Informatik-Biber

Der Wettbewerb zum digitalen Denken.

BUNDES WEIT INFORMATIK NACHWUCHS FÖRDERN

Bundesweite Informatikwettbewerbe

AUFGABEN 2024

bwinf.de/biber

Herausgeber:
Wolfgang Pohl und Susanne Datzko-Thut, BWINF

Der Aufgabenausschuss Informatik-Biber 2024

Susanne Datzko-Thut, BWINF
Franziska Kaltenberger, University of Edinburgh
Wolfgang Pohl, BWINF
Kirsten Schlüter, Bayerisches Staatsministerium für Unterricht und Kultus
Margaretha Schlüter, freiheit.com technologies GmbH
Karsten Schulz, Digital Technologies Institute
Jacqueline Staub, Universität Trier
Michael Weigend, WWU Münster

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

Masiar Babazadeh, Scuola universitaria professionale della Svizzera italiana
Simon Bachl, Österreichische Computer Gesellschaft (OCG)
Liam Baumann, OCG
Wilfried Baumann, OCG
Christian Datzko, Gymnasium Liestal
Nora A. Escherle, Schweiz. Verein für Informatik in der Ausbildung (SVIA)
Gerald Futschek, Technische Universität Wien (TU Wien)
Martin Kandlhofer, OCG
Regula Lacher, ETH Zürich / ABZ (Ausbildungs- und Beratungszentrum für Informatikunterricht)
Lukas Lehner, TU Wien
Gabriel Parriaux, Haute École Pédagogique Vaud / SVIA
Jean-Philippe Pellet, Haute École Pédagogique Vaud / SVIA
Zsuzsa Pluhár, ELTE Informatikai Kar
Dirk Schmerenbeck, Universität Trier
Giovanni Serafini, ETH Zürich / ABZ
Florentina Voboril, TU Wien
Andreas Zottl, OCG

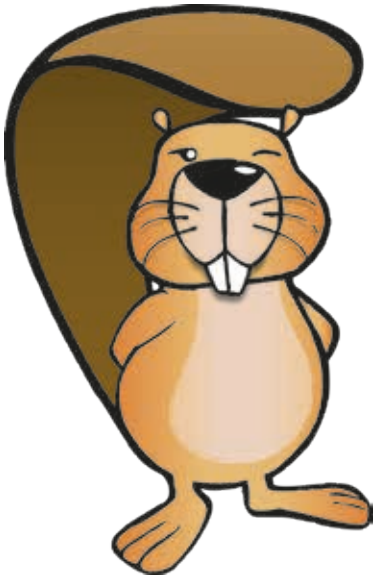
Der Informatik-Biber

ist einer 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.

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 18. Austragung im Jahr 2024 beteiligten sich 3.021 Schulen und andere Bildungseinrichtungen mit 542.997 Schülerinnen und Schülern; das sind neue Rekordwerte. Die Möglichkeit, auch in Zweierteams zu arbeiten, wurde gern genutzt.

Die Teilnahme am Informatik-Biber 2024 war mit Desktops, Laptops und Tablets möglich. Die Bearbeitung der Aufgaben sind durch verschiedene Interaktionsformen sehr abwechslungsreich. In diesem Biberheft ist diese Dynamik der Aufgabenbearbeitung nicht vorführbar. Handlungstipps in den Aufgabenstellungen und Bilder von Lösungssituationen geben aber eine Vorstellung davon. Die Online-Aufgaben können seit Frühling 2024 auch übers Jahr von den Lehrpersonen zu eigenen „Bibertests“ zusammengestellt werden. Diese Möglichkeit wird rege genutzt – über 1.500 Bibertests wurden durch Lehrpersonen erstellt und mehr als 20.000 Schüler und Schülerinnen haben an den individuell für sie erstellten Tests teilgenommen.

Der Informatik-Biber 2024 wurde in fünf Altersgruppen durchgeführt. Die Aufgaben jeder Altersgruppe sind in die Schwierigkeitsstufen leicht, mittel und schwer eingeteilt. In den Klassenstufen 3 bis 4 waren innerhalb von 30 Minuten 9 Aufgaben zu lösen, drei in jeder Schwierigkeitsstufe. In den Klassenstufen 5 bis 6 waren innerhalb von 35 Minuten 12 Aufgaben zu lösen, vier pro Schwierigkeitsstufe. 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 jeder Schwierigkeitsstufe.

Die 37 Aufgaben des Informatik-Biber 2024 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 zur jeweiligen Aufgabe stammt. Der Kasten am Aufgabenende enthält Erläuterungen zu 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 2025 wieder beim Informatik-Biber mitzumachen, und zwar in der Zeit vom 10. bis 21. 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 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 damaligen Informatikjahr als „EI:Spiel blitz!“ einen ersten Biber-Testlauf. Seitdem kamen viele Bebras-Länder hinzu. Zum Drucktermin sind es weltweit 82, und weitere Länderteilnahmen sind in Planung. Insgesamt hatte der Bebras Challenge 2023 weltweit fast 4 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 37 Aufgaben des Informatik-Biber 2024 stammen aus 19 Ländern: Deutschland, Belgien, Vietnam, Österreich, Australien, Bulgarien, Brasilien, Kanada, Schweiz, Tschechien, Finnland, Ungarn, Indonesien, Südkorea, Malaysia, Polen, Portugal, Slowenien, Slowakei und aus Taiwan.

Der aserbaidschanische Biber

Deutschland nutzt zusammen mit einer Vielzahl anderer Länder zur Durchführung des Informatik-Biber ein Online-System, das von der niederländischen Firma Cuttle b.v. betrieben und fortentwickelt wird.

Der neuseeländische Biber

Auch dieses Jahr hat sich BWINF bemüht, das Bebras-Mitglied Ukraine zu unterstützen und nach Deutschland geflüchteten ukrainischen Kindern und Jugendlichen eine Teilnahme am ukrainischen „Bober“ zu ermöglichen. 14 Schulen aus Deutschland und 3 aus der Schweiz haben dieses Angebot wahrgenommen.

Der namibische Biber

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

Bundesweite Informatikwettbewerbe

Bei jungen Menschen das Interesse für Informatik wecken, Begabungen entdecken und fördern: das ist das Ziel der Bundesweiten Informatikwettbewerbe (BWINF), an denen im Jahr 2024 über 600.000 junge Menschen teilnahmen. Der Informatik-Biber ist das BWINF-Einstiegsformat; außerdem werden zwei weitere Wettbewerbe und ein Format zur Spitzenförderung angeboten:

Jugendwettbewerb Informatik

Der Jugendwettbewerb Informatik 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 beiden 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 Online-Plattform des Wettbewerbs erworben werden (jwinf.de).

Bundeswettbewerb Informatik

Der Bundeswettbewerb Informatik 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. In der ersten Runde ist Gruppenarbeit möglich, in der zweiten Runde ist 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.

Informatik-Olympiade

Finalist:innen und einige ausgewählte Teilnehmende der zweiten Runde des Bundeswettbewerbs können sich im Folgejahr in mehreren Trainingsrunden für das vierköpfige deutsche Team qualifizieren, das dann an der Internationalen Informatik-Olympiade (IOI) teilnimmt. Auch zu Vorbereitungswettbewerben im europäischen Ausland werden regelmäßig deutsche Teams entsandt.

Austausch

Die Teilnahme an BWINF-Wettbewerben eröffnet Möglichkeiten zum Austausch mit Gleichgesinnten. Erste Anknüpfungspunkte bieten die BWINF-Accounts bei LinkedIn, Instagram (@bwinf) und BlueSky (@bwinf.sky.social), das Portal einstieg-informatik.de und die BWINF-Website. Auf unserem Discord-Server können Teilnehmende miteinander in Kontakt treten (discord.gg/bwinfcommunity). Schon 42 Jahrgänge von Teilnehmenden des Bundeswettbewerbs bilden ein wachsendes Netzwerk, auch im Alumni und Freunde e.V. Teilnehmende vom Bundeswettbewerb lernen sich bei Informatik-Workshops von Hochschulen und Unternehmen kennen.

Mädchenförderung

2020 hat BWINF girls@BWINF gestartet. Mit dieser Initiative girls@BWINF möchten die Bundesweiten Informatikwettbewerbe mehr Mädchen für Informatik begeistern und in Ihrem Interesse bestärken. Hierzu bietet BWINF Teilnehmerinnen die Möglichkeit an, sich zu vernetzen durch die Teilnahme an Camps und einer virtuellen Community. girls@BWINF richtet sich an Schülerinnen, die Lust und Interesse an Informatik und am Programmieren haben, die ihr eigenes Talent entdecken und fördern möchten, die bereits ihr informatisches Talent gezeigt haben.

Träger und Förderer

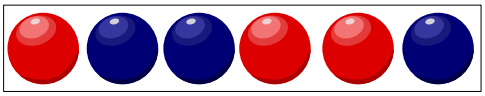
BWINF ist eine Initiative der Gesellschaft für Informatik (GI), des Fraunhofer-Verbunds IUK-Technologie und des Max-Planck-Instituts für Informatik. BWINF wird vom Bundesministerium für Bildung und Forschung (BMBF) gefördert. Die Bundesweiten Informatikwettbewerbe gehören zu den von der Kultusministerkonferenz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

Aufgabenliste

Die 37 Aufgaben des Informatik-Biber 2024 sind auf Seite 6 gelistet, nach ungefähr steigender Schwierigkeit und mit einer informatischen Klassifikation ihres Aufgabenthemas.

Bälle

Emil beschreibt Folgen von blauen und roten Bällen nur mit `0`en und `1`en. Das macht er auf ganz besondere Weise und zeigt das an einem Beispiel:



Im ersten Schritt zählt Emil für jeden Ball, wie viele **blaue** Bälle sich rechts davon befinden, und zählt den Ball selbst mit, wenn der blau ist. Diese Anzahlen schreibt er als Zwischenfolge auf:

`3, 3, 2, 1, 1, 1`

Im zweiten und letzten Schritt formt er die Zwischenfolge so um: Wenn die Zahl **gerade** ist, wird sie zu `0`, wenn sie **ungerade** ist, zu `1`. 0 gilt als gerade Zahl. Emil beschreibt die obige Bälle-Folge also so:

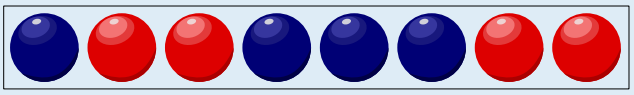
`1, 1, 0, 1, 1, 1`

Eine andere Bälle-Folge beschreibt Emil so:

`0, 1, 1, 1, 0, 1, 0, 0`

Erstelle diese Bälle-Folge!

So ist es richtig:



Es gibt mehrere Wege, die richtige Bälle-Folge zu bestimmen. Ein Weg besteht darin, die Beschreibung von rechts nach links durchzugehen und dabei jeweils zu überlegen, ob der Ball an der entsprechenden Position der Bälle-Folge rot oder blau sein muss.

Wir beginnen **ganz rechts.** Die `0` an dieser Stelle legt fest, dass der Ball rot sein muss: Nur bei einem roten Ball bleibt die Anzahl der blauen Bälle rechts ab dieser Stelle gerade (nämlich 0).

0	1	1	1	0	1	0	0
?	?	?	?	?	?	?	**rot**

An der **zweiten Stelle von rechts** steht wieder eine `0`. Auch hier gilt: Nur bei einem roten Ball bleibt die Anzahl der blauen Bälle rechts ab dieser Stelle gerade.

0	1	1	1	0	1	0	0
?	?	?	?	?	?	**rot**	rot

An der **dritten Stelle von rechts** steht eine `1`. Der Ball an dieser Stelle muss blau sein, um von einer geraden (0) zu einer ungeraden (1) Anzahl blauer Bälle rechts ab dieser Stelle zu kommen.

0	1	1	1	0	1	0	0
?	?	?	?	?	**blau**	rot	rot

An der **vierten Stelle von rechts** steht eine `0`. Der Ball an dieser Stelle muss blau sein, um von der bisherigen ungeraden (1) zu einer geraden (2) Anzahl blauer Bälle rechts ab dieser Stelle zu kommen.

0	1	1	1	0	1	0	0
?	?	?	?	**blau**	blau	rot	rot

An der **fünften Stelle von rechts** steht eine `1`. Der Ball an dieser Stelle muss blau sein, um von der bisherigen geraden (2) zu einer ungeraden (3) Anzahl blauer Bälle rechts ab dieser Stelle zu kommen.

0	1	1	1	0	1	0	0
?	?	?	**blau**	blau	blau	rot	rot

An der **sechsten Stelle von rechts** steht eine `1`. Der Ball an dieser Stelle muss rot sein, damit die Anzahl blauer Bälle rechts ab dieser Stelle ungerade (3) bleibt.

0	1	1	1	0	1	0	0
?	?	**rot**	blau	blau	blau	rot	rot

An der **siebten Stelle von rechts** steht erneut eine `1`. Der Ball an dieser Stelle muss rot sein, damit die Anzahl blauer Bälle rechts ab dieser Stelle ungerade (3) bleibt.

0	1	1	1	0	1	0	0
?	**rot**	rot	blau	blau	blau	rot	rot

An der **achten Stelle von rechts** (erste Stelle von links) steht eine `0`. Der Ball an dieser Stelle muss blau sein, um von der bisherigen ungeraden (3) zu einer geraden (4) Anzahl blauer Bälle rechts ab dieser Stelle zu kommen.

0	1	1	1	0	1	0	0
blau	rot	rot	blau	blau	blau	rot	rot

Zusammenfassend lässt sich sagen: Wenn es bei einem Schritt nach links einen Wechsel von einer `0` zu einer `1` oder von einer `1` zu einer `0` gibt, dann muss ein blauer Ball hinzukommen. Gibt es keinen Wechsel, ändert sich die Parität (also die Eigenschaft einer Zahl, gerade oder ungerade zu sein) der Anzahl blauer Bälle rechts ab der Stelle nicht, so dass dort ein roter Ball liegen muss.

0	1	1	1	0	1	0	0
4 blau	3 blau	3 blau	3 blau	2 blau	1 blau	0 blau	0 blau

Das ist Informatik!
Emil beschreibt die Bälle-Folge mit den Ziffern 0 und 1, und zwar so, dass man aus der Beschreibung die Bälle-Folge wieder erstellen kann. Eine Bälle-Folge und ihre Beschreibung sind also eindeutig einander zugeordnet. Für die Informatik sind solche eindeutigen Zuordnungen zwischen Folgen aus verschiedenen Elementemengen essenziell und heißen *Kodierungen.* Weil die technischen Bausteine der Computerhardware mit binären Werten arbeiten (0 und 1, `wahr` und `falsch`, …), muss es für jede Information, die von Computern verarbeitet werden soll, eine Kodierung mit binären Werten geben.

Bei der Kodierung von Information für die Speicherung in Informatiksystemen kann man versuchen, die Darstellung in binären Werten so kompakt wie möglich zu halten, um Speicherplatz zu sparen. Dann kann aus der Kodierung eine *Kompression* von Daten werden. Wenn aus dem Ergebnis der Kompression die Ausgangsinformation komplett und exakt wieder rekonstruiert werden kann, wird die Kompression als *verlustfrei* bezeichnet. In vielen Fällen sind aber auch *verlustbehaftete* Kompressionen akzeptabel. Bekannte Beispiele sind die Datenformate MP3, eine (mittlerweile veraltete) Kompression von Ton-Daten, und JPEG, eine Kompression von Bilddaten. In beiden Formaten wird unwichtige Information reduziert, nämlich vom Menschen ohnehin schlecht wahrgenommene Frequenzbereiche oder Farbwerte.

