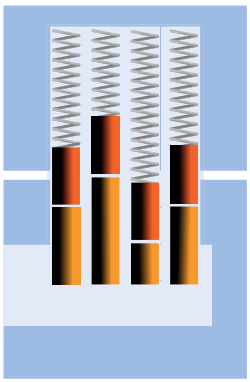
Passt der Schlüssel?

Dieser Schlüssel passt …	… in dieses Schloss.	Hier siehst du warum.
		

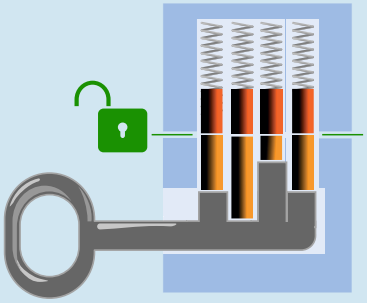
Hier ist ein anderes Schloss.

Klicke auf die Kästchen, bis der Schlüssel passt.



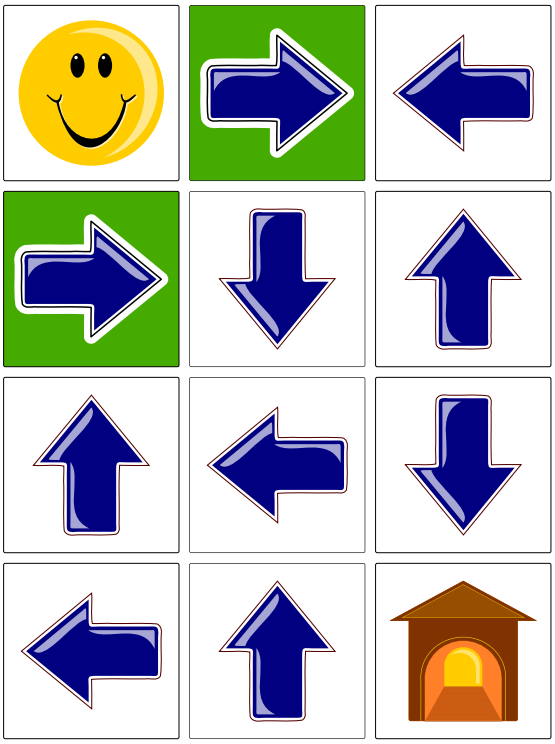


So ist es richtig:
Die Bartteile des Schlüssels müssen das Gegenstück zu den Stiften des Schlosses bilden. Ein Bartteil muss also umso länger sein, je kürzer der zugehörige Schlüsselstift ist.



Das ist Informatik!
Der Schlüssel passt, wenn jedes Teil seines Barts zum zugehörigen Teil im Schloss passt. Das durch die Bartteile gebildete Muster muss also zum Muster der Schlossteile passen.
Mustervergleich, auf Englisch „pattern matching“, ist ein wichtiger Bestandteil vieler Systeme und Methoden der Informatik. Jede Suchfunktion muss in der Lage sein, in Texten, Webseiten oder anderen Datenmengen die Stellen zu finden, die zum Suchmuster passen. Besonders nützlich und interessant sind Methoden zum Mustervergleich, bei denen das Muster nicht genau, sondern nur ungefähr passen muss. Dabei werden zwei Ansätze unterschieden: Entweder gibt der Benutzer direkt an (z. B. mit Hilfe so genannter regulärer Ausdrücke), welcher Spielraum beim Mustervergleich zugelassen ist. Das ist wie bei einem Generalschlüssel, der zu allen Schlössern eines Hauses, aber nicht zu Schlössern anderer Häuser passt. Oder der Mustervergleich ist grundsätzlich „unscharf“ und erklärt auch solche Muster als passend, die sich nur leicht voneinander unterscheiden. So können z. B. Internet-Suchmaschinen auch bei falsch geschriebenen Suchwörtern noch passende Webseiten finden. Auch eine am Klang von Wörtern orientierte phonetische Suche (siehe auch die Biberaufgabe „Klang-Code“ auf S. 42) ist unscharf.

Pfeil-Labyrinth



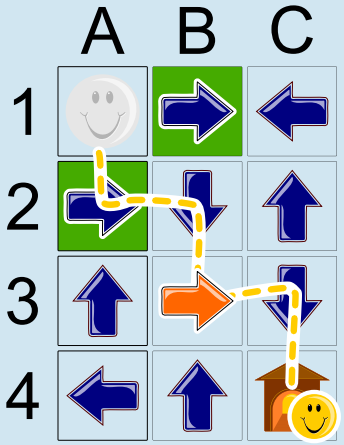
Das Smiley :-) möchte nach Hause.
Dazu muss es durch das Pfeil-Labyrinth.
Die grünen Felder sind die Eingänge.
Auf jedem Feld muss das Smiley
der Richtung des Pfeils folgen.

Aber so kann das Smiley nicht nach Hause finden!

**Ändere die Richtung nur eines Pfeils,
damit das Smiley nach Hause finden kann.**

Klicke mehrmals auf einen Pfeil,
um seine Richtung zu ändern.

So ist es richtig:
Der Weg des Smiley nach Hause ist zu erkennen: A1 – A2 – B2 – B3 – C3 – C4. Der Pfeil, dessen Richtung geändert wurde, um diesen Weg zu ermöglichen, ist orange dargestellt.



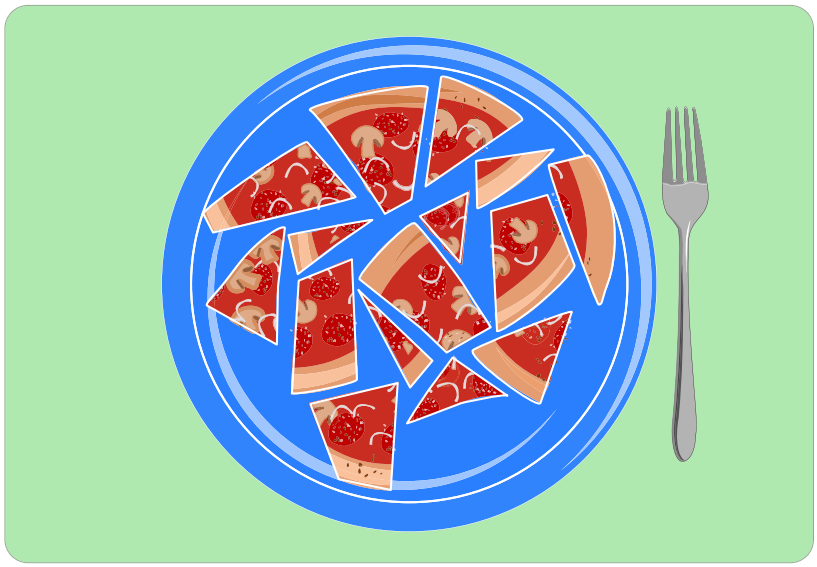
Man kann sogar beweisen, dass dies die einzig mögliche Lösung ist. Die Idee: Du beginnst am Ziel C4 und gehst rückwärts. Man könnte von zwei Feldern aus zum Ziel C4 gelangen: von B4 und von C3. Der Pfeil in B4 zeigt aber in die falsche Richtung und müsste geändert werden. Weil aber kein Pfeil zu B4 zeigt, müsste ein zweiter Pfeil geändert werden. Das ist aber nicht erlaubt. Folglich ist das Ziel nur über C3 erreichbar. Der Pfeil in diesem Feld zeigt bereits auf das Ziel.
Wie aber ist C3 zu erreichen? Wir gehen weiter rückwärts: C3 wäre von B3 oder C2 aus zu erreichen. Weil aber kein Pfeil zu C2 zeigt, müsste ein zweiter Pfeil geändert werden. Das ist aber nicht erlaubt. Folglich ist das Ziel nur über B3 erreichbar. Dazu ändern wir die Pfeilrichtung in B3 so, dass sie auf C3 zeigt.
Wie ist nun B3 zu erreichen? Von B3 aus können wir weiter rückwärts gehen, ohne Pfeile zu verändern, nämlich über B2 zum Eingang A2 zum Startfeld A1. Damit haben wir einen Weg für den Smiley gefunden. Für diesen Weg muss nur die Pfeilrichtung in B3 geändert werden.

Das ist Informatik!
Das Pfeil-Labyrinth in dieser Biberaufgabe ist so klein und übersichtlich, dass man die richtige Antwort mit ein wenig Überlegung und Ausprobieren finden kann. Wenn das Labyrinth aber sehr groß wäre, würden Menschen die Übersicht schnell verlieren. Da könnte ein Computer helfen, doch dem müsste ein systematisches Vorgehen einprogrammiert werden. Wie könnte das aussehen?
Nun, zum Beispiel genau so, wie beim obigen Beweis vorgegangen wurde: Vom Ziel aus schrittweise einen Weg zusammenstellen; bei jedem Schritt die Möglichkeiten merken, die zunächst nicht verfolgt werden; wenn es bei einer Möglichkeit nicht mehr weitergeht, dann zurücksetzen und die nächste verfolgen. Diese Art von Lösungssuche ist in der Informatik als „Backtracking“ bekannt (engl.: back = zurück; track = Spur): Wie im Märchen von Hänsel und Gretel wird bei der Suche eine Spur gelegt, entlang derer man zurückgehen kann.

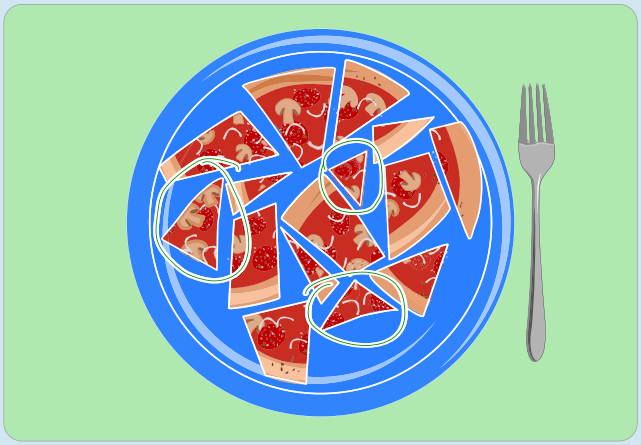
Pizza

Es gibt Pizza! Mutter hat für Lucy Stücke geschnitten.
Die Stücke ohne Kruste muss Lucy mit der Gabel essen.

Welche Stücke muss Lucy mit der Gabel essen?
Klicke auf die Stücke.



So ist es richtig:
Im Bild sind die Pizza-Stücke ohne Kruste eingekringelt.
Diese Stücke muss Lucy mit der Gabel essen.



Das ist Informatik!
Bei jedem Pizza-Stück muss Lucy fragen: Hat es eine Kruste oder nicht? Abhängig davon, wie die Antwort auf die Frage ausfällt, verhält Lucy sich unterschiedlich: Bei „ja“ isst sie das Stück so, wie sie Lust hat (also im Zweifel mit der Hand), bei „nein“ isst sie es mit der Gabel. Lucys Verhalten „verzweigt“ also abhängig davon, ob die Krusten-Bedingung für ein Pizza-Stück wahr ist oder nicht.

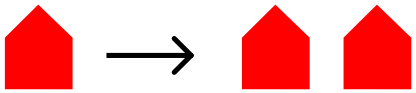
Solche bedingten Verzweigungen gehören zu den Grundbausteinen für Computerprogramme. Entscheidend ist, dass es für die zu prüfenden Bedingungen nur zwei Möglichkeiten gibt: Sie sind erfüllt bzw. wahr, oder sie sind nicht erfüllt bzw. falsch. Dann wird im Programm für eine oder für beide Möglichkeiten angegeben, wie sich der Computer zu verhalten hat. Ein „Programm“ für Lucys Pizza-Essen würde in einer Programmiersprache in etwa so beschrieben:

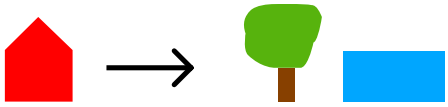
WENN das Stück eine Kruste hat
* DANN iss es so, wie du magst*
* SONST iss es mit der Gabel*

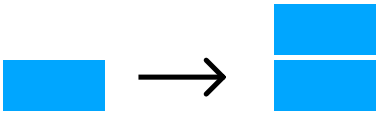
Planet Z

Die Leute auf Planet Z bauen ihre Städte immer auf die gleiche Weise.

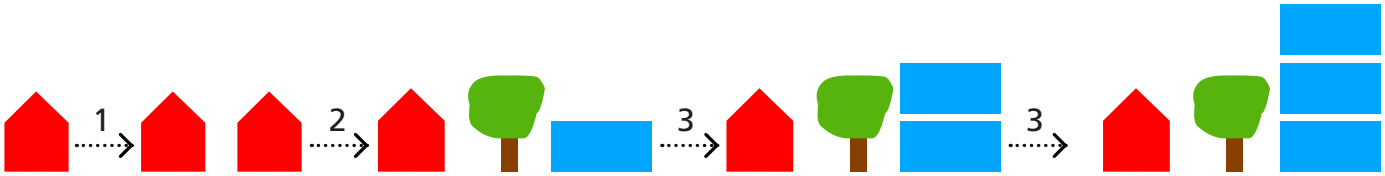
Sie beginnen mit einem Haus: 
Dann ersetzen Sie einzelne Objekte nach diesen drei Regeln:

1) 

2) 

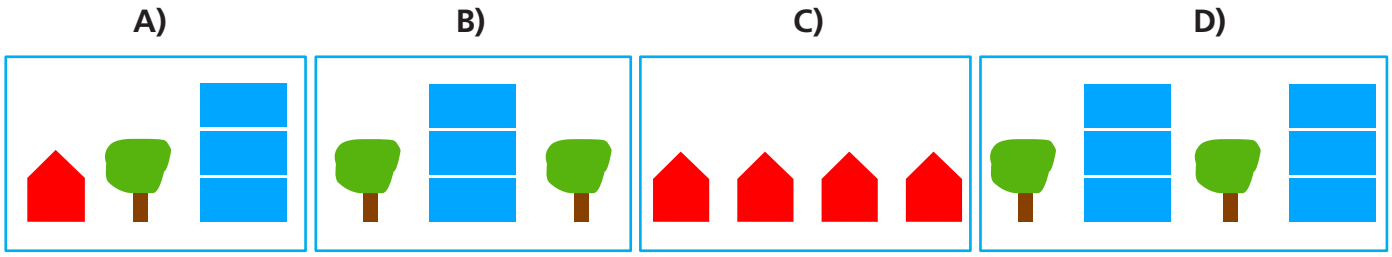
3) 

Ein Beispiel: Wenn man zuerst Regel 1, dann Regel 2 und dann zwei Mal Regel 3 anwendet, baut man eine Stadt wie ganz rechts im Bild:



Beachte, dass die Reihenfolge der Objekte nicht verändert wird.

Welche Stadt steht NICHT auf Planet Z?



Antwort B ist richtig:
Einen Baum kann man nur durch Anwendung von Regel 2 erhalten. Regel 2 besagt aber, dass rechts neben einem Baum immer ein Block stehen muss. In Stadt B ist rechts neben dem rechten Baum kein Block. Es gibt auch keine Regel, mit der man Bäume entfernen kann. Deshalb kann die Stadt B nicht auf dem Planeten Z stehen.
Stadt A ist die gleiche Stadt wie im Beispiel. Man kann sie also bauen, indem man die Regeln in dieser Reihenfolge anwendet: 1, 2, 3 und dann wieder 3.
Stadt C kann man bauen, indem man Regel 1 drei Mal anwendet.
Stadt D kann man so bauen: Zuerst wendet man Regel 1 an und erhält zwei Häuser. Nun wendet man Regel 2 auf jedes Haus an, und zuletzt wendet man Regel 3 auf die beiden einzelnen Blöcke und dann noch einmal auf zwei Blöcke an.

Das ist Informatik!
Auf Planet Z startet man ein Bauvorhaben mit einem Haus und wendet dann nach und nach die Ersetzungsregeln an. Auch bei Programmiersprachen wird mit Ersetzungsregeln, die von einem Startsymbol ausgehen, beschrieben, wie die Programme in dieser Sprache aufgebaut sind. Startsymbol und Ersetzungsregeln bilden zusammen die Grammatik einer Programmiersprache. Wie bei einer natürlichen Sprache gibt die Grammatik vor, ob etwas Geschriebenes korrekt ist.
In dieser Biberaufgabe war zu prüfen, ob eine Menge von Objekten der „Baugrammatik“ von Planet Z entspricht. Eine ähnliche Aufgabe hat ein Compiler; das ist eine Software, die in einer Programmiersprache geschriebenen Code in die vom Computer ausführbaren Maschinenbefehle übersetzt. Nur wenn der Compiler feststellen kann, dass der Programmcode gemäß der Grammatik der Programmiersprache korrekt ist, kann der Code übersetzt werden.
Wie die Bauregeln von Planet Z sind auch die Grammatiken von Programmiersprachen kontextfrei: In allen Ersetzungsregeln wird ein einzelnes Objekt durch andere Objekte ersetzt, ohne den Kontext (also das, was rechts und links von dem Objekt ist) zu beachten. Nur wenn eine Programmiersprache mit einer kontextfreien Grammatik beschrieben ist, kann der Compiler Programmcode leicht auf Korrektheit prüfen.