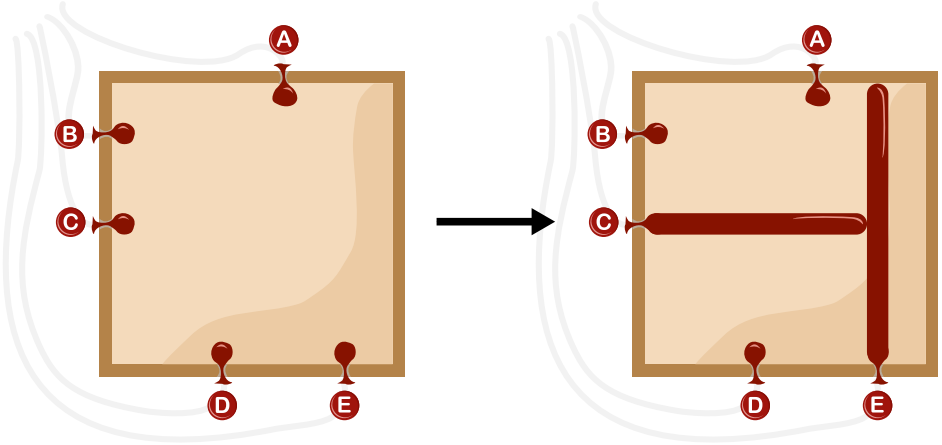
Emil bestimmt für jede Bälle-Folge eine (verlustfreie) Kodierung in binären Werten. Wäre es ihm nur darum gegangen, Folgen von roten und blauen Bällen mit binären Werten zu kodieren, hätte er einfach für die roten Bälle eine 0 und für die blauen eine 1 schreiben können. Aber dann wäre diese Biberaufgabe deutlich weniger interessant gewesen, nicht wahr?

Ballon-Maschine 1

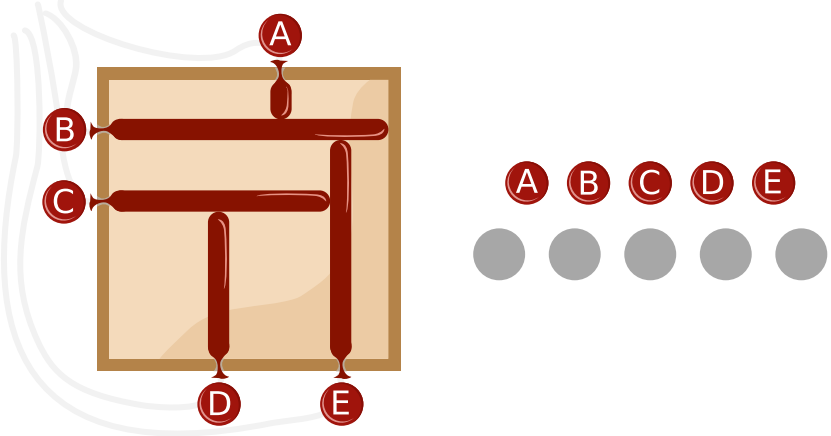
Eine Ballonmaschine kann Bilder erstellen, indem sie Ballons in einem quadratischen Rahmen aufbläst. Die Ballons sind mit den Buchstaben A, B, C, D und E beschriftet. Die Maschine liest nacheinander eine Folge von Buchstaben, von links nach rechts. Vor dem ersten Buchstaben sind alle Ballons leer.

Wenn die Maschine einen Buchstaben gelesen hat, macht sie Folgendes:
Der Ballon mit diesem Buchstaben wird aufgeblasen bis er
- entweder einen anderen Ballon berührt
- oder den gegenüberliegenden Rand des Rahmens berührt.

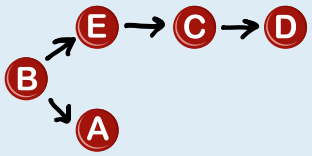
Wenn die Maschine zum Beispiel die Buchstabenfolge E C liest, macht sie dies:



Die Maschine liest eine Folge von fünf Buchstaben. Am Ende hat sie dieses Bild erstellt. Gib die Folge an:



So ist es richtig:



Aus dem Bild, das die Maschine erstellt hat, kann man einige Bedingungen für die richtige Reihenfolge beim Aufblasen der Ballons ablesen. Denn außer Ballon B wurde jeder Ballon aufgeblasen, bis er einen anderen Ballon berührt hat:

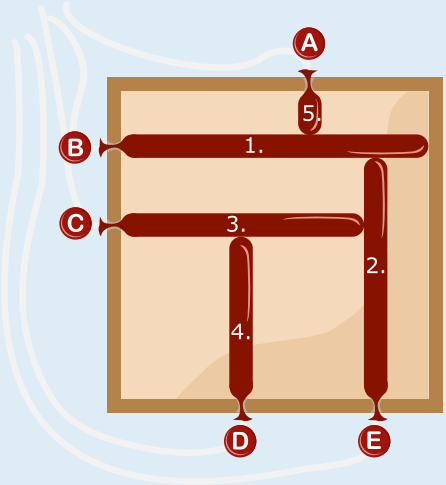
- Ballon B wurde vor A und E aufgeblasen.
- E wurde vor C aufgeblasen.
- C wurde vor D aufgeblasen.

Diese „Vorrangbeziehungen" zwischen den Ballons fasst das folgende Bild zusammen. Ein Pfeil vom (zum Beispiel) Ballon-Buchstaben B zum Ballon-Buchstaben A gibt an, dass Ballon B vor Ballon A aufgeblasen werden muss.

Jede richtige Antwort, also jede richtige Reihenfolge für das Aufblasen der Ballons, muss diese Vorrangbeziehungen einhalten. Das heißt, dass bei jeder richtigen Antwort am Anfang B sein muss. Zudem muss E vor C und C vor D sein. Das A kann jedoch irgendwann nach dem B kommen. Damit kann es genau vier mögliche richtige Folgen geben:

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

Hier sieht man, wie die Maschine mit der letzten Folge das gewünschte Bild erstellt:



Das ist Informatik!

Die Buchstabenfolge ist wie ein Programm, das die Ballonmaschine steuert. Jeder Buchstabe ist eine Anweisung, die die Maschine dazu bringt, einen Ballon aufzublasen. Die „Buchstaben-Anweisungen" können zu einer Anweisungsfolge kombiniert werden. So entsteht ein längeres Programm, mit dem die Maschine komplexere Bilder erstellen kann. Dabei ist die Reihenfolge der Anweisungen wichtig. Zum Beispiel erzeugt die Anweisungsfolge B E ein anderes Bild als die Anweisungsfolge E B. Das liegt daran, dass die Ausführung einer Anweisung Auswirkungen auf die Ausführung folgender Anweisungen haben kann. Bei der Ballonmaschine hat die Ausführung einer Anweisung insbesondere Auswirkungen auf die Länge der später aufgeblasenen Ballons.

Auch Computerprogramme bestehen aus Anweisungen, die zu größeren Programmen kombiniert werden. Die einfachste Form der Kombination nennt man in der Informatik eine *Sequenz,* also eine Folge von Anweisungen. Kompliziertere Kombinationen entstehen durch spezielle *Kontrollstrukturen* wie Wiederholungsanweisungen und bedingte Anweisungen. Auch bei Computerprogrammen kann das Ausführen einer Anweisung Auswirkungen auf die spätere Ausführung einer anderen Anweisung haben – zum Beispiel, wenn beide Anweisungen auf die gleichen Daten zurückgreifen und diese verändern. Es ist eine wichtige Aufgabe für Informatikerinnen und Informatiker, bei der Entwicklung von Programmen solche Auswirkungen sorgfältig zu beachten, damit die Programme so funktionieren wie gewünscht. So wie bei der Ballonmaschine in dieser Biberaufgabe die Reihenfolge der Anweisungen stimmen muss, damit die Maschine das gewünschte Bild erstellt.

Ballon-Maschine 2

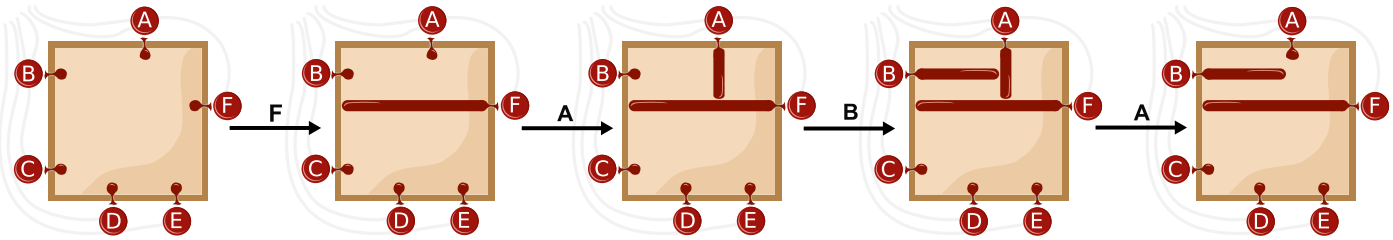
Eine Ballonmaschine kann Bilder erstellen, indem sie Ballons in einem quadratischen Rahmen aufbläst.

Die Ballons sind mit den Buchstaben A, B, C, D, E und F beschriftet.

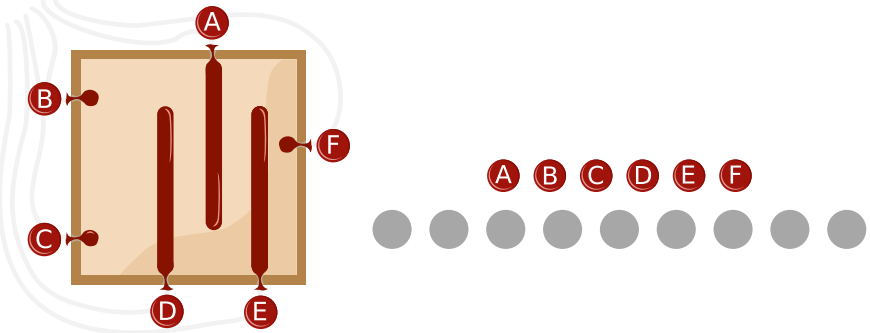
Die Maschine liest nacheinander eine Folge von Buchstaben, von links nach rechts. Vor dem ersten Buchstaben sind alle Ballons leer.

Wenn die Maschine einen Buchstaben gelesen hat, macht sie Folgendes:
- Wenn der Ballon mit diesem Buchstaben leer ist, wird er aufgeblasen, bis er entweder einen anderen Ballon oder den gegenüberliegenden Rand des Rahmens berührt.
- Wenn der Ballon mit diesem Buchstaben aufgeblasen ist, wird er entleert.

Wenn die Maschine zum Beispiel die Buchstabenfolge F A B A liest, macht sie dies:



Die Maschine liest eine Folge von neun Buchstaben. Am Ende hat sie dieses Bild erstellt. Gib die Folge an:

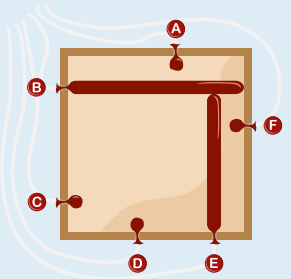
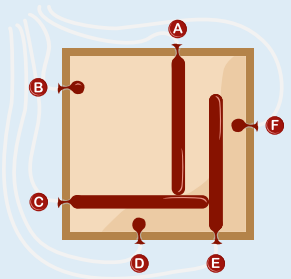
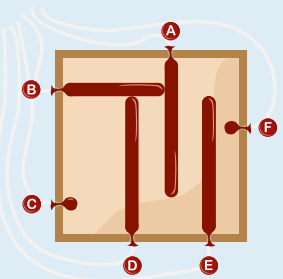


So ist es richtig:

Für die Folge der Buchstaben, die die Maschine liest (ab jetzt: Lesefolge), lassen sich einige Bedingungen feststellen:
1. Damit der Ballon von E (wir sagen ab jetzt kurz: Ballon E) eine Länge wie im Ergebnis erreicht, muss zuerst B und dann E aufgeblasen werden. Irgendwann später muss B wieder entleert werden. Die Lesefolge muss also B E B als Teil enthalten. (Ein solcher Teil kann durch andere Buchstaben unterbrochen werden.)
2. Die obige Feststellung für Ballon E gilt auch für Ballon D. Die Lesefolge muss also auch B D B als Teil enthalten.

3. Damit Ballon A eine Länge wie im Ergebnis erreicht, muss zuerst C und dann A aufgeblasen werden. Irgendwann später muss C wieder entleert werden. C A C ist also ein weiterer Teil der Lesefolge.
4. B muss entleert sein, bevor A aufgeblasen wird.
5. C muss entleert sein, bevor D aufgeblasen wird.
6. Wenn B für Ballon E aufgeblasen wird, muss A entleert sein – wie zu Beginn.
7. Damit am Ende A, D und E aufgeblasen sind, dürfen diese Buchstaben nur einmal vorkommen (oder dreimal, oder fünfmal, … – aber die Folge darf ja nur insgesamt neun Buchstaben haben). Wegen der Bedingungen 4 bis 6 muss E A D ein Teil der Lesefolge sein.

Hier ist eine Lesefolge, die alle Bedingungen erfüllt: B E B C A C B D B. Wir zeigen die Situationen nach B E, nach B E B C A und vor dem letzten B:

Eingelesene Buchstaben	B E	B E B C A	B E B C A C B D
Situation			

Durch die Bedingungen ist die Lesefolge nicht komplett festgelegt. In der obigen Folge ist an genau zwei Stellen die Reihenfolge nicht durch die Bedingungen vorgegeben: B E **B C** A **C B** D B.
Insgesamt gibt es also vier richtige Lesefolgen:
B E B C A C B D B | B E C B A C B D B | B E B C A B C D B | B E C B A B C D B

Das ist Informatik!
Die Lesefolge ist wie ein Programm, das die Ballonmaschine steuert. Jeder Buchstabe ist eine Anweisung, die die Maschine dazu bringt, einen Ballon aufzublasen oder zu entleeren. Die „Buchstaben-Anweisungen“ können zu einer Anweisungsfolge kombiniert werden. So entsteht ein längeres Programm, mit dem die Maschine komplexere Bilder erstellen kann. Dabei ist die Reihenfolge der Anweisungen wichtig. Zum Beispiel erzeugt die Anweisungsfolge B E ein anderes Bild als die Anweisungsfolge E B. Das liegt daran, dass die Ausführung einer Anweisung Auswirkungen auf die Ausführung folgender Anweisungen haben kann. Bei der Ballonmaschine hat die Ausführung einer Anweisung insbesondere Auswirkungen auf die Länge der später aufgeblasenen Ballons.

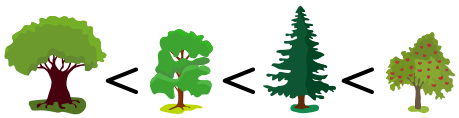
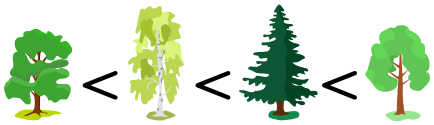
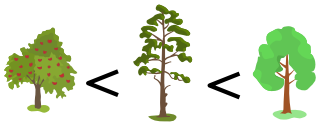
Auch Computerprogramme bestehen aus Anweisungen, die zu größeren Programmen kombiniert werden. Die einfachste Form der Kombination nennt man in der Informatik eine Sequenz, also eine Folge von Anweisungen. Kompliziertere Kombinationen entstehen durch spezielle Kontrollstrukturen wie Wiederholungsanweisungen und bedingte Anweisungen. Jede „Buchstaben-Anweisung“ der Ballonmaschine in dieser Biberaufgabe ist eine bedingte Anweisung. Im Stil einer Computer-Programmiersprache kann man diese Anweisung so formulieren – unter der Annahme, dass ein Ballon nur zwei Zustände haben kann (leer und nicht leer bzw. aufgeblasen):

```
Wenn der Ballon zu diesem Buchstaben leer ist, dann:
    Blase den Ballon auf, bis …
Sonst:
    Entleere den Ballon.
```

Es ist eine wichtige Aufgabe für Informatikerinnen und Informatiker, bei der Entwicklung von Programmen die Auswirkungen von Anweisungen sorgfältig zu beachten, damit die Programme so funktionieren wie gewünscht. Bei bedingten Anweisungen ist das besonders schwierig, da deren Ausführung von der Auswertung der in der Anweisung genannten Bedingung abhängt.

Baumtouren

Försterin Flora bietet Baumtouren an. Dabei stellt sie einige Bäume im Wald näher vor. Von drei früheren Touren weiß sie, wie beliebt die Bäume bei ihren Gästen waren.

Tour 1	
Tour 2	
Tour 3	

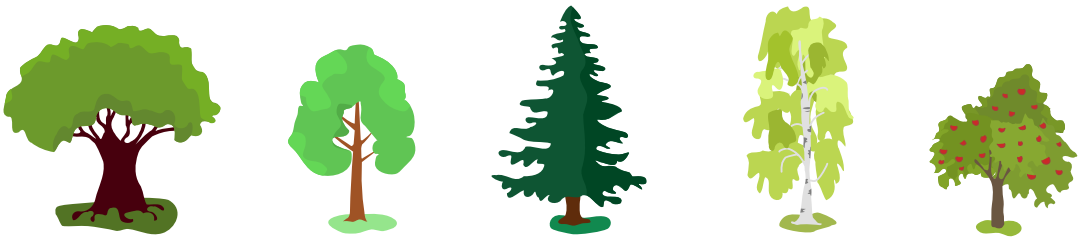
Baum 1 < Baum 2 bedeutet, dass Baum 1 weniger beliebt war als Baum 2.

Bei ihren nächsten Touren möchte Flora beliebtere Bäume erst später vorstellen – das Beste kommt zum Schluss. Sie will die Bäume immer in einer guten Reihenfolge vorstellen, die mit der Beliebtheit der Bäume bei allen drei früheren Touren übereinstimmt.

Ein Beispiel: Falls Flora die Bäume  und  in einer guten Reihenfolge vorstellen will, dann muss sie  vor  vorstellen, weil auf Tour 1  weniger beliebt war als  .

Auf ihrer nächsten Tour möchte Flora die Bäume  ,  ,  ,  und  vorstellen.

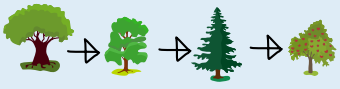
Bestimme eine gute Reihenfolge dieser Bäume!

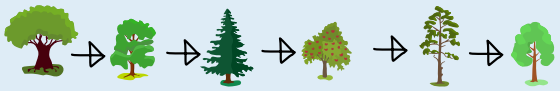


So ist es richtig:

Eine gute Reihenfolge muss mit der Beliebtheit der Bäume bei den drei früheren Touren übereinstimmen. Wenn wir die „Beliebtheitsordnungen“ der drei Touren zu einer Darstellung zusammenfassen, kann sich die Reihenfolge einer neuen Tour nach dieser Darstellung richten.

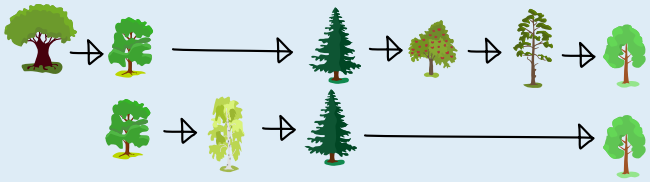
Diese Darstellung soll ein Diagramm mit Bäumen und Pfeilen sein. Darin werden zwei Bäume immer dann durch einen Pfeil verbunden, wenn ein Baum vor einem anderen Baum vorgestellt werden muss. Für jede frühere Tour lässt sich so ein Diagramm einfach erstellen. Hier das Diagramm für Tour 1:



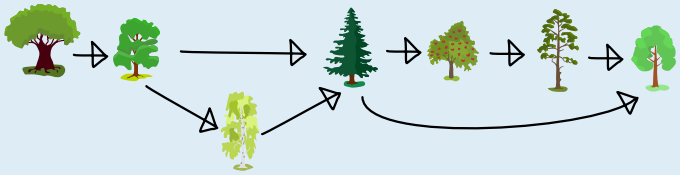
Das können wir leicht mit dem Diagramm für Tour 3 kombinieren, weil die beiden Touren nur den Baum  gemeinsam haben: 

Dieses Diagramm müssen wir nun noch mit dem für Tour 2 (im folgenden Bild unten) kombinieren.

Drei von vier Bäumen aus Tour 2 sind bereits vorhanden, nur  kommt neu hinzu:



Wie oben kommt im kombinierten Diagramm jeder Baum nur einmal vor, aber kein Pfeil geht verloren:



Mithilfe des Diagramms können wir die Aufgabe nun ganz einfach lösen: Die Bäume der neuen Tour besuchen wir im Diagramm von links nach rechts, entlang der Pfeile, und erhalten so eine eindeutige gute Reihenfolge, die mit der Beliebtheit der Bäume bei allen früheren Touren übereinstimmt:

Das ist Informatik!

In dieser Aufgabe soll eine Menge von Bäumen nach aufsteigender Beliebtheit sortiert werden. Es gibt Informationen über einige Bäume, die paarweise in ihrer Beliebtheit verglichen wurden. Für andere Baumpaare wie beispielsweise  und  gibt es keinen direkten Vergleich. Dennoch können auch diese Bäume verglichen werden. Dabei sind allerdings zwei Eigenschaften von zentraler Bedeutung:

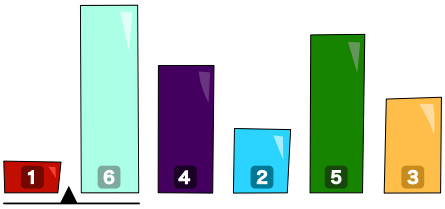
1. Der Beliebtheitsvergleich ist transitiv. Das heißt, falls Baum 1 < Baum 2 und Baum 2 < Baum 3, dann ist auch Baum 1 < Baum 3.
2. Die drei früheren Touren sind nicht im Widerspruch zueinander: Falls in irgendeiner der drei Touren Baum 1 < Baum 2, dann gibt es keine andere Tour mit Baum 2 < Baum 1.

Dank dieser Eigenschaften bilden die Beliebheitsvergleiche im mathematischen Sinne eine *Ordnung.* Es mag zunächst den Eindruck haben, dass diese Ordnung *partiell* ist, da nicht alle Bäume paarweise miteinander verglichen sind – wie eben  und  . Dennoch lassen sich die Bäume in dieser Biberaufgabe in einer Folge anordnen, die mit der Beliebtheitsvergleichs-Ordnung übereinstimmt; eine solche Folge nennen wir eine *topologische Sortierung.* Die Informatik kennt einen relativ einfachen Algorithmus, der eine topologische Sortierung bestimmen kann, wenn es in den bekannten paarweisen Vergleichen keine Widersprüche gibt.

Bei nähererer Betrachtung kann man feststellen, dass die Beliebtheit der Bäume dank der Transitivität eine *totale Ordnung* ist. Damit ist die topologische Sortierung aller Bäume eindeutig – was auch die Eindeutigkeit der richtigen Antwort zusätzlich beweist.

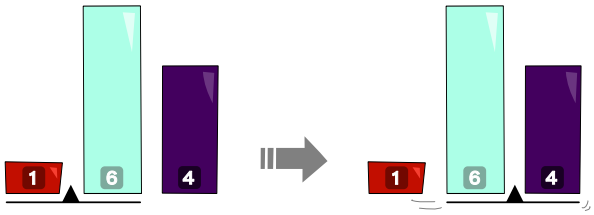
Baustein-Maschine 1

Eine Maschine kann sechs unterschiedlich hohe Bausteine bewegen. Dabei benutzt sie eine Markierung (▲), die in der Regel zwischen zwei Steinen steht. Am Anfang stehen Steine und Markierung so:

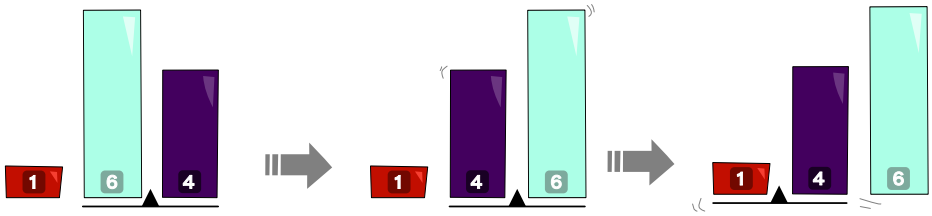


Die Maschine arbeitet dann nach diesen Vorschriften:

- Wenn der Stein links von der Markierung kleiner ist als der Stein rechts davon, geht die Markierung nach rechts.



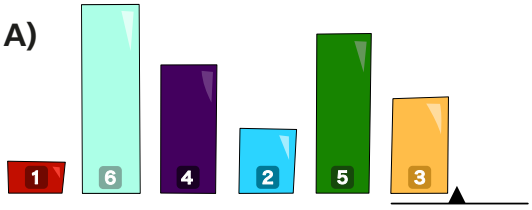
- Wenn der Stein links von der Markierung größer ist als der Stein rechts davon, werden die Steine vertauscht. Danach geht die Markierung nach links – aber nur, wenn sie dann immer noch zwischen zwei Steinen steht.



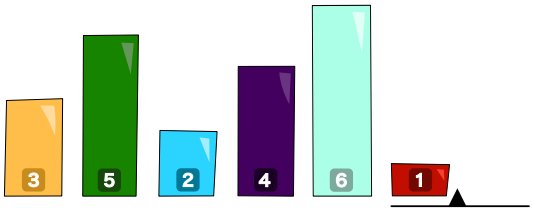
Die Maschine arbeitet so lange, bis die Markierung ganz rechts neben den Steinen steht. Dann stoppt die Maschine.

Wie stehen die Steine, nachdem die Maschine stoppt?

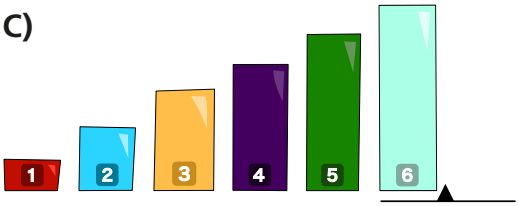
A)



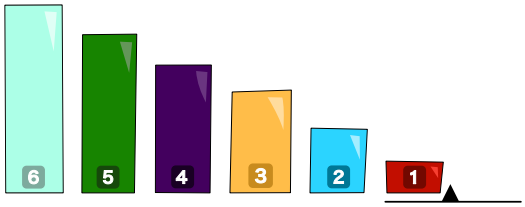
B)



C)



D)



Antwort C ist richtig:

Natürlich könnte man die Arbeit der Maschine Schritt für Schritt nachvollziehen, für diesen speziellen Fall. Es ist aber einfacher zu erkennen, dass die Maschine grundsätzlich Bausteine der Größe nach sortiert.

Am Anfang steht links von der Markierung nur ein Stein. Man kann sagen, dass dieser eine Stein der Größe nach sortiert ist. Insbesondere sind also alle Steine links von der Markierung der Größe nach sortiert. Man kann überlegen, dass die weiteren Schritte der Maschine an dieser Eigenschaft nichts ändern.

Die Markierung geht nur nach rechts, wenn der Baustein links von der Markierung kleiner ist als der Baustein rechts davon. Sind vor einem solchen *Rechtsschritt* alle Bausteine links von der Markierung der Größe nach sortiert, ist das nach dem Schritt immer noch der Fall.

Wenn das anders ist, also der linke Baustein größer ist als der rechte, wechselt der kleinere Baustein nach links und der größere nach rechts. Der kleinere Baustein, der nun links von der Markierung steht, könnte kleiner sein als einige Steine weiter links davon. Deshalb geht die Markierung auch nach links. Sind vor einem solchen *Linksschritt* alle Bausteine links von der Markierung der Größe nach sortiert, ist das nach dem Schritt immer noch der Fall.

Ist der vor einem Linksschritt kleinere Baustein auch kleiner als die Steine weiter links davon, folgen weitere Linksschritte. Dadurch wird der kleinere Baustein solange weiter nach links getauscht, bis er größer ist als der Baustein links von ihm (und damit als alle anderen links von ihm) – oder bis er ganz links steht, wenn er der kleinste Baustein ist. Dann folgen Rechtsschritte, bis ein weiterer kleinerer Baustein vorkommt, und dann wieder Linksschritte.

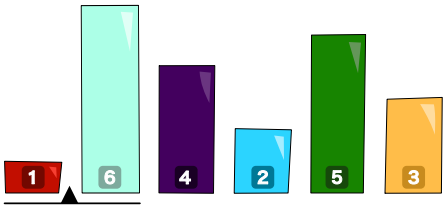
Das geht so lange, bis bei den Rechtsschritten kein kleinerer Baustein mehr vorkommt. Dann macht die Markierung so lange Rechtsschritte, bis sie rechts neben den Steinen steht, und die Maschine stoppt. Dann stehen alle Bausteine links von der Markierung, und alle sind der Größe nach aufsteigend sortiert, wie in Antwort C.

Das ist Informatik!
Computer verarbeiten Daten – häufig sehr viele Daten. Dabei ist es wichtig, dass in den Daten Ordnung herrscht. In der Informatik sagt man statt ordnen meist sortieren: Zum Beispiel können Zahlen nach aufsteigender Größe, Daten über Personen nach dem Geburtsdatum oder Daten über Bücher nach der ISBN-Nummer sortiert werden. Am Ende eines Sortiervorgangs stehen die Daten in der gewünschten Reihenfolge. Weil Computer ständig große Datenmengen sortieren müssen, beschäftigen sich Informatikerinnen und Informatiker unter anderem mit möglichst effizienten Sortierverfahren, die wenig Zeit und auch wenig Speicherplatz für ihre Arbeit benötigen.

Das Sortierverfahren, das die Baustein-Sortier-Maschine in dieser Biberaufgabe verwendet, heißt Gnomesort. Es ähnelt dem deutlich bekannteren Bubblesort (auch „Sortieren durch Aufsteigen“). Gnomesort ist, wie Bubblesort auch, nicht besonders effizient – zumindest dann, wenn die Daten stark durcheinander sind. Sind die Daten hingegen bereits annähernd sortiert, sind beide Verfahren aber sehr schnell – und vielleicht schneller als Sortierverfahren wie Quicksort oder Mergesort, die im Allgemeinen deutlich effizienter sind. Deshalb, und weil Gnomesort einfach zu verstehen und zu programmieren ist, ist das Verfahren für bestimmte Anwendungen durchaus interessant.

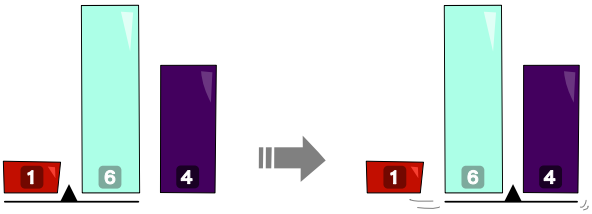
Baustein-Maschine 2

Eine Maschine kann sechs unterschiedlich hohe Bausteine bewegen. Dabei benutzt sie eine Markierung (▲), die in der Regel zwischen zwei Steinen steht. Am Anfang stehen Steine und Markierung in einer Startkonfiguration, zum Beispiel so:

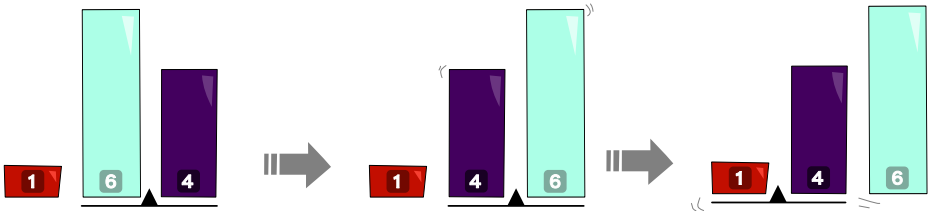


Die Maschine arbeitet dann nach diesen Vorschriften:

- Wenn der Stein links von der Markierung kleiner ist als der Stein rechts davon, macht die Markierung einen Schritt nach rechts.



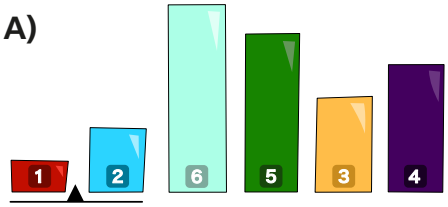
- Wenn der Stein links von der Markierung größer ist als der Stein rechts davon, werden die Steine vertauscht. Danach macht die Markierung einen Schritt nach links – aber nur, wenn sie dann immer noch zwischen zwei Steinen steht.



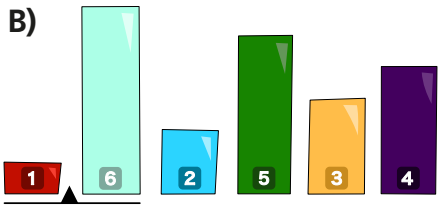
Die Maschine arbeitet so lange, bis die Markierung ganz rechts neben den Steinen steht. Dann stoppt die Maschine.

Je nach Startkonfiguration macht die Markierung unterschiedlich viele Schritte.
Bei welcher dieser Startkonfigurationen macht die Markierung die wenigsten Schritte?