Probenplan

Fünf Tänzer proben für einen Auftritt: Alex, Bojan, Coco, Deniz und Emil.
Beim Auftritt bilden die Tänzer diese verschiedenen Paare:

- Alex - Bojan
- Coco - Alex
- Emil - Deniz
- Alex - Emil
- Coco - Deniz
- Bojan - Coco
- Deniz - Alex
- Coco - Emil

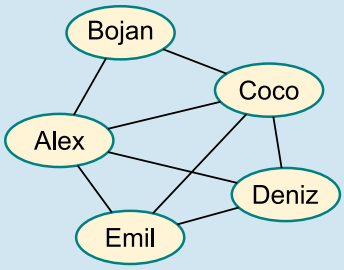
Morgen sollen die Paare hintereinander proben. Dazu macht die Trainerin zwei Ansagen:

1. Der Probenplan wird so gestaltet, dass immer ein Mitglied eines Paares auch zum nächsten Paar gehört und direkt weitermachen kann. Ein Beispiel: Nach der Probe des Paars Alex-Bojan probt ein anderes Paar, zu dem entweder Alex oder Bojan gehört: Coco-Alex, Alex-Emil, Bojan-Coco oder Deniz-Alex.
2. Jeder Tänzer muss höchstens zweimal direkt hintereinander proben.

Einer der Tänzer kann später zur Probe kommen: Wenn die Ansagen der Trainerin gelten, wird er auf keinen Fall zum ersten Paar auf dem Probenplan gehören.

Welcher Tänzer ist das? **A)** Alex **B)** Bojan **C)** Coco **D)** Deniz **E)** Emil

B ist die richtige Antwort

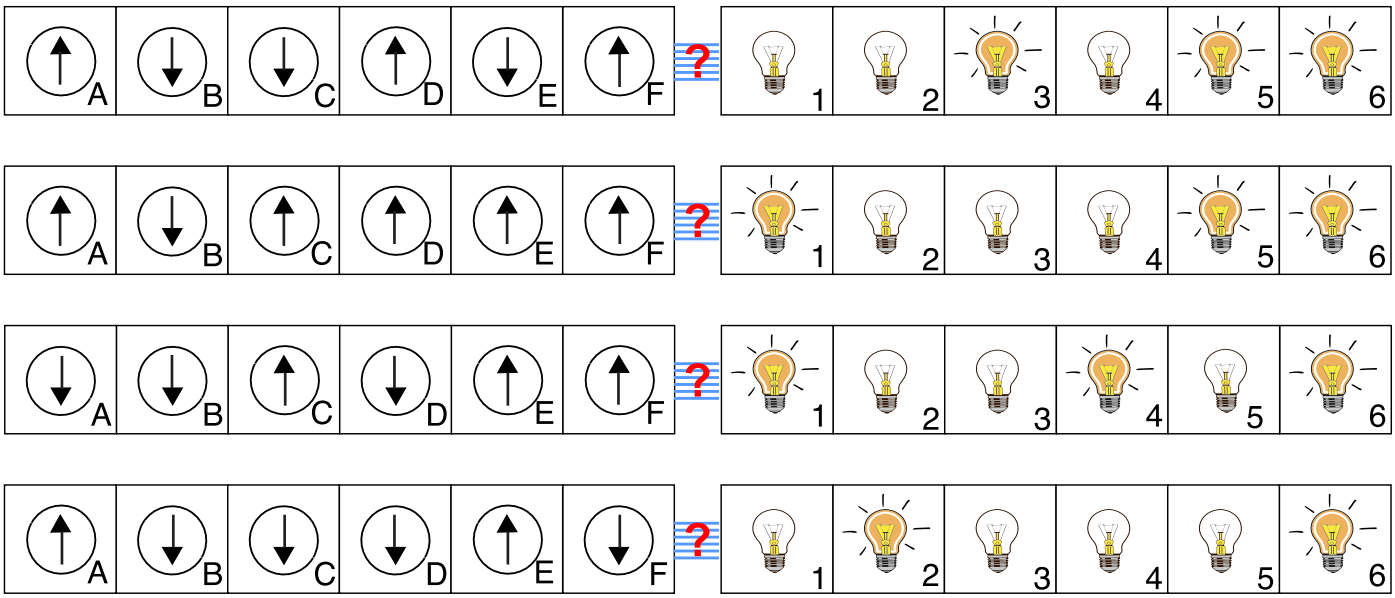


Im Bild ist für jeden Tänzer ein Oval mit seinem Namen zu sehen. Außerdem sind zwei Ovale mit einer Linie verbunden, wenn die beiden Tänzer ein Paar bilden. Ein Probenplan, für den die Ansage der Trainerin gilt, ist dann ein „Weg" durch das Bild: Dieser beginnt bei einem Oval und geht genau einmal an jeder Paar-Linie entlang. Dabei muss der Weg jedes Oval, zu dem er zwischendurch kommt, auch wieder verlassen. Von diesen Ovalen muss also eine gerade Anzahl an Linien ausgehen. Ausnahmen sind die Ovale, an denen der Weg beginnt bzw. endet; von diesen Ovalen geht eine ungerade Anzahl Linien aus. Allein von den Ovalen von Deniz und Emil geht eine ungerade Anzahl Linien aus (nämlich jeweils drei Linien). Nur diese beiden können also zum ersten (oder zum letzten) Paar gehören. Bojan ist der einzige Tänzer, der weder mit Deniz noch mit Emil ein Paar bildet. Nur er kann auf keinen Fall zum ersten Paar des Probenplans gehören.

Das ist Informatik!
Das Bild oben zeigt, wie die Tanzpaare als *Graph* dargestellt werden. Ein Graph besteht aus Knoten (hier: die Tänzer) und Kanten (hier: die Paar-Bildungen der Tänzer). Das ist eine sehr vielseitige Struktur, die bei vielen Informatik-Aufgabenstellungen zur Modellierung insbesondere von Netzwerken verwendet wird, z. B. bei Verkehrs- oder Kommunikationsnetzen. In dieser Biberaufgabe bilden die Tänzer ein Paar-Netzwerk. In vielen Netzwerken kann es nötig sein, auf einem Weg von einem Start- zu einem (unterschiedlichen) Zielknoten alle Verbindungen abzugehen. Dabei stellt sich die Frage (z. B. aus Gründen der Effizienz), ob es möglich ist, auf einem solchen Weg jede Verbindung nur einmal zu begehen. Falls das möglich ist, wird ein entsprechender Weg auch als Eulerweg bezeichnet. Denn schon der Erfinder der Graphentheorie, Leonhard Euler, hat bewiesen, dass das genau dann möglich ist, wenn exakt zwei Knoten eine ungerade Anzahl Verbindungen mit anderen Knoten haben (und alle anderen Knoten folglich eine gerade Anzahl von Verbindungen haben). Nur diese beiden Knoten können Anfangs- oder Endpunkt des Eulerweges sein. Vielleicht hast du das Paar-Netzwerk dieser Biberaufgabe schon einmal gesehen. Wenn man es ein wenig dreht und die Knoten verkleinert, erkennt man die Ähnlichkeit besser: Das ist das Haus vom Nikolaus! Wer das Haus vom Nikolaus in einem Zug zeichnet, geht also einen Eulerweg entlang der Linien des Hauses.

Schalter und Lampen

Die sechs Schalter A bis F sind jeweils mit genau einer der Lampen 1 bis 6 verbunden.
Aber welcher Schalter schaltet welche Lampe?
Um das herauszufinden, wurden diese Versuche durchgeführt:



Leider weiß man bei den alten Drehschaltern nicht so genau, wie sie funktionieren: Einige Schalter sind „an“, wenn der Pfeil nach oben zeigt, bei den anderen Schaltern ist es genau umgekehrt.

Welcher Schalter schaltet welche Lampe?

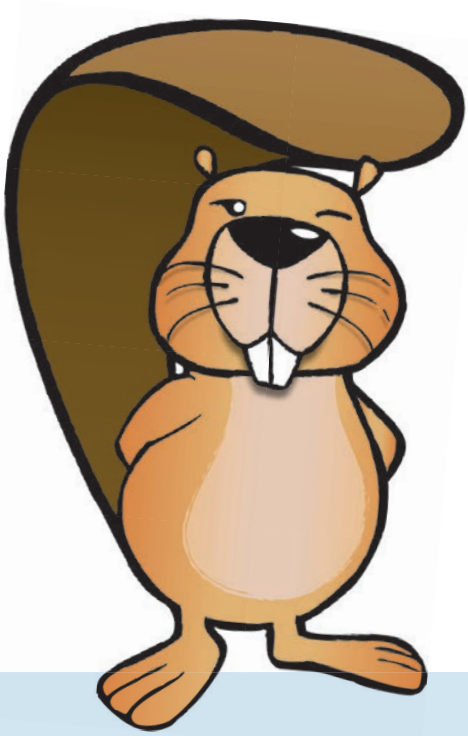
A) A-4, B-6, C-1, D-5, E-3, F-2

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

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

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

Du hast fast das ganze Biberheft gelesen.
Zielgruppe: Informatik-interessiert.
Du solltest Programmieren lernen:
wettbewerb.jwinf.de
cscircles.cemc.uwaterloo.ca/de



Antwort A ist richtig:
Wir schauen uns für jeden Schalter seine Positionen bei den vier Versuchen an und vergleichen diese mit den Zuständen der Lampen. Da für den einzelnen Schalter nicht bekannt ist, in welcher Position er an bzw. aus ist, gibt es immer zwei Möglichkeiten, welche Lampenzustände zu den Schalterpositionen passen.

Schalter A: ↑ ↑ ↓ ↑ passt zu einer Lampe mit an-an-aus-an (gibt es nicht) oder aus-aus-an-aus (Lampe 4).
Schalter B: ↓ ↓ ↓ ↓ passt zu einer Lampe mit aus-aus-aus-aus (gibt es nicht) oder an-an-an-an (Lampe 6).
Schalter C: ↓ ↑ ↑ ↓ passt zu einer Lampe mit aus-an-an-aus (Lampe 1) oder an-aus-aus-an (gibt es nicht).
Schalter D: ↓ ↑ ↑ ↑ passt zu einer Lampe mit an-an-aus-aus (Lampe 5) oder aus-aus-an-an (gibt es nicht).
Schalter E: ↓ ↑ ↑ ↑ passt zu einer Lampe mit aus-an-an-an (gibt es nicht) oder an-aus-aus-aus (Lampe 3).
Schalter F: ↑ ↑ ↑ ↓ passt zu einer Lampe mit an-an-an-aus (gibt es nicht) oder aus-aus-aus-an (Lampe 2).

Damit können alle Schalter eindeutig einer Lampe zugeordnet werden: A-4, B-6, C-1, D-5, E-3, F-2.
Außerdem haben wir herausgefunden, wie die An-Positionen der Schalter A bis F lauten:
↓, ↓, ↑, ↑, ↓ und ↓.

Das ist Informatik!
Um die innere Funktionsweise des Systems aus Schaltern und Lampen zu enthüllen, haben wir „Reverse Engineering“ betrieben. Darunter versteht man ganz allgemein, den Bauplan eines Systems herauszufinden, indem man das Verhalten des Systems beobachtet und untersucht. Die Informatik benutzt Reverse Engineering zum Beispiel, um den Quellcode eines Programms herauszufinden, von dem nur der Maschinencode bekannt ist. Die für diese Art des Reverse Engineering bekannten Verfahren sind in Informatik-Werkzeugen wie „Disassemblern“ oder „Decompilern“ zusammengefasst. Ein zweites Beispiel ist das „Sniffen“, also das Beobachten der Kommunikation zwischen Systemen, um die Regeln dieser Kommunikation zu entdecken oder herauszufinden, auf welche Eingaben die Systeme wie reagieren. In dieser Biberaufgabe haben wir bei der Beobachtung der Versuche mit Schaltern und Lampen also auch ein wenig „gesnifft“.

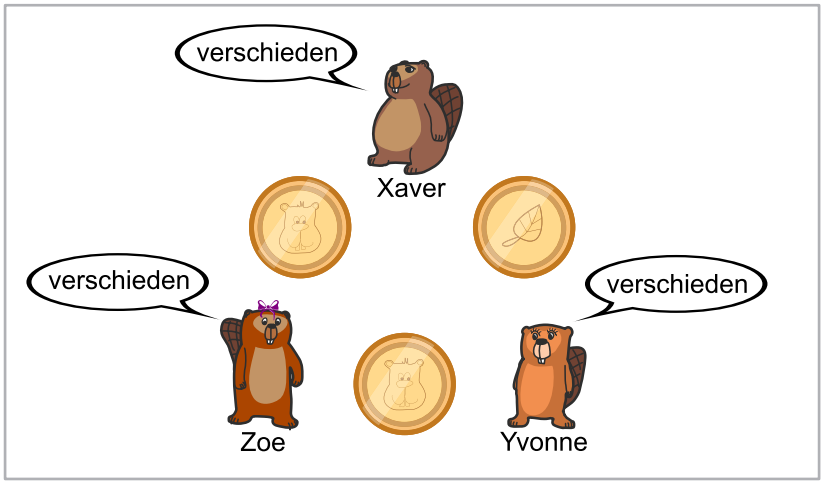
Streng geheim

Die drei Freunde Xaver, Yvonne and Zoe besuchen die berühmte Eisdiele „Al Goritmo“.
Als sie fertig sind, sagt der Kellner: „Die Rechnung ist schon bezahlt.“
Wer hat das Eis bezahlt: Einer von ihnen, oder jemand anderes?
Das wollen sie herausfinden, ohne sich direkt zu fragen – und zwar so:

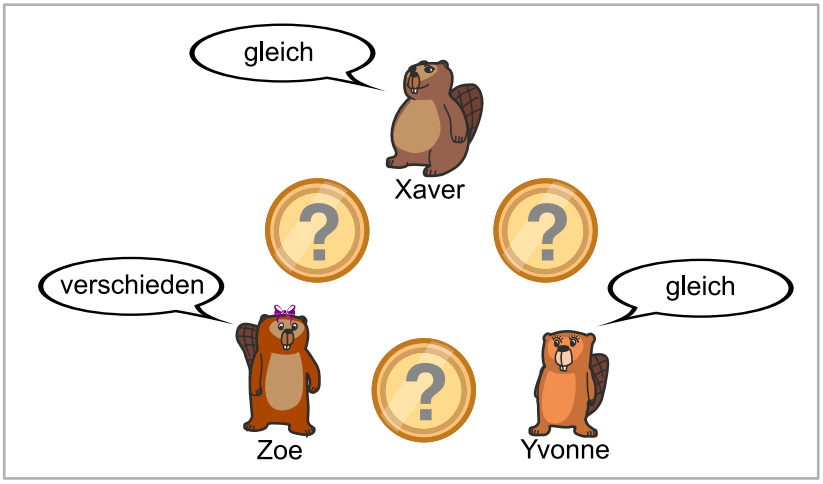
Zuerst werfen sie paarweise je die gleiche Münze: Xaver und Yvonne, Xaver und Zoe sowie Yvonne und Zoe. Anschließend kennt jeder zwei Münzwurf-Ergebnisse und sagt den anderen, ob diese gleich oder verschieden ausgefallen sind. Aber Achtung:

- Wenn jemand von ihnen das Eis bezahlt hat, sagt sie oder er die Unwahrheit.
- Wer nicht bezahlt hat, sagt die Wahrheit.

Ein Beispiel: Wir nehmen an, Zoe habe das Eis bezahlt – streng geheim.
Die Biber werfen Münzen, das Bild zeigt die Ergebnisse. Alle sagen „verschieden“.



In der Eisdiele wird nun die Münze geworfen.
Biber Nosy am Nachbartisch hört nur, was die Freunde nach den Münzwürfen sagen:
Xaver sagt „gleich“, Yvonne sagt „gleich“ und Zoe sagt „verschieden“.



Was weiß der schlaue Nosy jetzt?

A) Keiner der drei Freunde hat das Eis bezahlt.
B) Einer der drei Freunde hat das Eis bezahlt, aber Nosy weiß nicht wer.
C) Einer der drei Freunde hat das Eis bezahlt, und Nosy weiß genau wer.
D) Nosy weiß nicht, ob einer der drei Freunde das Eis bezahlt hat.

Antwort B ist richtig:
Einer der drei Freunde hat das Eis bezahlt, aber Nosy weiß nicht wer.
Bei drei Münzwürfen gibt es genau zwei Möglichkeiten, wie die Ergebnisse ausfallen können:

1. Alle drei Würfe haben das gleiche Ergebnis.
2. Ein Wurf hat ein anderes Ergebnis als die beiden anderen Würfe.

Wenn alle Biber die Wahrheit sagen (weil jemand anderes das Eis bezahlt hat), gilt für diese beiden Möglichkeiten:

1. Alle drei Freunde sagen „gleich".
2. Einer der Freunde sagt „gleich" und die zwei anderen sagen „verschieden".

Das bedeutet: Wenn zwei Freunde „gleich" und einer „verschieden" sagt, sagen nicht alle drei Freunde die Wahrheit. Einer von ihnen muss also das Eis bezahlt haben. Wer das ist, kann Nosy aber nicht wissen, ohne die Ergebnisse der Münzwürfe zu kennen.