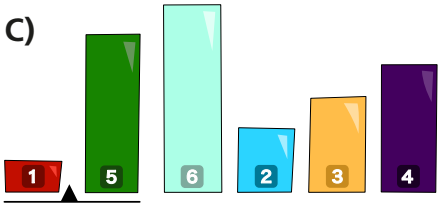
A)



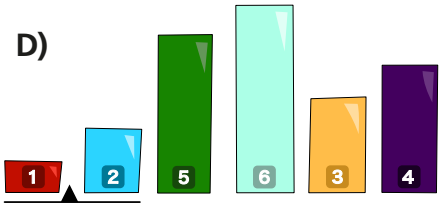
B)



C)



D)



Antwort D ist richtig:

Um die richtige Antwort zu bestimmen, könnte man für jede Startkonfiguration die Arbeit der Maschine bis zum Stopp simulieren und dabei die Schritte der Markierung zählen. Einfacher ist, die vier Startkonfigurationen bzgl. der Anzahl der Schritte zu vergleichen, ohne diese zu zählen.

Zur Vereinfachung stellen wir die Konfigurationen nur durch Zahlen (die Höhen der Bausteine) dar, mit der Markierung zwischen zwei Zahlen. In dieser Darstellung sehen die Startkonfigurationen so aus:

A	B	C	D
1▲2 6 5 3 4	1▲6 2 5 3 4	1▲5 6 2 3 4	1▲2 5 6 3 4

Wenn die Maschine mit der Konfiguration von Antwort B (kurz: Konfiguration B) startet, ist nach zwei Schritten Konfiguration A erreicht:
1▲6 2 5 3 4 → 1 6▲2 5 3 4 → 1▲2 6 5 3 4

Für Konfiguration B macht die Markierung also mehr Schritte als für Konfiguration A. Damit kann Antwort B nicht richtig sein.

Wenn die Maschine mit Konfiguration A startet, macht die Markierung zuerst diese drei Schritte:
1▲2 6 5 3 4 → 1 2▲6 5 3 4 → 1 2 6▲5 3 4 → 1 2▲5 6 3 4

Die somit erreichte Konfiguration wird von Konfiguration D aus in einem Schritt erreicht:
1▲2 5 6 3 4 → 1 2▲5 6 3 4.

Für Konfiguration A macht die Markierung also mehr Schritte als für Konfiguration D. Damit kann Antwort A nicht richtig sein.

Wenn die Maschine mit Konfiguration C startet, ist nach vier Schritten Konfiguration D erreicht.
1▲5 6 2 3 4 → 1 5▲6 2 3 4 → 1 5 6▲2 3 4 → 1 5▲2 6 3 4 → 1▲2 5 6 3 4

Also macht die Markierung auch für Konfiguration C mehr Schritte als für Konfiguration D. Insgesamt macht sie für Konfiguration D die wenigsten Schritte.

Das ist Informatik!
Computer verarbeiten Daten – häufig sehr viele Daten. Dabei ist es wichtig, dass in den Daten Ordnung herrscht. In der Informatik sagt man statt ordnen meist sortieren: Zum Beispiel können Zahlen nach aufsteigender Größe, Daten über Personen nach dem Geburtsdatum oder Daten über Bücher nach der ISBN-Nummer sortiert werden. Am Ende eines Sortiervorgangs stehen die Daten in der gewünschten Reihenfolge. Weil Computer ständig große Datenmengen sortieren müssen, beschäftigen sich Informatikerinnen und Informatiker unter anderem mit möglichst effizienten Sortierverfahren, die wenig Zeit und auch wenig Speicherplatz für ihre Arbeit benötigen.

Wenn die Maschine in dieser Biberaufgabe stoppt, sind die Steine der Größe nach sortiert. Das Verfahren, das die „Baustein-Sortier-Maschine“ verwendet, heißt *Gnomesort.* Es ähnelt dem deutlich bekannteren *Bubblesort* (auch „Sortieren durch Aufsteigen“). Gnomesort ist, wie Bubblesort auch, nicht besonders effizient – insbesondere, wenn die Daten stark durcheinander sind. Für die Startkonfiguration, bei denen die Steine in der falschen Reihenfolge stehen (im Beispiel dieser Biberaufgabe also: 6▲5 4 3 2 1), würde die Maschine (genau genommen: die Markierung) mehr Schritte machen als für jede andere Startkonfiguration.

Sind die Daten aber bereits annähernd sortiert, sind Gnomesort wie auch Bubblesort sehr schnell – und vielleicht schneller als Verfahren wie Quicksort oder Mergesort, die im Allgemeinen deutlich effizienter, aber auch deutlich komplizierter sind. Gnomesort hingegen ist einfach zu verstehen und zu programmieren und sortiert zudem „in-place“: Für die zu sortierenden Daten wird kein zusätzlicher Speicherplatz benötigt. Wegen dieser positiven Eigenschaften ist das Verfahren für bestimmte Anwendungen durchaus interessant.

Berukone

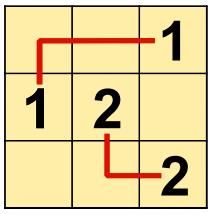
Lasst uns Berukone spielen!

in Berukone-Rätsel besteht aus einem Raster mit Zahlen auf den Feldern.
Zwei gleiche Zahlen bilden ein Paar.

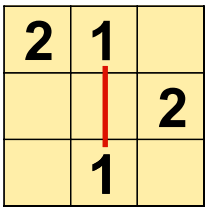
Um das Rätsel zu lösen, musst du für alle Paare die beiden Zahlen durch eine Linie verbinden. Diese Verbindungslinie muss waagrecht oder senkrecht von Feld zu Feld gehen, darf sich aber auf einem Feld um 90 Grad drehen. Eine Linie darf nicht durch eine andere Zahl und nicht durch eine andere Linie gehen.

Leider gibt es Berukone-Rätsel, die nicht gelöst werden können:

Dieses Berukone kann gelöst werden.



Dieses Berukone kann nicht gelöst werden.



Genau eines dieser Berukones kann nicht gelöst werden. Welches?

A)

			3
3			
	2	1	2
1			

B)

3			3
		1	
	2		2
1			

C)

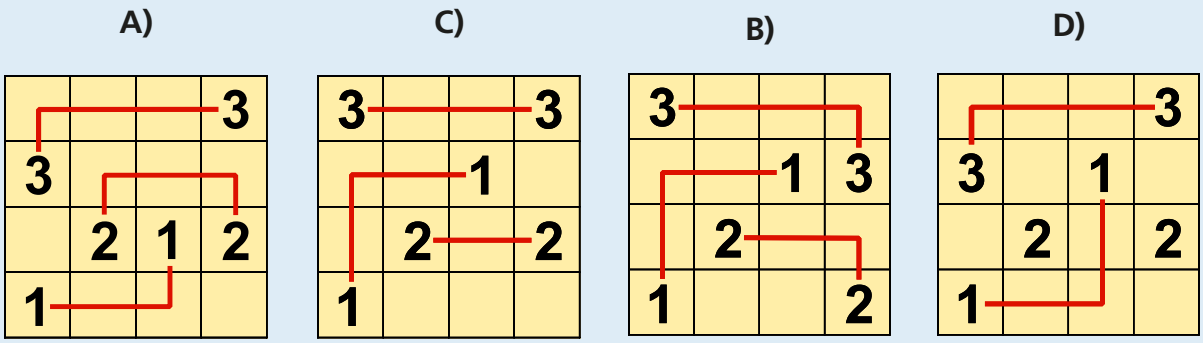
3			
		1	3
	2		
1			2

D)

			3
3		1	
	2		2
1			

Antwort D ist richtig:

Berukone D ist das einzige der vier Berukone-Rätsel, das nicht gelöst werden kann. So kannst du die Rätsel A, B und C lösen und versuchen Rätsel D zu lösen:



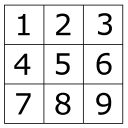
Im Rätsel D gibt es keine Möglichkeit, für alle Paare die beiden Zahlen nach den Berukone-Regeln zu verbinden. Insbesondere kann es nur für eines der beiden Paare aus 1en und 2en eine regelgemäße Verbindungslinie geben: Die einzige Linie, mit der man die 1en verbinden kann, ohne dass die Linie durch eine andere Zahl geht, ist oben dargestellt. Alle möglichen Linien zwischen den 2en, die nicht durch mindestens eine andere Zahl gehen, würden die Linie zwischen den 1en kreuzen. Es gibt also keine Möglichkeit mehr, auch noch die 2en mit einer Linie zu verbinden, ohne die Regeln zu verletzen.

Das ist Informatik!
Für die kleinen Berukone-Rätsel in dieser Biberaufgabe ist relativ einfach zu erkennen, ob sie gelöst werden können oder nicht und wie sie gelöst werden können. Aber stell dir größere Rätsel vor, mit viel mehr Zahlen: Dann ist es vielleicht nicht mehr so einfach, eine Lösung zu finden. Es könnte viele Möglichkeiten geben, die Paar-Zahlen zu verbinden, und es würde viel Zeit in Anspruch nehmen, alle auszuprobieren.

Aber du kannst versuchen, das schlauer anzugehen. Du kannst Schritt für Schritt vorgehen, wenn du ein Rätsel löst: Verbinde die Paare nacheinander oder versuche, die Rasterfelder nacheinander zu füllen. Sobald du voraussehen kannst, dass dein aktueller Weg nicht mehr zu einer Lösung führt, war mindestens ein vorheriger Schritt falsch. Dann kannst du den letzten Schritt rückgängig machen und vielleicht sogar noch mehr Schritte, bis du zu einer Stelle kommst, an der ein alternativer Schritt möglich ist, der die bessere Wahl sein könnte. Das heißt, du gehst auf der Spur (engl.: *track*), die du mit deinen bisherigen Schritten gelegt hast, zurück (engl.: *back*), um dann einen anderen Weg zu verfolgen.

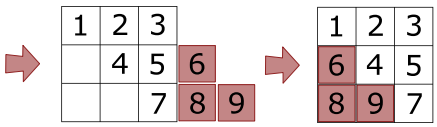
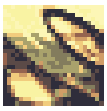
Diese Methode heißt in der Informatik *Backtracking.* Sie wird oft verwendet, um Computerprogramme Probleme – ob reale Probleme oder nur Rätsel – automatisch lösen zu lassen. Backtracking-Algorithmen funktionieren wie oben beschrieben: Sie versuchen, Lösungen Schritt für Schritt zu konstruieren und Schritte rückgängig zu machen, wenn sie voraussehen können, dass der aktuelle Weg nicht zu einer Lösung führt. Ihre Fähigkeit, (a) gute Entscheidungen beim nächsten Schritt zu treffen und (b) vorauszusehen, ob der aktuelle Weg zu einer Lösung führen kann oder nicht, ist entscheidend für ihren Erfolg. Backtracking-Algorithmen müssen oft mit schwierigen Problemen umgehen, für die spezifische Lösungsansätze nicht bekannt sind. Im Jahr 2010 fanden japanische Forscher heraus, dass das Lösen von Arukone-Rätseln (eine Variante von Berukone, auch bekannt als Numberlink, bei der in einer Lösung kein Feld frei bleiben darf) zu den schwierigsten bekannten Problemen in der Informatik gehört.

Bilder verschlüsseln

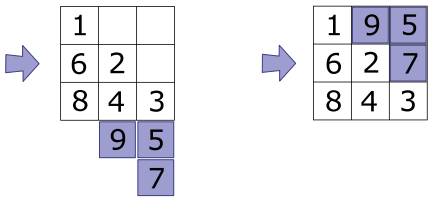
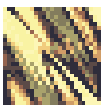
Beschreibung	Beispiel 3x3 Pixel	Bild 25x25 Pixel
Ein Bild ist ein Rechteck, das sich aus Zeilen und Spalten von Pixeln (Farbpunkte) zusammensetzt.		

Leo hat sich ein Verfahren zur Verschlüsselung von Bildern überlegt.
Er verwendet dabei diese zwei Operationen:

Operation H (für horizontal)

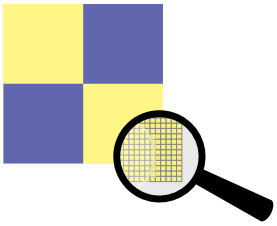
In Zeile 1 bleiben die Pixel unverändert. In Zeile 2 rückt jedes Pixel um 1 nach rechts. In Zeile 3 rückt jedes Pixel um 2 nach rechts. *Jedes Pixel in der n-ten Zeile rückt um n-1 Spalten nach rechts.* Pixel, die dabei über den rechten Bildrand hinausrücken, werden in derselben Zeile links wieder eingefügt. Die Pixelreihenfolge wird dabei nicht verändert.		

Operation V (für vertikal)

Jedes Pixel in der n-ten Spalte rückt um n-1 Zeilen nach unten. Pixel, die über den unteren Bildrand hinaurücken, werden in derselben Spalte oben wieder eingefügt.		

Die Operationen können hintereinander durchgeführt werden, als Folge.
Im Beispiel hat Leo das Bild (25x25 Pixel) mit der Folge **HV** verschlüsselt.
Unmittelbar oberhalb rechts siehst du das Ergebnis.

Leo verschlüsselt das folgende Bild (1000x1000 Pixel) mit der Folge **VH:**



Wie sieht das Ergebnis aus?

A)


B)


C)

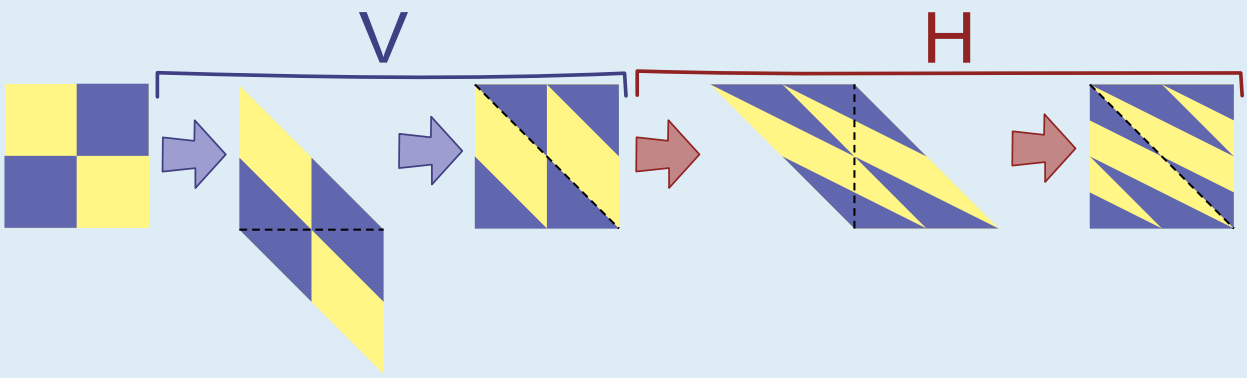

D)


E)


Antwort E ist richtig:

Im Beispiel deutet die Verzerrung des Bildes der Mona Lisa darauf hin, dass die Operation **H** geometrisch eine horizontale bewirkt und die Operation V eine vertikale *Scherung.*

Das Bild mit den vier farbigen Quadraten wird durch die Operation **V** zunächst nach unten geschert. Das Ergebnis ist ein Parallelogramm mit einem verzerrten Bild. Das überhängende Dreieck am unteren Rand wird oben eingefügt (durch *Parallelverschiebung*), so dass wieder ein rechteckiges Bild entsteht. In ähnlicher Weise erfolgt danach durch die Operation H eine Scherung nach rechts. Der Überhang wird links eingefügt:



Dieses Ergebnis der Verschlüsselung mit der Operationsfolge **VH** entspricht dem Bild aus Antwort E.