Das ist Informatik!
Der amerikanische Informatiker David Chaum hat sich in Theorie und Praxis damit beschäftigt, wie Anonymität im Zusammenhang mit Einsatz und Benutzung von Informatiksystemen funktionieren kann. Unter anderem beschrieb er Anfang der 1980er Jahre das „Dining Cryptographers Problem". Es stellt die Frage, ob Information so übertragen werden kann, dass sowohl der Absender anonym bleibt als auch der Empfänger nicht identifizierbar ist.
Als Lösung schlägt er genau die Vorgehensweise vor, an die sich die drei Freunde in dieser Biberaufgabe halten. So kann einer der drei als Absender ein Bit an Information übertragen: ob er das Eis bezahlt hat oder nicht. Das tut er, indem er als einziger nicht die logische XOR-Funktion (die sagt, ob zwei Bits gleich oder verschieden sind) auf die beiden ihm bekannten Münzwurf-Bits („Kopf" oder „Zahl") anwendet, sondern deren Negation. Damit am Ende aus den drei von den Freunden mitgeteilten Bits („gleich" oder „verschieden") wieder ein Bit wird, muss abschließend noch einmal die XOR-Funktion angewendet werden. Das Gesamtergebnis ist genau dann 1 (oder „wahr" oder „Kopf" oder …), wenn der Absender das Eis bezahlt hat, also auch das Ausgangsbit 1 ist. Aber der Absender bleibt anonym, denn das Ergebnis hängt nicht davon ab, wer das Eis bezahlt hat. Auch einen besonderen Empfänger gibt es nicht, da alle, einschließlich Außenstehender wie Nosy, die Nachricht gleichermaßen erhalten.

https://en.wikipedia.org/wiki/Dining_cryptographers_problem

Treffpunkt

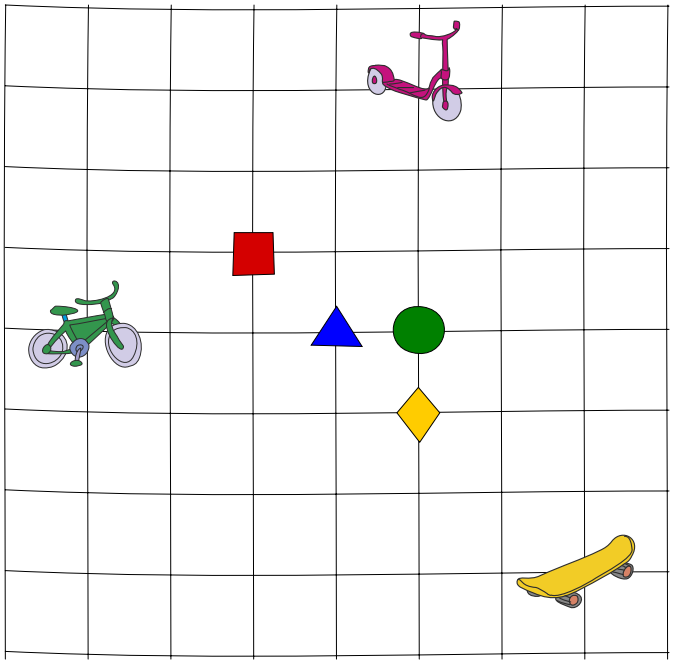
Drei Freunde wollen sich treffen. Sie starten mit ihren Fahrzeugen an verschiedenen Kreuzungen.
Zwischen zwei Kreuzungen fahren sie immer den kürzesten Weg,
über die Straßen zwischen den Kreuzungen.

Ein Beispiel: Zur Kreuzung  muss der Roller  4 Straßen fahren.

Die Freunde können sich an den Kreuzungen  ,  ,  oder  treffen.
Der beste Treffpunkt ist der, zu dem sie insgesamt möglichst wenig Straßen fahren müssen.

Was ist der beste Treffpunkt?

Klicke auf die richtige Kreuzung.



So ist es richtig:
Der beste Treffpunkt ist die Kreuzung mit dem grünen Kreis  :

Zu dieser Kreuzung müssen die Freunde 3 + 4 + 5 = 12 Straßen fahren.

Zur Kreuzung  müssen sie 4 + 3 + 8 = 15 Straßen fahren.

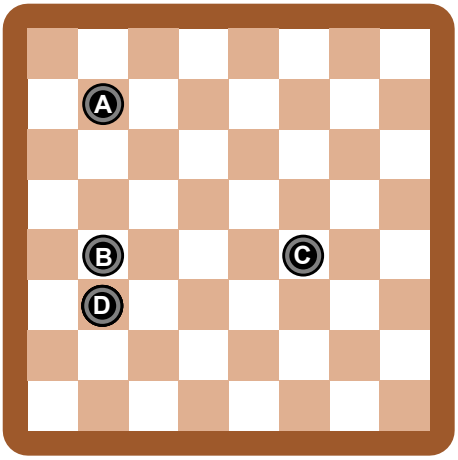
Zur Kreuzung  müssen sie 4 + 3 + 6 = 13 Straßen fahren.

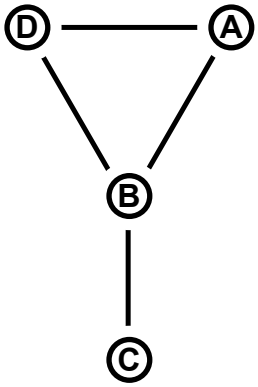
Zur Kreuzung  müssen sie 4 + 5 + 4 = 13 Straßen fahren.

Das ist Informatik!
Diese Biberaufgabe erscheint zunächst als Rechenaufgabe: Für jeden der vier Treffpunkte wird zunächst ausgerechnet, wie viele Straßen jeder der drei Freunde dorthin fahren muss, und dann wird die Summe dieser Zahlen gebildet. Der beste Treffpunkt ist dann der mit der niedrigsten Summe. Das ist nicht schwierig, aber es sind doch 4 [Treffpunkte] * (3 [Freunde] + 1 [Summenbildung]) = 16 Berechnungen zu erledigen. Um da nicht die Übersicht zu verlieren, muss man die Treffpunkte und die Freunde systematisch abarbeiten. Die Planung und Beschreibung einer systematischen Vorgehensweise als Algorithmus ist eine der häufigsten Tätigkeiten von Informatikerinnen und Informatikern.

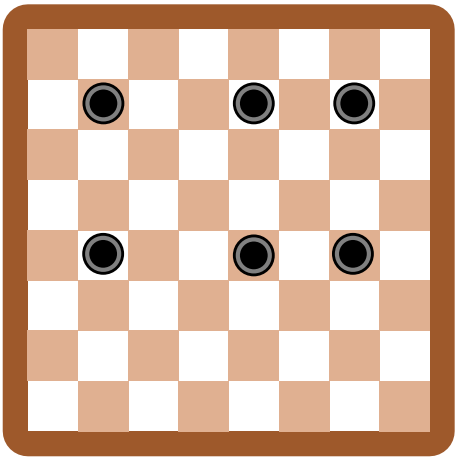
Verbunden

Auf einem Spielbrett liegen vier Spielsteine.
Zwei Spielsteine, die auf der selben Zeile oder auf der selben Spalte des Spielbretts liegen, gelten als *verbunden.*
Das Bild neben dem Spielbrett beschreibt die Spielsituation. Die Linien zeigen, wie die Steine verbunden sind.



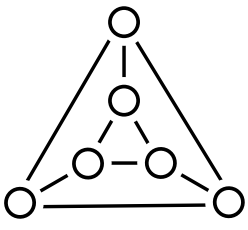


Jetzt liegen sechs Spielsteine auf dem Spielbrett.

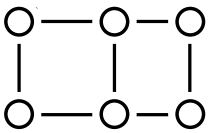


Welches Bild beschreibt die neue Spielsituation?

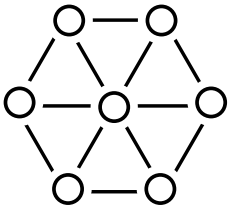
A)



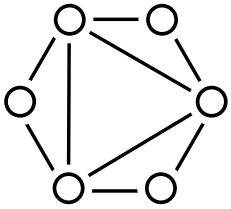
B)



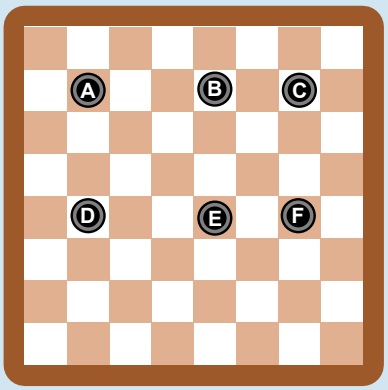
C)

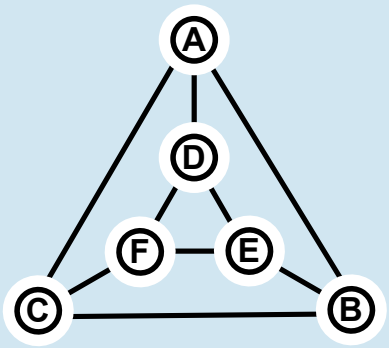


D)



Antwort A ist richtig:
Das ist leichter zu erkennen, wenn die Spielsteine und die Kreise in Antwort A beschriftet sind.





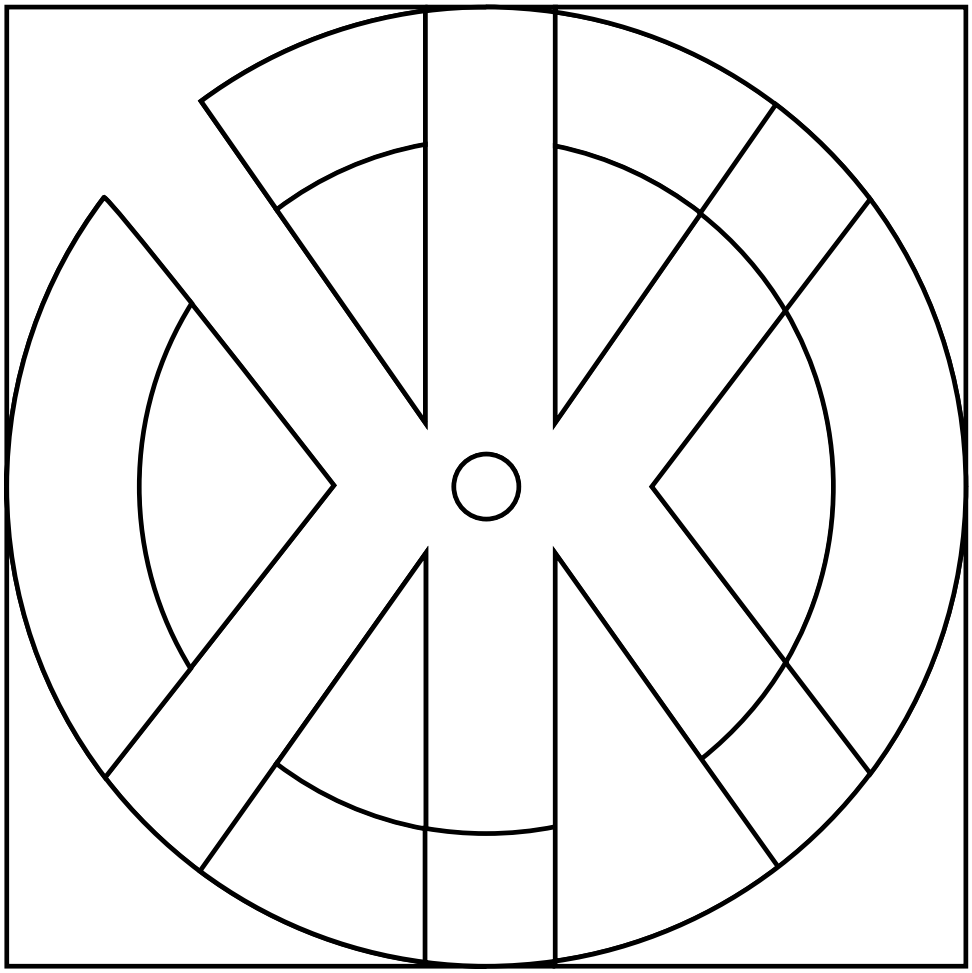
Aber die richtige Antwort kann auch ohne passende Beschriftung gefunden werden. Es genügt, allein die Eigenschaften der in den Antworten dargestellten Strukturen zu betrachten. Auf dem Spielbrett befinden sich 6 Steine; also kann nur ein Bild mit sechs Kreisen die Spielsituation beschreiben. Antwort C mit sieben Kreisen scheidet also aus. Auf dem Spielbrett liegt jeder Spielstein mit zwei anderen Spielsteinen auf derselben Zeile und mit einem anderen Spielstein auf der selben Spalte. In einem Bild, das diese Spielsituation darstellt, muss also jeder Kreis mit genau 2 + 1 = 3 anderen Kreisen verbunden sein. Nur die in Antwort A dargestellte Struktur hat diese Eigenschaft.

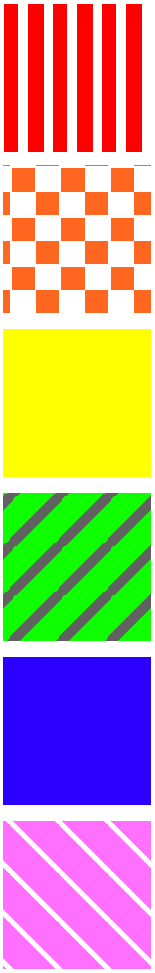
Das ist Informatik!
Die Informatik befasst sich, wie der Name sagt, mit der Verarbeitung von Information. Wenn ein Informatiksystem für eine bestimmte Anwendung entwickelt wird, ist eine der wichtigsten Fragen, welche und wie viel Information verarbeitet werden soll. Um unnötigen Aufwand zu vermeiden, wird Information, die für die Anwendung keine Rolle spielt, außer Acht gelassen.
In dieser Biberaufgabe werden Graphen verwendet, um darzustellen, ob zwei Spielsteine in einer Reihe bzw. Spalte liegen. Die Spielsteine sind die Knoten des Graphen, und zwei Knoten sind verbunden, wenn die zugehörigen Steine in einer Reihe bzw. Spalte liegen. Außer Acht gelassen sind z. B. die Farben der Felder, auf denen die Steine liegen. Würde in einer Anwendung nur die Anzahl der Spielsteine auf dem Brett eine Rolle spielen, wären wiederum die Graphen überflüssig; eine einzige Zahl genügte.

Wie viele Farben?

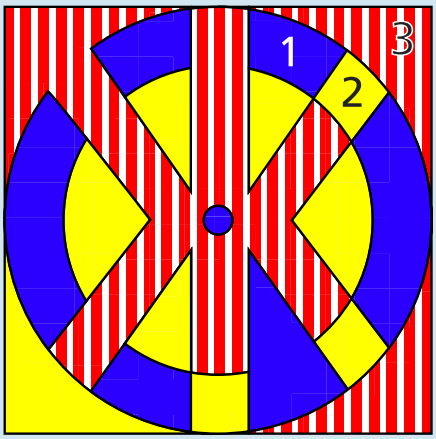
Das Bild unten soll schön bunt werden.
Aber: Wenn zwei Flächen nebeneinander liegen, müssen sie verschiedene Farben haben.
Zum Ausprobieren kannst du die Farben auf das Bild ziehen.
Verwende so wenige Farben wie möglich!

Wie viele verschiedene Farben genügen?



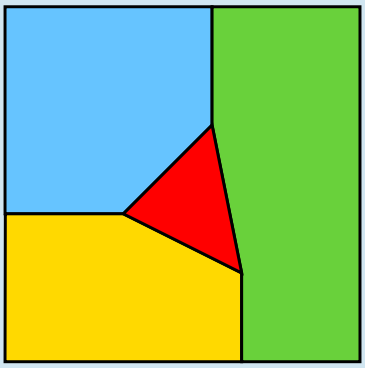


3 ist die richtige Antwort:
Hier ist ein Beispiel, wie man mit drei verschiedenen Farben das Bild bunt einfärben kann, so dass alle Flächen, die nebeneinander liegen, verschiedene Farben haben. Die große Fläche mit der Ecke oben links und alle anderen Flächen, die nicht daneben liegen (nämlich die Ecken oben und unten rechts), wurden rot(-weiß) gefärbt. Dann wurden die Flächen im Kreis so weit wie möglich blau gefärbt, wieder von links oben aus. Alle restlichen Flächen wurden gelb gefärbt.



Mit weniger Farben geht es nicht. Schauen wir uns die Flächen 1, 2 und 3 an. 1 und 2 liegen nebeneinander, müssen also verschiedene Farben haben. Beide Flächen liegen jeweils auch neben Fläche 3. Deshalb muss 3 eine andere Farbe als 1 und auch eine andere Farbe als 2 haben. Allein für diese drei Flächen werden also drei verschiedene Farben benötigt.

Das ist Informatik!
In dieser Biberaufgabe genügen drei Farben, um ein Bild mit Flächen so einzufärben, dass benachbarte Flächen nicht die gleiche Farbe haben. Aber ist das immer so? Die Antwort lautet: Nein; aber man kann immer mit vier Farben auskommen. Hier ist ein Bild, in dem jede Fläche drei Nachbarn hat. Deshalb werden vier Farben benötigt.