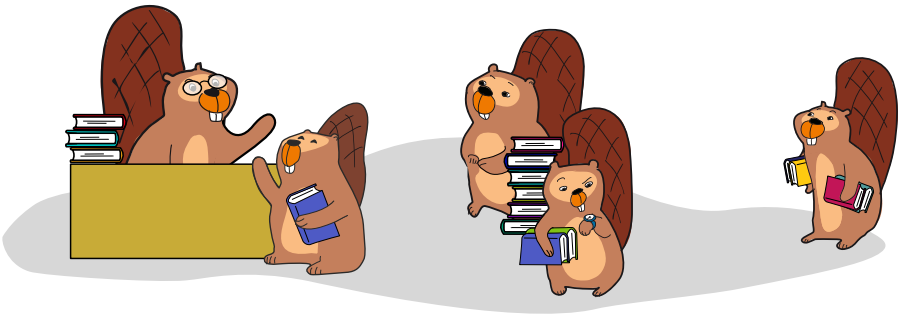
Das ist Informatik!
Bei der Übertragung von Daten kann es wichtig sein, die Daten zu verschlüsseln. Sender und Empfänger der Daten verabreden dazu, welches Verfahren zur Verschlüsselung (und zur Entschlüsselung) verwendet wird. Wenn man kurz die Daten mit *D* und das Verschlüsselungsverfahren mit *K* (für „Krypto“) bezeichnet, kann man sagen, dass das Ergebnis der Anwendung von *K* auf *D* übertragen wird, also *K(D)*. Der Empfänger muss dann den Entschlüsselungsteil des Verfahrens *(K')* anwenden und hat dann *K'(K(D)) = D*. So kann jemand, der die Übertragung von *K(D)* beobachtet, die originalen Daten *D* nicht ohne Weiteres ermitteln.

Das Verschlüsselungsverfahren für Bilder aus dieser Biberaufgabe, mit den Operationen **H** und **V**, hat den Vorteil, dass es einfach angewendet werden kann. Wenn der Empfänger den bei der Verschlüsselung verwendeten Schlüssel, hier also die Operationsfolge (z. B. **HV**) kennt, kann er die verschlüsselten Daten leicht entschlüsseln. Dennoch lässt das Ergebnis der Verschlüsselung das Original mit dem Auge nicht mehr erkennen. Bei digitaler Übertragung der Daten kann aber mit Hilfe von Computern gearbeitet und das Original relativ leicht entziffert (also: ohne Kenntnis des Schlüssels ermittelt) werden.

Je höher der Sicherheitsanspruch an ein Verschlüsselungsverfahren ist, desto größer ist oft der Aufwand, der zur Ver- und Entschlüsselung betrieben werden muss (z. B. beim Rechenaufwand und bei der Übermittlung eines komplexen geheimen Schlüssels zwischen Sender und Empfänger der verschlüsselten Information). Die Informatik kennt aber moderne Verfahren, die wenig Aufwand für Ver- und Entschlüsselung erfordern, die aber durch ihre besonderen (meist mathematischen) Eigenschaften eine Entzifferung übertragener Daten praktisch unmöglich machen.

Der Nächste bitte!

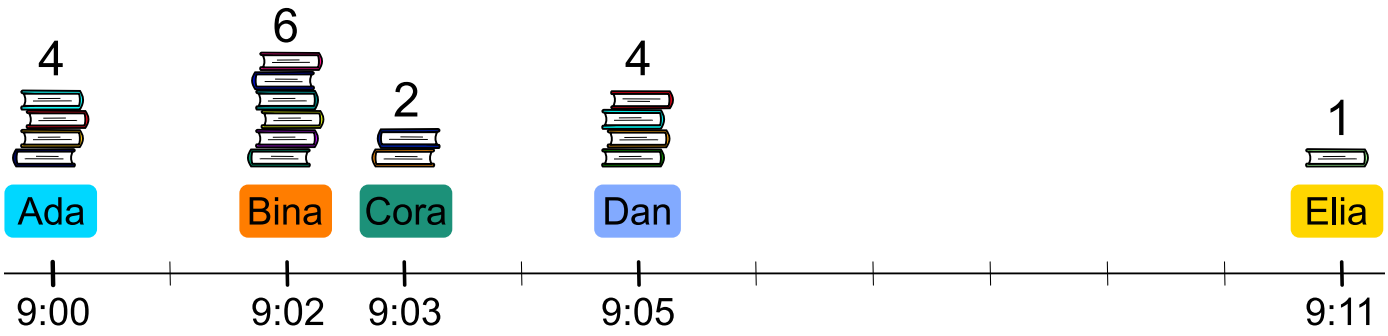
Die Biber in Holzdorf sind fleißige Leserinnen und Leser. In der Bibliothek müssen die Biber deshalb oft warten, wenn sie ihre Bücher zurückgeben wollen.



Wenn ein Biber an der Reihe ist, gibt er alle mitgebrachten Bücher zurück. Die Rückgabe eines Buchs dauert immer genau eine Minute. Ist ein Biber fertig, kommt der nächste Biber aus dem Wartebereich an die Reihe. Das ist immer der Biber mit den wenigsten Büchern.

An einem Morgen kommen nach und nach 5 Biber und wollen ihre Bücher zurückgeben.

Das Bild zeigt für jeden Biber,
- wann er im Wartebereich ankommt und
- wie viele Bücher er mitgebracht hat.



Ada kommt als erste und kann sofort ihre 4 Bücher zurückgeben.

In welcher Reihenfolge geben die Biber ihre Bücher zurück?

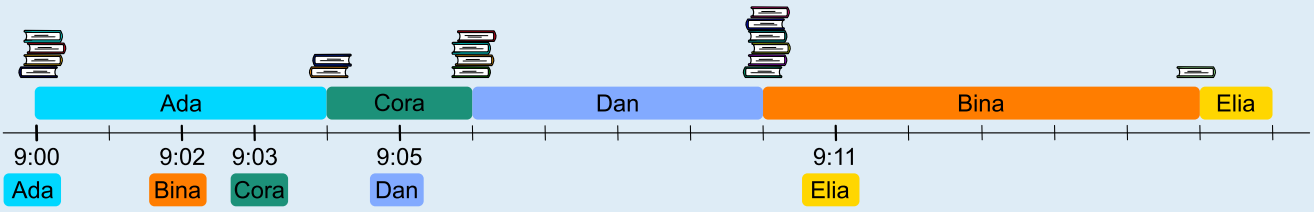
Ada Bina Cora Dan Elia

1. ____ 2. ____ 3. ____ 4. ____ 5. ____

So ist es richtig:

1. Ada 2. Cora 3. Dan 4. Bina 5. Elia

Die Abbildung zeigt, wie die Bücher zurückgegeben werden.



- Zuerst ist Ada an der Reihe, um 09:00 Uhr. Es dauert 4 Minuten, bis sie ihre 4 Bücher zurückgegeben hat. Es ist nun 09:04 Uhr. Erst dann kommt der nächste Biber an die Reihe.
- Inzwischen sind Bina und Cora angekommen und warten. Cora hat 2 Bücher und Bina hat 6. Deshalb ist Cora als nächste an der Reihe. Nach 2 Minuten, um 09:06 Uhr, hat sie ihre Bücher zurückgegeben.
- Inzwischen ist Dan gekommen und wartet zusammen mit Bina. Dan hat 4 Bücher und Bina hat 6. Dan hat am wenigsten Bücher und ist als nächster dran. Nach weiteren 4 Minuten, um 9:10 Uhr, ist auch Dan fertig.
- Nun ist endlich Bina an der Reihe. Bina ist um 9:10 Uhr der einzige Biber im Wartebereich, deshalb hat sie nun die wenigsten Bücher.
- Inzwischen ist Elia gekommen. Wenn Bina um 9:16 fertig ist, ist er der einzige Biber im Wartebereich. Er kommt dann als letzter an die Reihe.

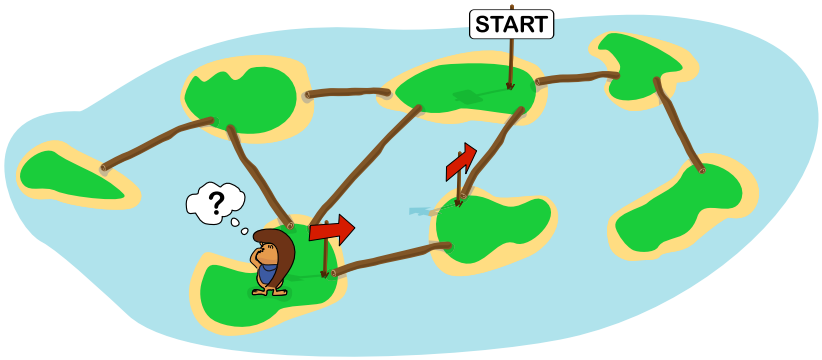
Das ist Informatik!
Während ein Computer läuft, müssen viele Aufgaben bearbeitet werden. Für jede Aufgabe wird ein Prozess gestartet, der die Aufgabe bearbeitet. Hier kommt der *Scheduler* ins Spiel. Das ist ein Programm, das zum Betriebssystem eines Computers gehört. Der Scheduler steuert den Ablauf der Prozesse. Er bestimmt, wann und für wie lange ein Prozess von der Zentraleinheit des Computers (CPU) ausgeführt wird. Wenn die CPU gut ausgelastet ist, kann es passieren, dass Prozesse warten müssen. Die Informatik kennt für diesen Fall unterschiedliche Strategien, nach denen Scheduler den nächsten Prozess zur Bearbeitung auswählen; unter anderem:

- *Priority Scheduling:* Der Prozess mit der höchsten Priorität wird als nächster (vollständig) ausgeführt.
- *First Come First Serve (FCFS):* Der Prozess, der als erster in die Warteschlange gekommen ist, wird als erster (vollständig) ausgeführt. Nach diesem Prinzip funktionieren die meisten Warteschlangen im Alltag.
- *Round Robin:* Die wartenden Prozesse werden reihum nur für eine bestimmte Zeit ausgeführt. Wenn ein Prozess dann noch nicht fertig ist, wird er unterbrochen und muss wieder an das Ende der Warteschlange. Dann ist der nächste Prozess an der Reihe.
- *Shortest Job First (SJF)*: Der Prozess mit der kürzesten erwarteten Bearbeitungszeit wird als erster (vollständig) ausgeführt.

In dieser Biberaufgabe verwendet die Bibliothek in Holzdorf bei der Buchrückgabe also die Strategie Shortest Job First: Der Biber mit den wenigsten Büchern (und deshalb der kürzesten Bearbeitungszeit) wird als nächster bedient.

Erkundung

Harry erkundet eine Gruppe von Inseln, die durch Baumstämme verbunden sind. Er hat verschiedene Schilder dabei: ein START-Schild und Pfeile. Wenn Harry sich auf einer Insel befindet, kann er alle damit verbundenen Inseln sehen und feststellen, ob auf ihnen ein Schild steht.



Auf dem Bild hat Harry bereits drei Inseln betreten. Er bewegt sich nach der folgenden Anleitung. Darin ist die „aktuelle Insel" die Insel, auf der sich Harry gerade befindet, und eine „Nachbarinsel" ist mit der aktuellen Insel über einen Baumstamm verbunden. Die Anleitung funktioniert, wenn Harry mit ihrer Hilfe jede Insel wenigstens einmal betritt.

Leider hat die Anleitung noch Lücken.

Fülle die Lücken mit passenden Textbausteinen, so dass die Anleitung funktioniert.

Betritt eine beliebige Insel, setze dort das START-Schild und erkunde diese Insel.

Jedes Mal, wenn du eine Insel erkundest, tue Folgendes:

Wenn auf der aktuellen Insel kein Schild steht, dann:
Setze einen Pfeil; der Pfeil zeigt zu der Insel, von der aus du die aktuelle Insel betreten hast.

Wenn es eine Nachbarinsel gibt, auf der [ ] steht, dann:
Betritt diese Insel und erkunde sie.

Sonst:

Wenn auf der aktuellen Insel ein Pfeil steht, dann:
Betritt diejenige Insel, [ ],
und erkunde sie.

Sonst:
Bleibe stehen. Du hast jede Insel wenigstens einmal betreten.

kein Schild

ein Pfeil

ein START-Schild

auf die der Pfeil zeigt

von der du gerade gekommen bist

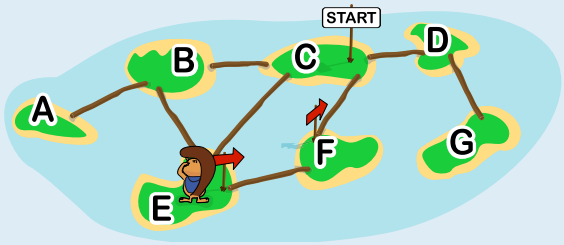
So ist es richtig:

Wir besprechen die beiden Lücken der Anleitung nacheinander.

Erste, gelbe Lücke:
Mache dir zunächst Folgendes klar: Harry stellt immer dann ein Schild auf, wenn er eine Insel ohne Schild betritt. Deshalb steht auf allen Inseln, die Harry bereits betreten hat, genau ein Schild und auf allen Inseln, die er noch nicht betreten hat, kein Schild.

Der Text in dieser Lücke bestimmt, wie Harry von einer Insel aus weitergeht. Damit Harry überhaupt alle Inseln betreten kann, muss diese Lücke also „kein Schild" enthalten. Hier ist ein Beispiel: Harry war schon auf den Inseln C, E, F – dort stehen Schilder. Von E aus geht er zu B, weil dies die einzige Nachbarinsel von E ist, auf der kein Schild steht.

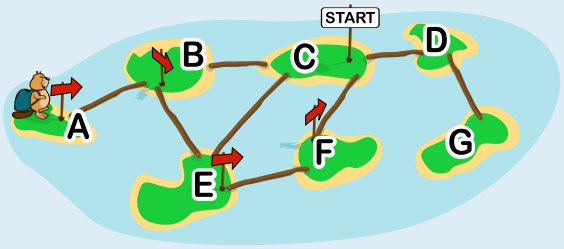
Die Textbausteine „ein Pfeil“ und „ein START-Schild“ sind beide falsch. Wenn Harry auf der ersten Insel entscheiden soll, wie er weitergeht, gibt es nur das START-Schild auf der aktuellen Insel und insbesondere noch keinen Pfeil. Beide Optionen würden also bewirken, dass Harry für immer auf der ersten Insel bleibt.



Zweite, grüne Lücke:
Der Text in der Lücke bestimmt, wie Harry sich auf einer Insel verhält, die keine Nachbarinsel ohne Schild hat; er hat dann alle Nachbarinseln bereits betreten. Auf seinem Weg zu dieser Insel hat Harry mit den Pfeilen eine Spur gelegt, und zwar zurück zu der Insel, an der er seine Erkundung gestartet hat. Der Text in der Lücke muss nun dafür sorgen, dass Harry diese Spur zurückverfolgt, bis er auf eine Insel kommt, die Nachbarinseln ohne Schild hat – oder bei der er begonnen hat. Das macht der Textbaustein „auf die der Pfeil zeigt“.

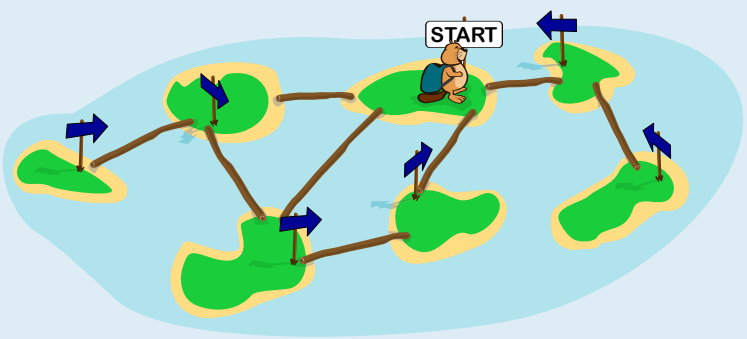
Der Textbaustein „von der du gerade gekommen bist“ ist falsch. Damit könnte Harry für immer zwischen zwei Inseln ohne unbetretene Nachbarinseln hin- und herpendeln.

Hier ist ein Beispiel: Harry befindet sich auf Insel A, die keine Nachbarinsel ohne Schild hat. Auf der einzigen Nachbarinsel, B, steht ein Schild. Und auch auf jeder Nachbarinsel von B steht je ein Schild. Würde Harry nun immer zu der Insel gehen, „von der er gerade gekommen ist“, würde er für immer zwischen A und B hin und her gehen. Wenn Harry hingegen immer zu der Insel geht, „auf die der Pfeil zeigt“, folgt er den Pfeilen zurück bis Insel C und betritt dann Insel D, weil darauf noch kein Schild steht.