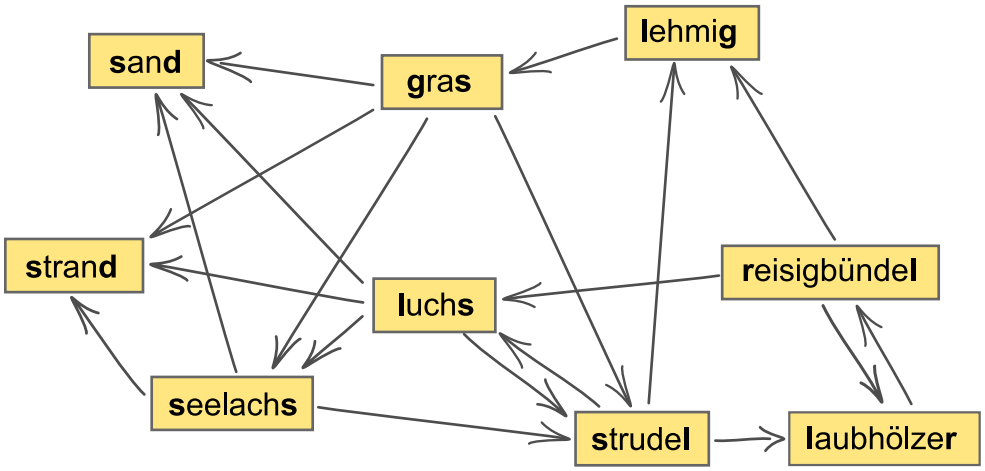
Die Frage nach der Farbenzahl stammt aus der Mathematik. Ihre allgemeine Beantwortung war so schwierig und kompliziert, dass Computerprogramme dabei zum Einsatz kamen. Schwierig kann auch das Problem sein, für ein gegebenes Bild, z. B. eine Landkarte, die kleinste Anzahl von Farben zu finden. Das ist schade, denn deshalb sind auch andere Probleme schwierig. Zum Beispiel ist auch die Erstellung eines Stundenplans ein Einfärbungsproblem: Die „Flächen“ sind dann die zu unterrichtenden Stunden, und „Nachbarn“ sind die Stunden, die von der gleichen Lehrkraft oder in der gleichen Klasse unterrichtet werden. Sie dürfen nicht der gleichen „Farbe“, also der gleichen Schulstunde (z. B. Dienstag 4. Stunde) zugeordnet werden.

https://de.wikipedia.org/wiki/Färbung_(Graphentheorie)

Wörterkette

Die Biber spielen „Wörterkette“.

Aus einer Menge von Wörtern wählen sie ein beliebiges Anfangswort aus.
Danach verlängern sie nach und nach die Wörterkette um ein Wort,
welches mit dem letzten Buchstaben des vorigen Worts beginnt.
Jedes Wort darf höchstens einmal verwendet werden.



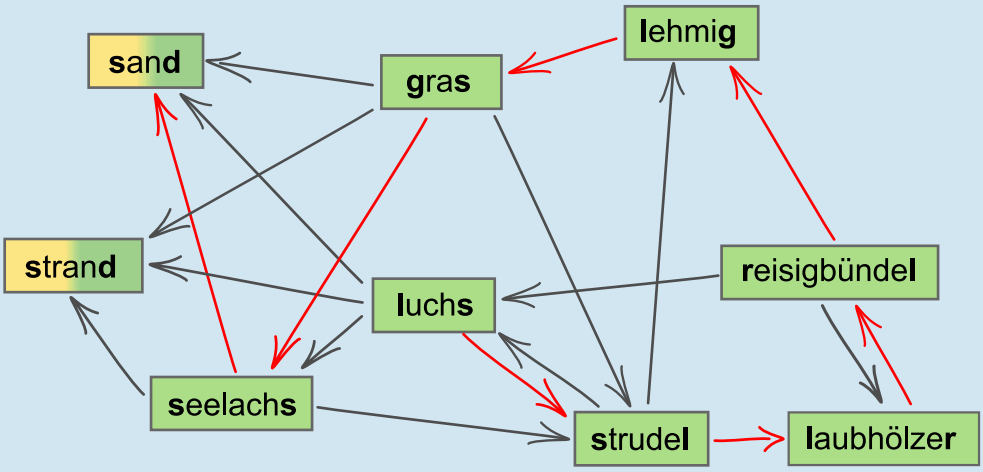
Heute spielen die Biber mit der Wörtermenge, die im Bild gezeigt ist.
Eine Wörterkette aus dieser Menge ist zum Beispiel: strudel ⟶ luchs.
Sie besteht aus zwei Wörtern.

Aus wie vielen Wörtern kann eine Wörterkette aus dieser Menge höchstens bestehen?

8 ist die richtige Antwort:
Das ist eine Wörterkette aus der im Bild gezeigten Menge:
luchs ⟶ strudel ⟶ laubhölzer ⟶ reisigbündel ⟶ lehmig ⟶ gras ⟶ seelachs ⟶ sand
Sie besteht aus acht Wörtern. Kannst du eine andere Kette aus acht Wörtern finden?

Eine bestimmte Wörterkette aus acht Wörtern anzugeben beweist, dass die Antwort auf die Frage dieser Biberaufgabe mindestens 8 lautet. Da die Wörtermenge neun Wörter enthält, kann eine Wörterkette aus dieser Menge grundsätzlich höchstens aus 9 Wörtern bestehen. Nun betrachten wir die Wörter „sand" und „strand". Beide Wörter enden mit „d". Es gibt aber kein Wort, das mit „d" beginnt. Also muss eines dieser beiden Wörter das letze Wort der Kette sein. Da eine Wörterkette nur ein letztes Wort haben kann, kann eines der beiden Wörter nicht verwendet werden. Deshalb kann eine Wörterkette aus dieser Menge höchstens aus acht Wörtern bestehen.

Mit welchem Wort sollte man am besten beginnen, um eine Wörterkette aus möglichst vielen Wörtern zu bilden? Vielleicht ist dir aufgefallen, dass drei Wörter mit „l" beginnen, aber nur zwei Wörter auf „l" enden. Damit man alle drei Wörter mit „l" am Anfang auch verwenden kann, sollte man also mit einem von ihnen beginnen. Das Bild zeigt die oben angegeben Wörterkette. Gestartet wird bei „luchs", und dann folgt man den roten Pfeilen. Das letzte Wort könnte auch „strand" sein statt „sand".



Das ist Informatik!
In dieser Biberaufgabe wird eine Beziehung zwischen den Elementen einer Menge betrachtet, nämlich die zwischen Wörtern mit übereinstimmenden End- und Anfangsbuchstaben. Eine solche Beziehung kann gut als Graph modelliert und, wie im Bild in der Aufgabe, durch Linien zwischen den Elementen veranschaulicht werden. Hier ist der Graph gerichtet und wird deshalb mit Pfeilen dargestellt. Die Beziehung zwischen den Wörtern hat nämlich eine Richtung: Man kann von „gras" zu „sand" gehen, aber nicht umgekehrt.
Für viele Probleme, die mit Graphen modelliert werden können, kennt die Informatik gute Lösungen. Aber es gibt auch Gegenbeispiele: In der Sprache der Graphen ist die längste Wörterkette der längste Weg, den man über die Pfeile des Wörter-Graphen gehen kann. Im Allgemeinen gehört das Finden eines längsten Weges zu den schwierigsten Graphen-Problemen und zu den schwierigsten Problemen, die die Informatik überhaupt kennt.

https://de.wikipedia.org/wiki/Längster_Pfad

Zimmerverteilung

Die „Hacking Girls“, ein Computer-Club für Mädchen, planen einen Ausflug mit Übernachtung. Die Jugendherberge hat große Mehrbettzimmer. Aber wer soll mit wem auf ein Zimmer?

Jedes Clubmitglied schreibt auf eine Wunsch-Karte,

- mit welchen Mädchen sie unbedingt (+) und
- mit welchen Mädchen sie absolut nicht (–)

im gleichen Zimmer sein möchte.

Die Club-Vorsitzende muss die Mädchen auf die Zimmer verteilen.
Sie will alle Zimmer-Wünsche erfüllen.