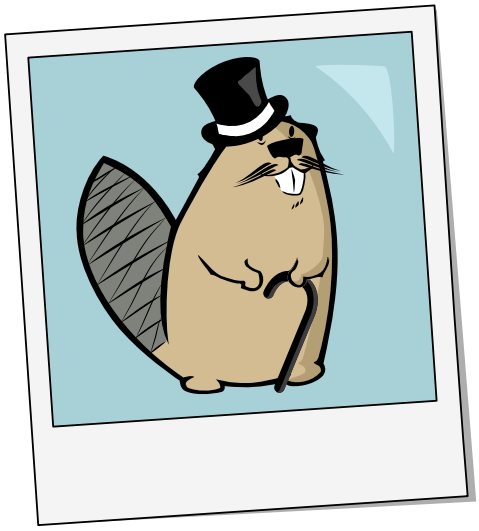


Das ist Informatik!
Das Netzwerk aus miteinander verbundenen Inseln kann durch einen *Graph* modelliert werden. Ein Graph ist eine Datenstruktur. Er besteht aus Knoten (hier sind das die Inseln) und Kanten (hier sind das die Baumstämme), die die Knoten verbinden. Viele Systeme der Wirklichkeit können durch Graphen beschrieben werden. Zum Beispiel sind Menschen durch Freundschaften verbunden, Bahnhöfe sind durch Schienenstrecken verbunden, und die möglichen Stellungen eines Schachspiels sind durch gültige Spielzüge verbunden. Manchmal ist es wichtig, einen Graphen vollständig zu durchlaufen, z. B. wenn man einen Knoten mit bestimmten Eigenschaften sucht. Vor allem bei großen Graphen mit vielen Knoten und Kanten braucht man eine Strategie, die sicherstellt, dass man wirklich jeden Knoten besucht und keine Knoten auslässt.

Die Strategie, die in dieser Biberaufgabe beschrieben ist, kennt man in der Informatik als *Tiefensuche*. Nach der vollständigen Anwendung des Algorithmus auf die Inselgruppe in der Aufgabe ist der Biber wieder auf der Insel, auf der er gestartet ist, und hat so seine Schilder aufgestellt:



Faktencheck



Als der berühmte Biber Alfred zu einem großen Fest kam, wurde er fotografiert.

Vier Influencer schrieben über diesen Moment, aber nicht alles in sozialen Medien ist wahr:

Nur eine der vier Aussagen ist wahr. Welche?

A) Alfred hatte einen Hut auf **und** benutzte keinen Stock .

B) Alfred trug eine Krawatte **oder** eine Fliege .

C) Alfred hatte einen Hut auf **und** trug eine Fliege .

D) Alfred hatte Flipflops an **oder** sogar keine Schuhe .

Antwort D ist richtig:

Um die richtige Antwort zu bestimmen, muss der Wahrheitsgehalt (wahr oder falsch) der vier Aussagen festgestellt werden, im Vergleich zum Foto. Jede Aussage besteht aus zwei Teilaussagen, die entweder mit **und** oder mit **oder** verknüpft sind. Der Wahrheitsgehalt der gesamten Aussage lässt sich aus dem Wahrheitsgehalt der Teilaussagen bestimmen:

und: Die gesamte Aussage ist *wahr,* wenn beide Teilaussagen *wahr* sind; sonst ist sie *falsch.*
oder: Die gesamte Aussage ist *wahr,* wenn mindestens eine der Teilaussagen *wahr* ist; sonst ist sie *falsch.*

In der deutschen Sprache sind manche der zweiten Teilaussagen verkürzt, damit Subjekt oder Prädikat nicht wiederholt werden. In der folgenden Analyse sind die Teilaussagen vollständig angeführt:

Aussage A:
Alfred hatte einen Hut  auf *(wahr)* **und** Alfred benutzte keinen Stock  .
⟶ Diese Aussage ist *falsch,* weil nicht beide Teilaussagen *wahr* sind.

Aussage B:
Alfred trug eine Krawatte  **oder** Alfred trug eine Fliege  .
⟶ Diese Aussage ist *falsch,* weil nicht mindestens eine Teilaussage *wahr* ist.

Aussage C:
Alfred hatte einen Hut  auf *(wahr)* **und** Alfred trug eine Fliege  .
⟶ Diese Aussage ist *falsch,* weil nicht beide Teilaussagen *wahr* sind.

Aussage D:
Alfred hatte Flipflops  an *(falsch)* **oder** Alfred hatte sogar keine Schuhe  an.
⟶ Diese Aussage ist *wahr,* weil (mindestens) eine Teilaussage *wahr* ist.

Das ist Informatik!
In den sogenannten „sozialen Medien" ist nicht immer alles wahr. Und leider gibt es nicht immer ein Foto, mit dem wir den Wahrheitsgehalt einer Aussage dort überprüfen können.

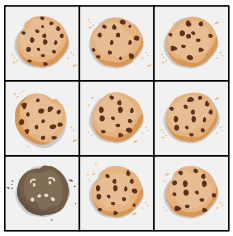
Auch die Logik beschäftigt sich mit der Wahrheit von Aussagen: Aussagen können *wahr* oder *falsch* sein. Logische Operatoren können verwendet werden, um (einfachere) Aussagen zu komplexeren Aussagen zu verknüpfen. Der Wahrheitsgehalt der gesamten Aussage lässt sich dann aus dem Wahrheitsgehalt der Teilaussagen berechnen. In dieser Biberaufgabe werden zwei Operatoren verwendet, **und** und **oder.**

Die *Wahrheitswerte* der gesamten Aussagen können für beide Operatoren in einer *Wahrheitstabelle* dargestellt werden:

Teilaussage A	Teilaussage B	A und B	A oder B
wahr	*wahr*	*wahr*	*wahr*
falsch	*wahr*	*falsch*	*wahr*
wahr	*falsch*	*falsch*	*wahr*
falsch	*falsch*	*falsch*	*falsch*

Der Umgang mit Logik ist fundamental in der Informatik, unter anderem beim Programmieren. Da verwendet man logische Ausdrücke, um festzulegen, unter welchen Bedingungen ein Programm was tun soll. In manchen Programmen wird das auch bei der Benutzung offensichtlich, etwa beim Aufsetzen eines Spam-Filters in einem E-Mail-Programm. Verwendet die Bedingung eines Spam-Filters ein **und** (zum Beispiel: die E-Mail stammt von einem unbekannten Absender **und** die E-Mail hat das Wort „Bitcoin" im Betreff), dann landet eine Mail nur dann im Spam-Ordner, wenn beide Teilaussagen wahr sind.

Frowny

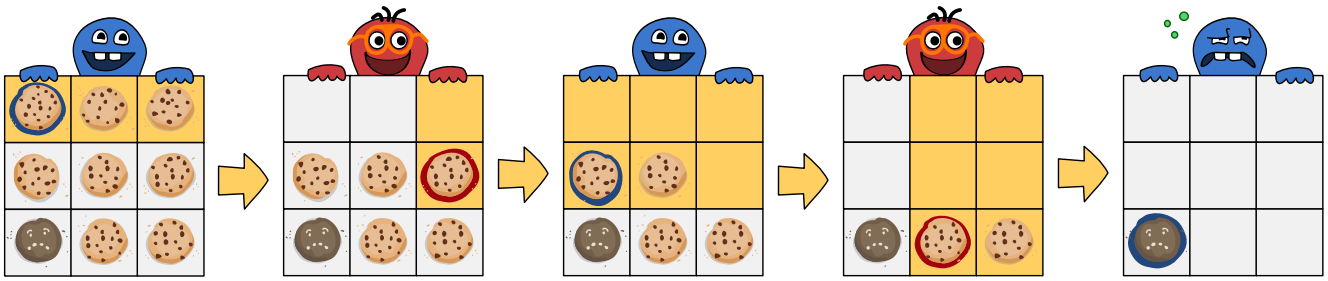


In einer Schachtel sind neun Kekse. Alle sind lecker, bis auf den Frowny  unten links.

Cleo  und Dan  nehmen abwechselnd Kekse aus der Schachtel, bis sie leer ist. Cleo darf immer anfangen.

Man darf die Kekse nur so aus der Schachtel nehmen:
- Man wählt zuerst einen beliebigen Keks aus.
- Dann muss man diesen Keks nehmen und zusätzlich alle Kekse aus dem rechteckigen Bereich **darüber und rechts** davon.

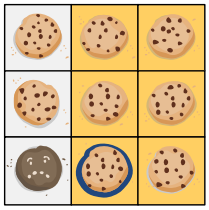
Hier ist ein Beispiel. Der ausgewählte Keks ist jeweils blau oder rot markiert. Dummerweise muss Cleo am Ende den Frowny nehmen, um die Schachtel zu leeren.



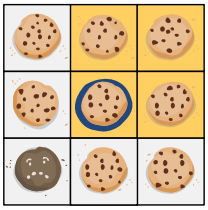
Cleos Ziel ist, dass Dan auf jeden Fall den Frowny nehmen **muss,** egal welche Kekse Dan zwischendurch nimmt. Sie überlegt, welche Kekse sie am Anfang nehmen soll, um ihr Ziel sicher zu erreichen. Auch wenn Cleo zwischendurch weitere Kekse nimmt, verfolgt sie ihr Ziel.

Hier sind vier Möglichkeiten, welche Kekse Cleo am Anfang nehmen kann. Mit einer davon kann sie ihr Ziel sicher erreichen. Mit welcher?

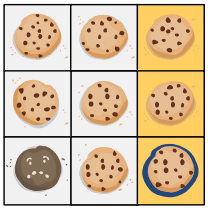
A)



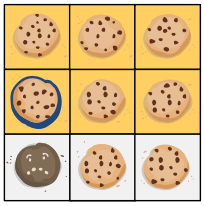
B)



C)



D)



Antwort B ist richtig:

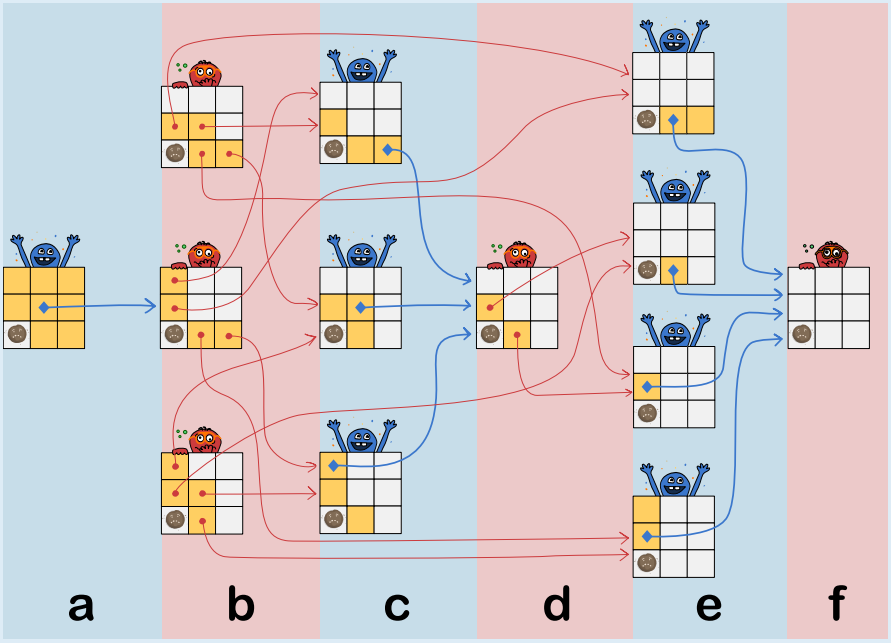
Wenn Cleo die Kekse am Anfang so nimmt, muss Dan auf jeden Fall den Frowny nehmen, und Cleo erreicht sicher ihr Ziel. Bei den anderen drei Möglichkeiten kann Dan einen Weg finden, das zu vermeiden. Zur Erklärung betrachten wir das Kekse-Nehmen wie ein Spiel:
- Wenn Cleo oder Dan Kekse nehmen, machen sie einen Spielzug. Jeder Spielzug ändert die Spielsituation (nämlich welche Kekse wo sind).

- Wenn man am Zug ist und nur noch verlieren kann, ist man in einer Verlustsituation. Das ist auf jeden Fall die Situation, in der nur noch der Frowny in der Schachtel liegt. Das ist aber auch jede andere Situation, in der man den Gegner mit allen möglichen Zügen in eine …
- … *Gewinnsituation* bringt, in der der Gegner mit einem Zug wieder eine *Verlustsituation* erzeugen kann.

Cleo braucht also in der Startsituation einen Zug, mit dem sie Dan in eine Verlustsituation bringen kann.

Im Bild sind einige Situationen dargestellt und in Bereiche eingeteilt:
- Bereich f zeigt die Verlustsituation, in der nur noch der Frowny in der Schachtel liegt.
- Bereich e zeigt vier Gewinnsituationen, aus denen jeweils ein Zug zur Verlustsituation in f führt.
- Bereich d zeigt eine Verlustsituation, denn beide möglichen Züge führen zu einer Gewinnsituation in e.
- Bereich c zeigt drei Gewinnsituationen: Für jede Situation gibt es einen Zug, der zur Verlustsituation in d führt.
- Bereich b enthält wieder Verlustsituationen: In jeder Situation führen alle möglichen Züge zu den Gewinnsituationen in c und e.

Man kann sich überlegen, dass man von allen weiteren, im Bild nicht gezeigten möglichen Situationen aus mit einem Zug zu einer Verlustsituation kommt. Das Bild zeigt davon in Bereich a nur die Startsituation des Spiels, aus der genau der Zug von Antwort B (und kein anderer) zu einer Verlustsituation führt. Nur mit diesem Zug kann Cleo also ihr Ziel sicher erreichen.



Das ist Informatik!
Dieses Spiel kann als eine Variante des bekannten Strategiespiels Nim angesehen werden. Im Spiel werden Konzepte aus der Spieltheorie gebraucht. Ein *Spielbaum*, der alle erdenkliche Züge beschreibt, wäre viel zu groß, um dargestellt zu werden. Nach manchen Spielsituationen verzweigen sich die Wege, nach anderen nicht. Längere Ketten von Spielsituationen ohne Verzweigungen können kompakter dargestellt werden (wie mit dem Pfeil in der obigen Zeichnung). Jedes mal, wenn wir neu anfangen zu spielen und über andere Wege eine vorbemerkte Spielsituation erreichen, können wir direkt nach vorne (d.h. über den Pfeil) springen, ohne weitere Berechnungen vorzunehmen.

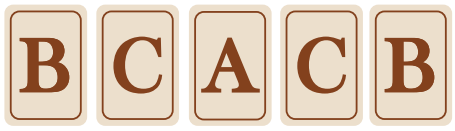
Die Idee, Spielsituationen vorzumerken und diese geschickt wiederzuverwenden, ist eine allgemeine Problemlösungsstrategie, die in der Informatik oft verwendet wird und den Namen *Dynamische Programmierung* trägt.

Mithilfe eines Computers können wir ein Programm schreiben, das diese Strategie umsetzt und das Spiel für ein beliebig großes Spielfeld löst. Das Programm entscheidet in jedem Schritt, welchen Zug eine Spielerin (zum Beispiel Cleo) vornehmen muss, um eine Gewinnsituation zu erzeugen.

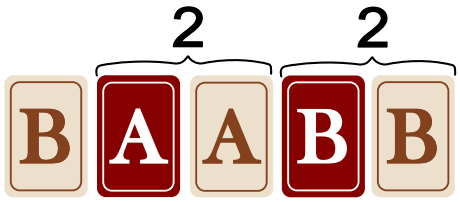
Gleiche Buchstaben

Lege Buchstaben in eine Reihe und bekomme Punkte: Wenn 2 gleiche Buchstaben direkt nebeneinander liegen, bekommst du 2 Punkte, bei 3 gleichen Buchstaben 3, und so weiter.

Ein Beispiel:



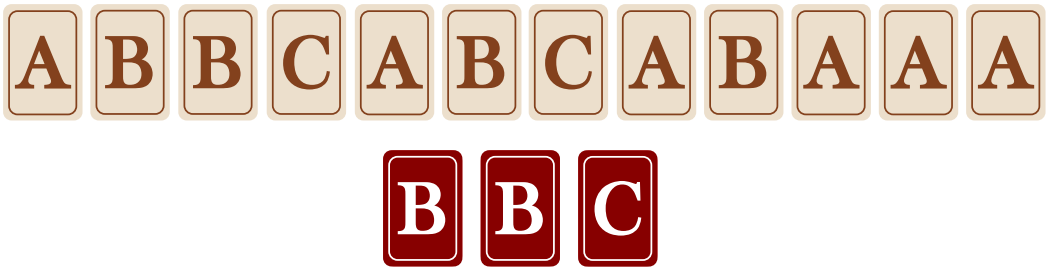
Für diese Reihe bekommst du 0 Punkte, denn es liegen nirgendwo gleiche Buchstaben direkt nebeneinander.



Nun ersetzt du das erste C durch ein A und das zweite C durch ein B. Für das Ergebnis bekommst du 4 Punkte: 2 für die beiden A's und 2 für die beiden B's hintereinander.

Jetzt hast du eine neue Reihe gelegt, mit 12 Buchstaben. Du kannst darin drei beliebige Buchstaben ersetzen, und zwar durch die Buchstaben B, B und C.

Ersetze so, dass du für das Ergebnis so viele Punkte bekommst wie möglich.



So ist es richtig:

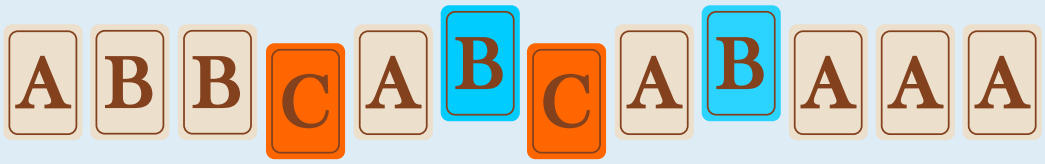
Für diese Reihe bekommst du 2 + 2 + 4 + 3 = 11 Punkte; von links nach rechts liegen 2 B's, 2 C's, 4 B's und 3 A's in Gruppen hintereinander. Mehr Punkte kannst du nicht bekommen:

- Die Reihe hat 12 Buchstaben. Du könntest also höchstens 12 Punkte bekommen.
- Das A ganz links kann jedoch nicht zu einer Gruppe gehören, denn du hast kein A, um das B rechts daneben zu ersetzen.
- Alle anderen 11 Buchstaben gehören zu einer Gruppe und tragen jeweils 1 Punkt zur Gesamtpunktzahl bei.

Gibt es vielleicht noch andere Ersetzungen, durch die man 11 Punkte bekommen kann? Um möglichst viele Punkte zu bekommen, muss man die Buchstaben B, B und C so in der Reihe platzieren, dass neue Gruppen entstehen und keine bestehende Gruppe aufgelöst oder verkleinert wird:

- Wenn ein Buchstabe passend an eine bestehende Gruppe gelegt wird, dann kommt 1 Punkt hinzu.
- Wenn ein Buchstabe an einen gleichen Einzelbuchstaben gelegt wird, so dass eine neue Gruppe entsteht oder der Einzelbuchstabe an eine bestehende Gruppe angebunden wird, dann kommen 2 Punkte hinzu.
- Wenn ein Buchstabe zwischen zwei gleiche Einzelbuchstaben gelegt wird, dann entsteht eine neue Dreiergruppe, und es kommen sogar 3 Punkte hinzu.

Für die neue Reihe bekommt man ursprünglich 5 Punkte. Mit einer einzelnen Ersetzung durch B, B oder C auf einen Schlag 3 Punkte hinzuzugewinnen, ist nicht möglich. Um auf 11 Punkte zu kommen, müssen also B, B und C so gelegt werden, dass jedes Mal eine neue Zweiergruppe entsteht. Dafür stehen nur zwei Einzel-B's und zwei Einzel-C's zur Verfügung:



Es bleibt nur eine Möglichkeit, mit B, B und C drei neue Zweiergruppen zu bilden, ohne die bestehenden Gruppen zu verkleinern:

Das ist Informatik!
Das Buchstaben-Ersetzungs-Problem in dieser Biberaufgabe ist ein *Optimierungsproblem.* Es ist nicht irgendeine Ersetzung gesucht, sondern die, mit der sich die höchste Punktwertung erzielen lässt. Diese Ersetzung ist dann eine optimale Lösung des Problems.

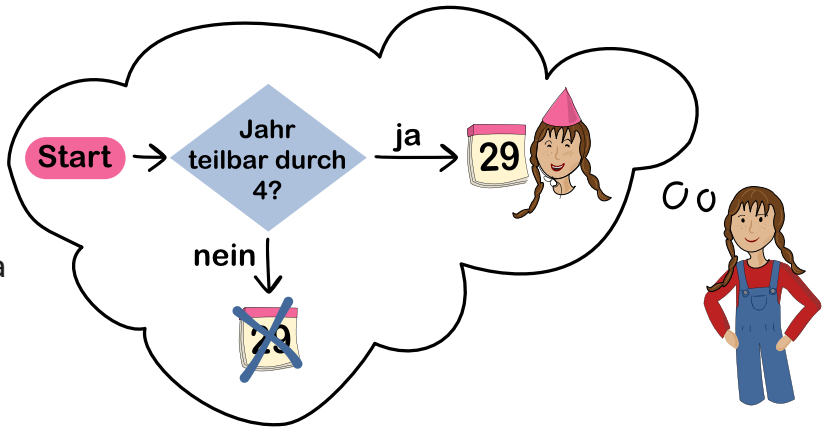
Die optimale Lösung muss im *Lösungsraum,* also in der Menge aller denkbaren „Lösungen" gesucht werden. In dieser Aufgabe ergibt sich der Lösungsraum aus allen Möglichkeiten, drei Buchstaben der Reihe durch B, B und C zu ersetzen. Da die Reihe zwölf Buchstaben hat, gibt es $12 \times 11 \times 10 = 1320$ Ersetzungsmöglichkeiten, für die jeweils die Punktzahl berechnet und mit den Punktzahlen der anderen Möglichkeiten verglichen werden muss, um die beste Lösung zu bestimmen.

Bei automatischer Verarbeitung durch ein Computerprogramm ist ein Lösungsraum mit 1320 Elementen recht klein. In der Informatik ist es aber immer wichtig zu fragen, wie sich der Lösungsaufwand bei wachsender Problemgröße verhält. Bei längeren Reihen und mehr Ersetzungsbuchstaben ist der Lösungsraum entsprechend größer; bei zum Beispiel einer Reihe mit 20 Buchstaben und 5 Ersetzungsbuchstaben besteht der Lösungsraum schon aus annähernd 2 Millionen Elementen.

Deshalb werden Optimierungsprobleme in der Informatik immer sorgfältig analysiert, um den Lösungsraum möglichst einzuschränken. In dieser Aufgabe kann der Lösungsraum deutlich eingeschränkt werden, wenn man berücksichtigt, dass B, B und C nur dann Punkte erzielen, wenn sie neben den gleichen Buchstaben liegen. Bezieht man zusätzlich ein, dass bestehende Gruppen nicht verkleinert werden sollten, wird der Lösungsraum weiter eingeschränkt. Wenn man auch noch beachtet, dass die Anbindung von Einzelbuchstaben mehr Punkte bringt als die Vergrößerung bestehender Gruppen, dann bleibt nur eine optimale Platzierung übrig.

Happy Birthday

Johanna wurde in einem Schaltjahr an einem 29. Februar geboren. Nur wenn wieder ein Schaltjahr ist, kann sie ihren Geburtstag am 29. Februar feiern. Bisher war das alle vier Jahre. Um schnell bestimmen zu können, ob ein Jahr ein Schaltjahr ist, hat Johanna diesen „Entscheidungsplan“ gemacht:



Nach einiger Zeit lernt Johanna, dass es doch etwas komplizierter ist, ein Schaltjahr zu bestimmen:

- Wenn das Jahr durch 100 teilbar ist, ist es kein Schaltjahr (zum Beispiel 1900).
- Ist das Jahr aber durch 400 teilbar, dann ist es doch ein Schaltjahr (zum Beispiel 2000).

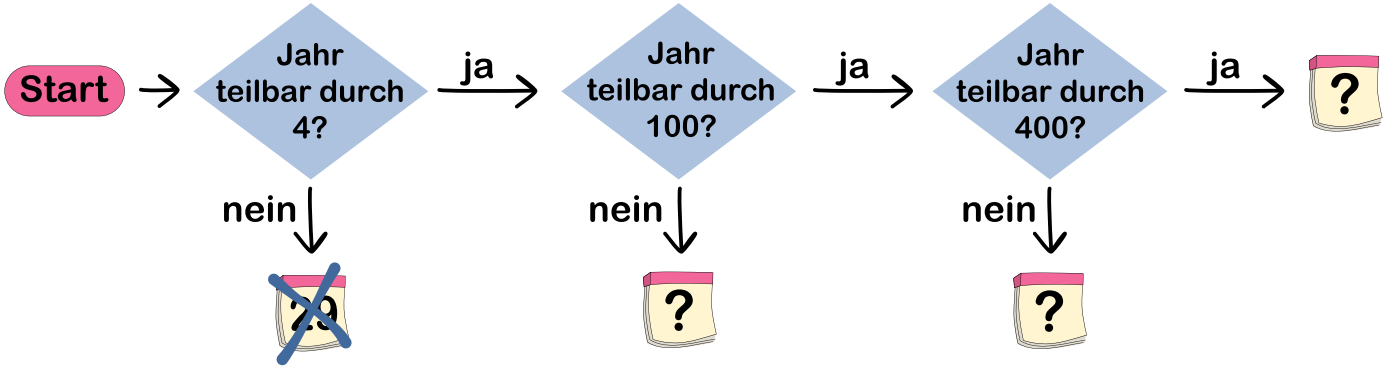
Johanna erweitert ihren Plan und fügt zwei Fragen hinzu.

Nur die Entscheidungen sind noch offen: Schaltjahr (

)oder nicht ().

**Hilf Johanna und wähle für jedes Fragezeichen 

die richtige Entscheidung aus.**



So ist es richtig:

Johannas Entscheidungsplan ist eine andere Darstellung der in der Aufgabe formulierten Regeln für Schaltjahre. Die Fragen, die Johanna ihrem Plan hinzugefügt hat, stellen die neuen Regeln dar, die Johanna später gelernt hat. Wichtig ist, dass die Regel für Jahre, die durch 400 teilbar sind, die Regel für Jahre, die durch 100 teilbar sind, aufhebt. Deshalb hat Johanna die Frage zur „400er-Regel“ auch richtig hinter die Frage zur „100er-Regel“ gesetzt. Beide Fragen werden nur gestellt für Jahre, die durch 4 teilbar sind.

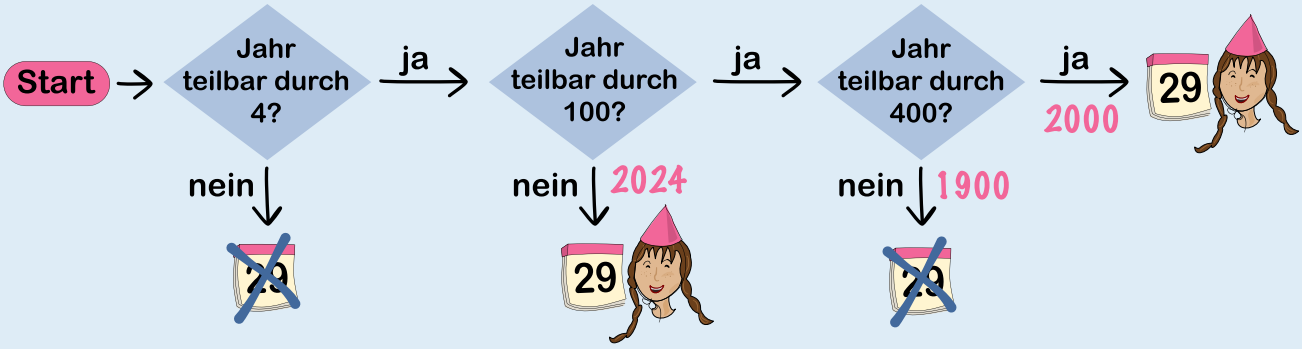
- Die Entscheidung unten Mitte wird also für Jahre getroffen, die durch 4 teilbar sind, aber nicht durch 100; solche Jahre sind dann auch nicht durch 400 teilbar. Die beiden neuen Regeln treffen hier also nicht zu, und es gilt weiter die Regel für Jahre, die durch 4 teilbar sind: Sie sind ein Schaltjahr.

- Die Entscheidung unten rechts wird für Jahre getroffen, die durch 4, durch 100 aber nicht durch 400 teilbar sind. Von den neuen Regeln trifft also nur die 100er-Regel zu: kein Schaltjahr.

- Die Entscheidung ganz rechts wird für Jahre getroffen, die durch 4, 100 und 400 teilbar sind. Hier trifft die 400er-Regel zu: Schaltjahr.

Wir testen den fertigen Plan mit den schon vorher erwähnten Jahren 1900, 2000 und 2024:

1. Das Jahr 2000 ist ein Schaltjahr. 2000 ist durch 4, 100 und 400 teilbar. Im Entscheidungsplan landen wir also bei der Entscheidung ganz rechts: Schaltjahr. Passt!
2. Das Jahr 1900 ist kein Schaltjahr. Weil 1900 durch 4 und durch 100 aber nicht durch 400 teilbar ist, führt der Entscheidungsplan für dieses Jahr zur Entscheidung unten rechts: kein Schaltjahr. Passt auch!
3. Das Jahr 2024 ist ein Schaltjahr. 2024 ist durch 4, aber nicht durch 100 (und damit auch nicht durch 400) teilbar. Der Entscheidungsplan führt für dieses Jahr zur Entscheidung unten mitte: Schaltjahr. Richtig!



Das ist Informatik!
Mit dem Entscheidungsplan aus dieser Biberaufgabe kann Johanna für jedes Jahr (genauer: für jede Jahreszahl) bestimmen, ob das Jahr ein Schaltjahr ist oder nicht. Der Plan selbst hat einen endlichen Umfang (siehe oben), die Entscheidungsfragen sind eindeutig formuliert und können für jede Jahreszahl beantwortet werden, Johanna kann den Plan für jede Jahreszahl in endlich vielen Schritten bis zu einer Entscheidung durchlaufen, und sie muss sich dafür nur die Zahl selbst merken. Damit hat der Plan alle Eigenschaften eines Algorithmus. Er beschreibt einen Algorithmus, der für eine Jahreszahl entscheidet, ob das Jahr ein Schaltjahr ist oder nicht.

Johanna hat ihren Plan bzw. die Beschreibung ihres Schaltjahr-Entscheidungs-Algorithmus wie einen (einfachen) *Programmablaufplan* gestaltet. Mittels Programmablaufplänen können Algorithmen beschrieben werden, ohne auf eine bestimmte Programmiersprache zurückgreifen zu müssen. In der Informatik ist es üblich, einen Algorithmus mit einem Programmablaufplan oder auch einem anderen, Programmiersprachen-neutralen Format wie Struktogrammen oder Pseudocode zu beschreiben. Eine solche Beschreibung untermauert, dass das beschriebene Verfahren nicht von technischen Details einer bestimmten Programmiersprache abhängt. Der so beschriebene Algorithmus kann aus dem neutralen Format dann in eine Programmiersprache übersetzt werden.

Im Park

Alia  ist bei der Biberstatue  im Park.

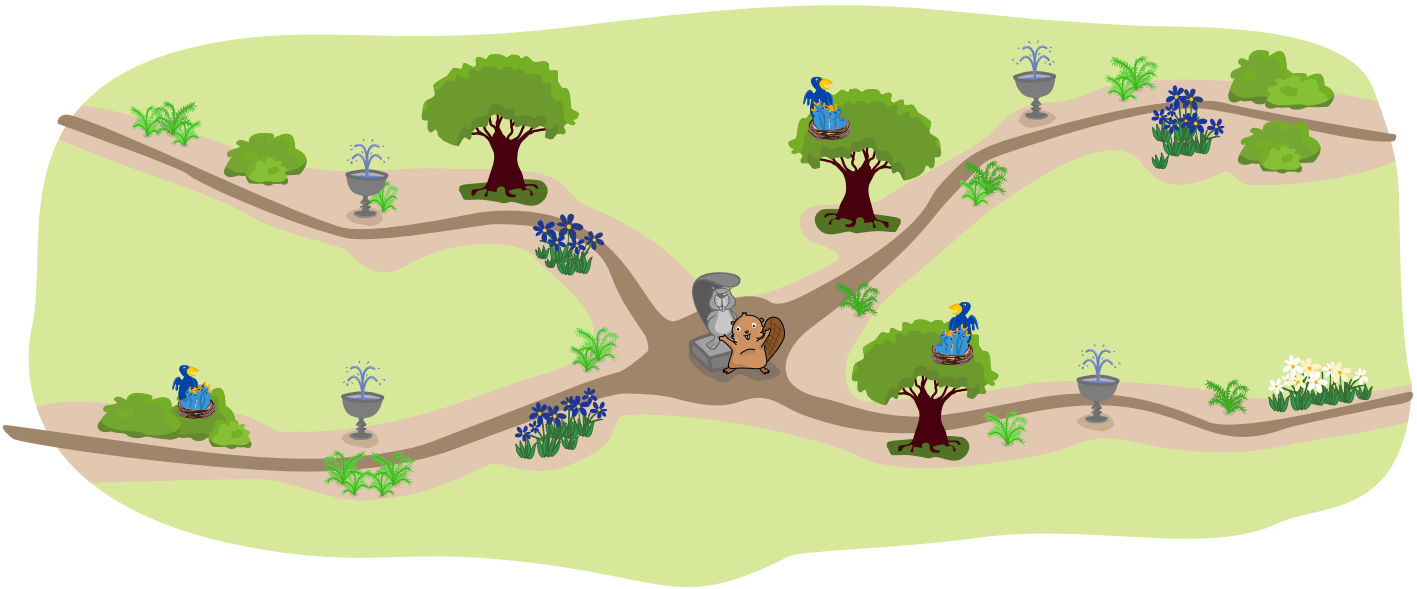
Auf ihrem Weg dorthin hat sie einige Dinge gesehen:

Zuerst hat sie blaue Blumen gesehen.

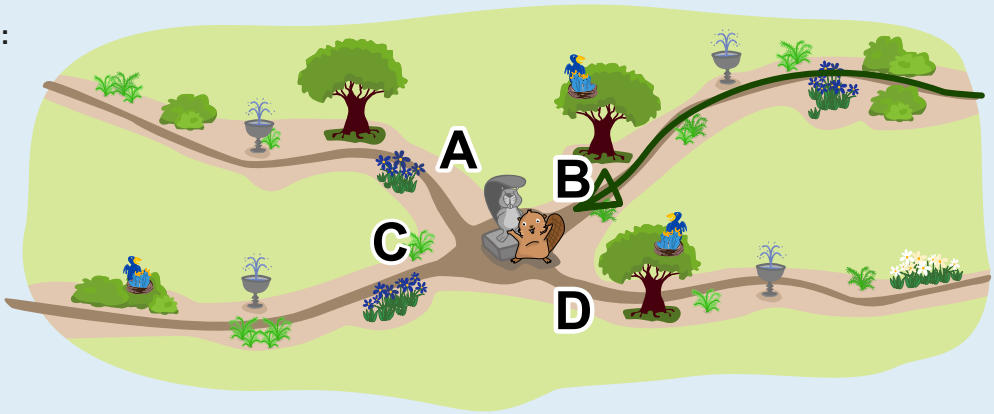
Danach hat sie einen Brunnen gesehen.

Zuletzt hat sie ein Nest entdeckt.

Auf welchem Weg ist Alia zur Biberstatue gegangen?



So ist es richtig:

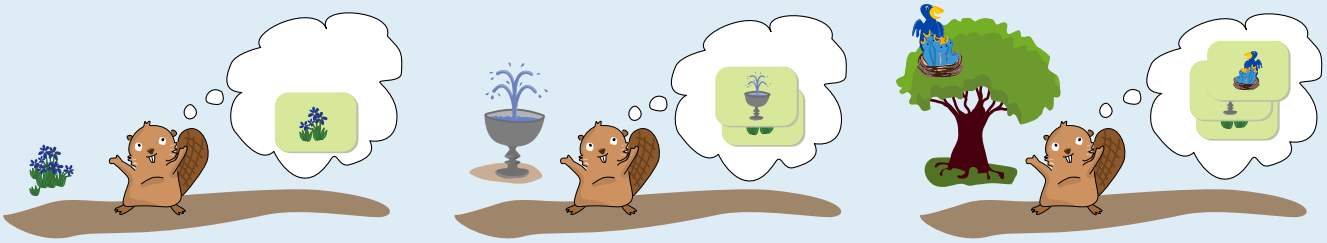


Um die richtige Antwort zu finden, prüft man zunächst, auf welchem Weg alle Dinge vorkommen, die Alia gesehen hat. Anschließend prüft man ausgehend von der Biberstatue, auf welchem Weg die Dinge, die Alia gesehen hat, in der umgekehrten Reihenfolge vorkommen.
Auf ihrem Weg zur Biberstatue hat Alia zuerst die blauen Blumen, dann den Brunnen und zuletzt das Nest gesehen. Auf dem korrekte Weg müssen die gesehenen Dinge also ausgehend von der Statue in der folgenden Reihenfolge vorkommen: Zuerst das Nest, dann der Brunnen und zuletzt die blauen Blumen.

Weg A ist nicht richtig, weil kein Nest vorkommt.
Weg C ist nicht richtig, weil die Dinge in umgekehrter Reihenfolge vorkommen.
Weg D ist nicht richtig, weil die Blumen am Ende nicht blau, sondern weiß sind.

Der einzige Weg, auf dem alle Dinge von der Statue aus in dieser Reihenfolge vorkommen, ist Weg B.

Das ist Informatik!
Auf ihrem Weg in den Park sieht Alia verschiedene Dinge in einer bestimmten Reihenfolge: zuerst die Blumen, dann den Brunnen und zuletzt das Vogelnest:



Wenn Alia sich auf dem Hinweg diese Informationen in genau dieser Reihenfolge merken will, um den Rückweg leichter zu finden, kann sie sie wie auf einem *Stapel* ablegen – zuletzt das Vogelnest. Auf dem richtigen Rückweg wird sie zuerst wieder das Vogelnest sehen. Das löscht sie dann aus ihrem „Stapel-Merkspeicher“ und geht weiter. Kurz gesagt: Was auf dem Hinweg zuletzt auf ihren Stapel gekommen ist und nun dort zuoberst liegt, nimmt sie auf dem Rückweg zuerst heraus. Das könnte man auf Englisch noch kürzer so ausdrücken: „last in, first out“ (zuletzt hinein, zuerst heraus), kurz: *LIFO.*

Die Informatik kennt eine Struktur zur Speicherung von Daten, die nach diesem LIFO-Prin
zip funktioniert. Diese Struktur heißt *Stack,* auf Deutsch: Stapel. Natürlich kann man sie nur dann sinnvoll verwenden, wenn man auf die gespeicherten Daten nach dem LIFO-Prinzip zugreifen will. Das scheint eine starke Einschränkung zu sein. Aber weil Stacks in Computern sehr einfach zu realisieren sind, werden sie dennoch recht häufig verwendet. Bei einem Computerspiel zum Beispiel speichert der Computer deine letzten Spielzüge in einem Stapel, so dass du bei Bedarf Zug um Zug rückgängig machen kannst.

Karten drehen

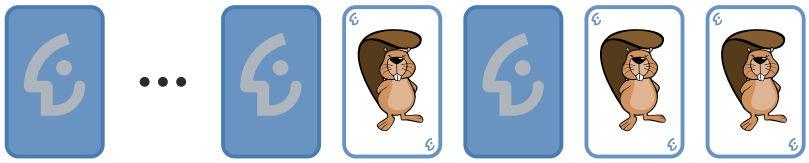
Jemand schenkt dir einen Satz gleicher Karten. Die Karten sehen so aus:

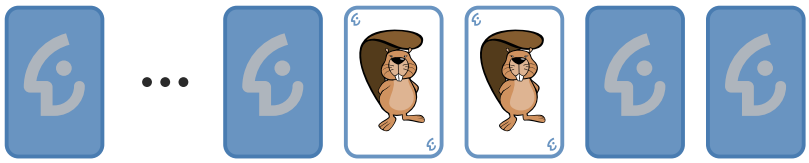
Aufgedeckt:  Verdeckt: 

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 Beispiel zeigt, wie sich die Karten in einem Spielzug verändern können:

Vorher: 

Nachher: 

- Die beiden rechten Karten sind aufgedeckt und werden umgedreht.
- Die dritte Karte von rechts ist verdeckt und wird aufgedeckt.

Damit ist der Spielzug beendet, die übrigen Karten bleiben unverändert.

Diesmal beginnt das Spiel mit 16 verdeckten Karten.

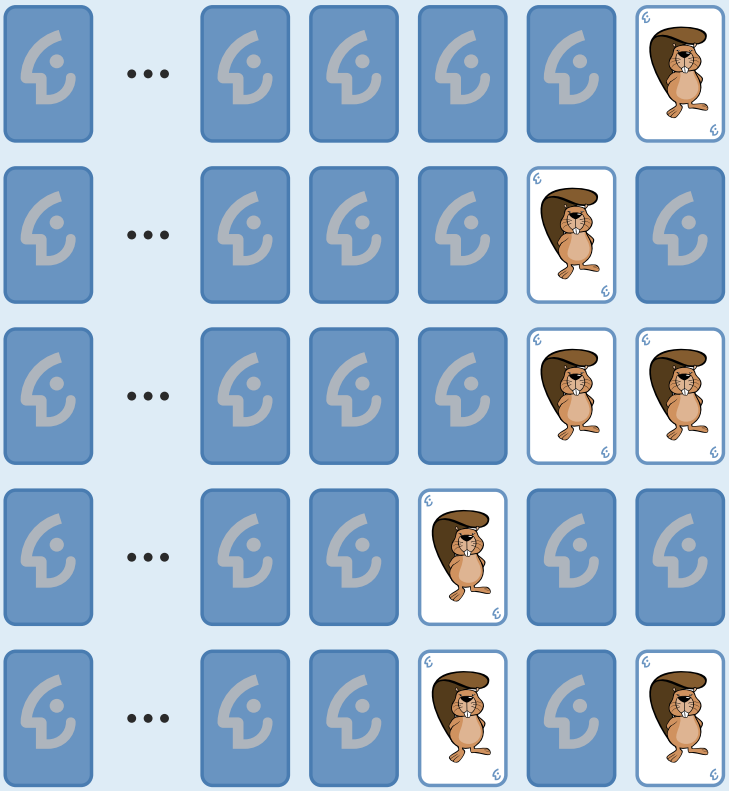
Wie viele Karten sind nach 16 Spielzügen aufgedeckt?

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 fünf Spielzüge, die also von 1 bis 5 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.

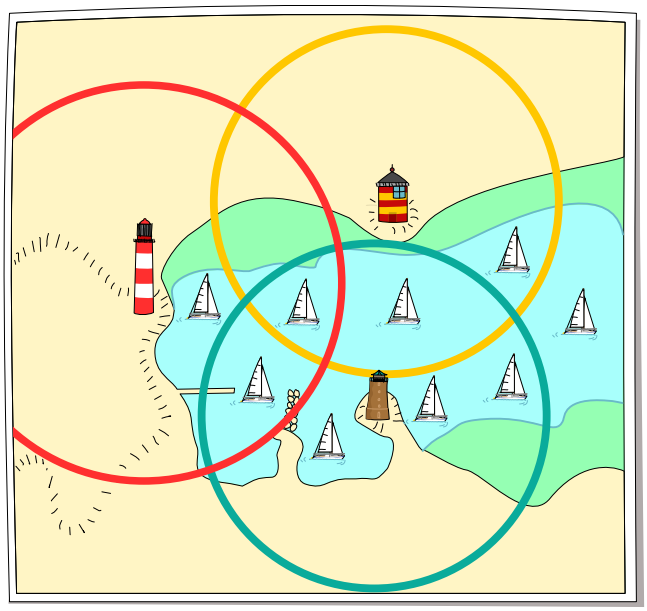
Seitdem es Computer gibt, werden darin solche Operationen möglichst effizient realisiert. Es gibt Operationen, die zwei Binärzahlen miteinander verknüpfen, wie etwa die Rechenoperationen Addition oder Multiplikation. Es gibt aber auch Operationen, die eine einzelne Binärzahl verändern, etwa das Verschieben 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.

Leuchttürme

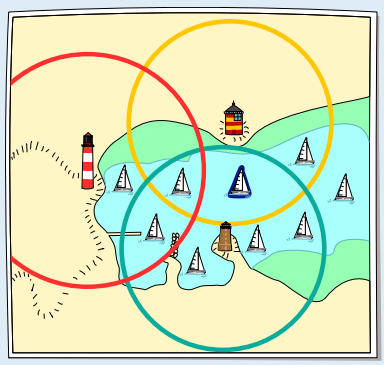
Ben ist auf seinem Segelboot.
Ben hat eine Karte. Sie zeigt die Leuchttürme.
Um jeden Leuchtturm ist ein Kreis gezeichnet.
Wenn Bens Boot in dem Kreis ist, kann Ben den Leuchtturm sehen.

Ben sieht die Leuchttürme  und  . Den Leuchtturm  sieht er nicht.

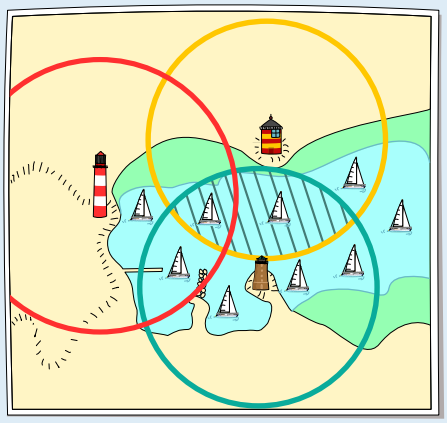
Welches ist Bens Boot?



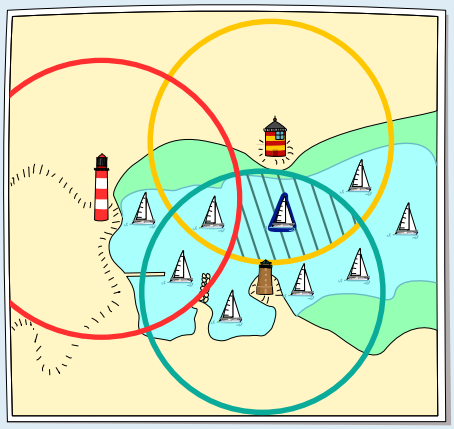
So ist es richtig:



Ben sieht die Leuchttürme  und  . Bens Boot ist also in dem Gebiet, in dem sich die Kreise dieser Leuchttürme überschneiden.



Ben sieht den Leuchtturm  nicht. Also ist Bens Boot nicht im Kreis dieses Leuchtturms.



Dies ist das einzige Boot, das in den beiden Kreise um die Leuchttürme  und  ist, aber nicht im Kreis um  .
Also ist dieses Boot Bens Boot.

Das ist Informatik!
In der Informatik ist die exakte Beschreibung eines Problems und des Lösungsverfahrens von großer Bedeutung. Ein nützliches Werkzeug dafür ist die Aussagenlogik, bei der mit Ausssagen gearbeitet wird, die jeweils wahr oder falsch sein können. In dieser Biberaufgabe kann ein Gebiet, in dem ein Boot ist, mit diesen drei Aussagen beschrieben werden:

- Aussage A lautet: „Leuchtturm  ist sichtbar.“
- Aussage B lautet: „Leuchtturm  ist sichtbar.“
- Aussage C lautet: „Leuchtturm  ist sichtbar.“

Für Bens Boot sind die Aussagen B und C wahr, aber die Aussage A ist falsch:

Ben sieht die Leuchttürme  und  , aber nicht den Leuchtturm  . In der Aussagenlogik wird das so aufgeschrieben: „B UND C UND NICHT A“.

Diese Aussage kombiniert die „einfachen“ Aussagen A, B und C mit Hilfe von UND und NICHT. Sie beschreibt das in der Erklärung gezeigte Gebiet, in dem genau Bens Boot ist.

In der Karte entspricht zum Beispiel „A UND B UND C“ dem Gebiet, in dem sich alle drei Kreise überschneiden, denn nur dort kann man alle drei Leuchttürme sehen. „NICHT A UND NICHT B UND NICHT C“ entspricht dem Gebiet, das außerhalb der drei Kreise liegt. Dort ist keiner der drei Leuchttürme sichtbar.

In der Informatik kommt die Aussagenlogik unter anderem bei der Wissensrepräsentation, bei der Verifizierung von Programmen und vor allem bei der Modellierung digitaler Schaltungen zur Anwendung.