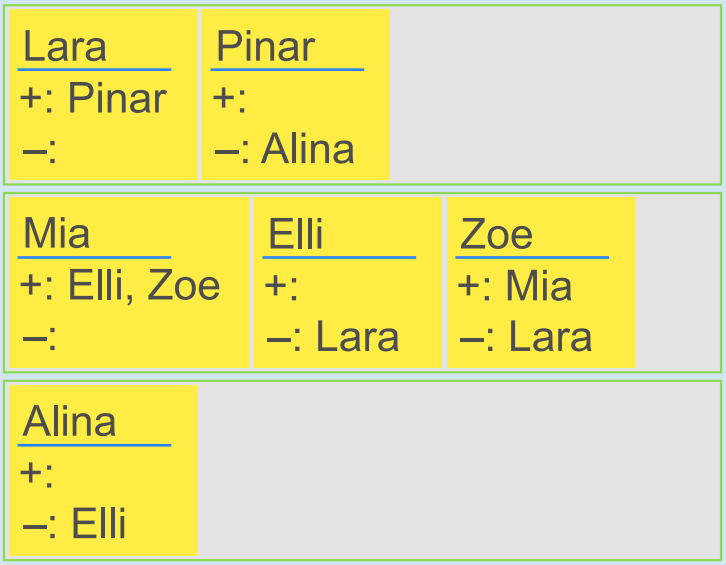
Hilf ihr und verteile die Mädchen auf die Zimmer!

Ziehe dazu ihre Wunsch-Karten so auf die drei „Zimmer“, dass alle Wünsche erfüllt werden.
Zum Glück ist das möglich.*

Alina	Elli	Lara	Mia	Pinar	Zoe
+:	+:	+: Pinar	+: Elli, Zoe	+:	+: Mia
–: Elli	–: Lara	–:	–:	–: Alina	–: Lara

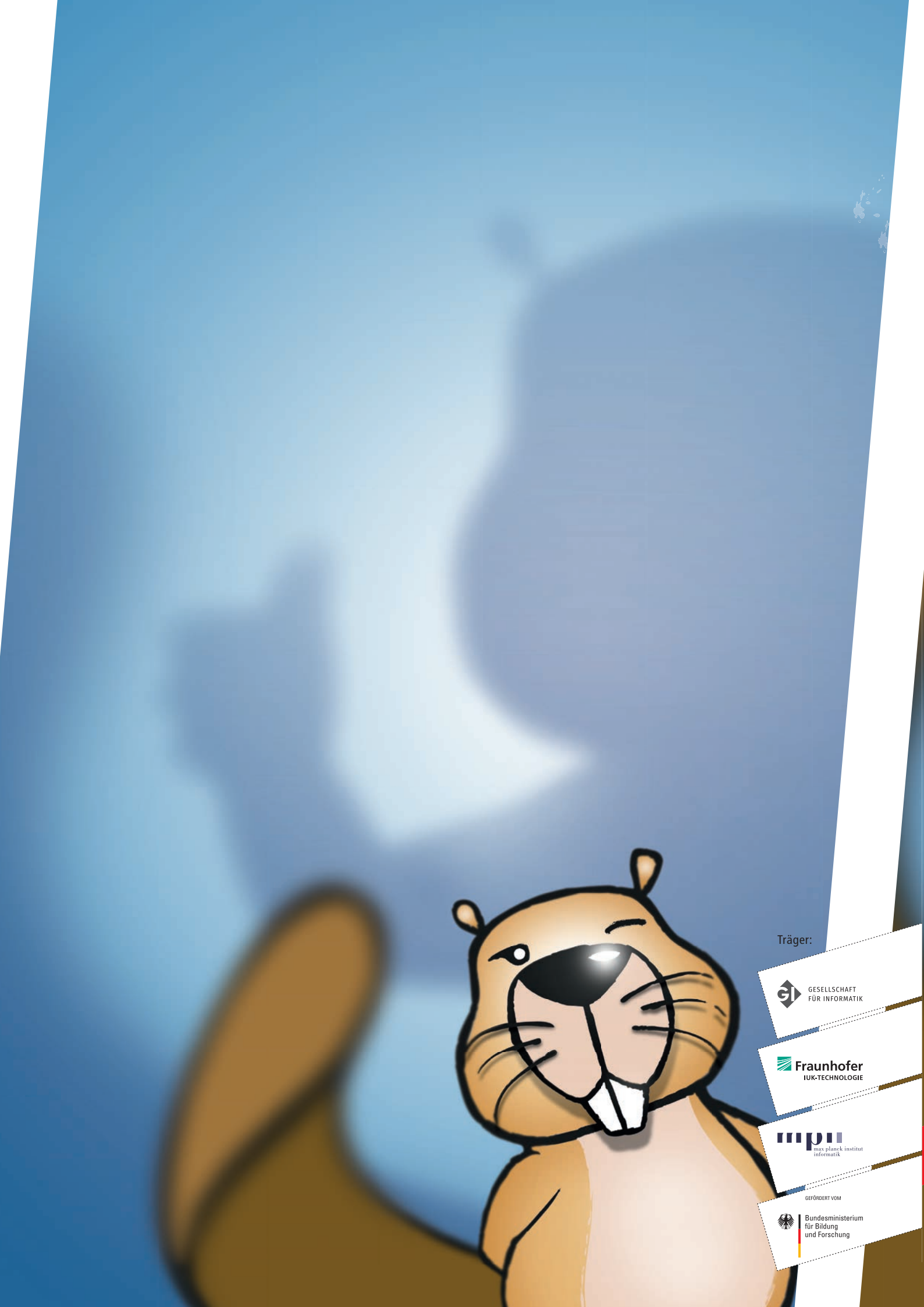
** Im Informatik-Biber 2018 wurde diese Aufgabe mit teils anderen Namen verwendet.*

So ist es richtig:
Wenn genügend Zimmer vorhanden sind, genügt es, sich auf die positiven Wünsche zu konzentrieren. Das geht so:
Für jedes Mädchen M suchen wir ein Zimmer (also: ein Rechteck) mit mindestens einer Wunsch-Karte eines anderen Mädchens M2, so dass M2 auf der Liste „+“ von M oder M auf der Liste „+“ von M2 steht. Gibt es so ein Zimmer, dann ziehen wir die Karte von M auf das Zimmer; sonst ziehen wir die Karte auf ein leeres Zimmer.
Jetzt sind alle positiven Wünsche erfüllt. Aber es kann sein, dass wir dabei negative Wünsche verletzt haben. Das müssen wir prüfen: Gibt es ein Zimmer mit einer Karte, die auf der Liste „–“ einer anderen Karte im gleichen Zimmer steht, lassen sich nicht alle Wünsche erfüllen.



Zum Glück ist es möglich, alle Wünsche zu erfüllen. Auf die oben beschriebene Weise erhalten wir die abgebildete Zimmerverteilung. Diese ist die einzige Verteilung, die alle Wünsche erfüllt: Lara muss mit Pinar im gleichen Zimmer sein. Aber Lara (und damit auch Pinar) kann weder mit Elli noch mit Zoe im gleichen Zimmer sein, und damit auch nicht mit Mia, die wiederum mit Elli und Zoe zusammen sein muss. Alina muss in ein eigenes Zimmer, da sie weder mit Pinar (und damit auch Lara) noch mit Elli (und damit auch Mia und Zoe) zusammen sein kann.
Es kann natürlich leicht passieren, dass nicht alle Wünsche erfüllbar sind. Wenn z. B. Lara unbedingt mit Pinar, aber Pinar absolut nicht mit Lara im gleichen Zimmer sein möchte, lassen sich diese beiden Wünsche nicht gleichzeitig erfüllen.

Das ist Informatik!
Die Club-Vorsitzende hat ein Problem: Sie sucht nach einer Zimmerverteilung, die alle Wünsche erfüllt. Solch eine Verteilung ist eine „Lösung“ ihres Problems. Aber wie kann sie diese Lösung finden? Eine einfache Möglichkeit: Sie verteilt die Mädchen auf alle möglichen Weisen auf die Zimmer, ohne auf die Wünsche zu achten, und betrachtet dann jede fertige Verteilung, ob sie alle Wünsche erfüllt. Wenn die Gruppe größer ist, kann es aber ziemlich viele Verteilungen geben; diese Methode kostet dann viel Zeit. Es geht schneller, wenn man die durch die Wünsche der Mädchen gegebenen Bedingungen an die Verteilung von vornherein berücksichtigt und nur solche Verteilungen erzeugt, die alle Bedingungen erfüllen und deshalb Lösungen sind.
Probleme, bei denen Bedingungen erfüllt werden müssen, um zu einer Lösung zu kommen, werden auch „Constraint-Satisfaction-Probleme“ genannt und treten in der Informatik häufig auf. Dabei geht es wie in dieser Biberaufgabe häufig darum, dass bestimmte Dinge (auch „Ressourcen“ genannt) ohne Konflikt benutzt werden sollen, wie etwa die Start- und Landebahnen eines Flughafens. Die Informatik kennt Verfahren, mit denen die meisten solcher Probleme effizient gelöst werden können.


Träger:
GESELLSCHAFT FÜR INFORMATIK
Fraunhofer IUK-TECHNOLOGIE
max planck institut informatik
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung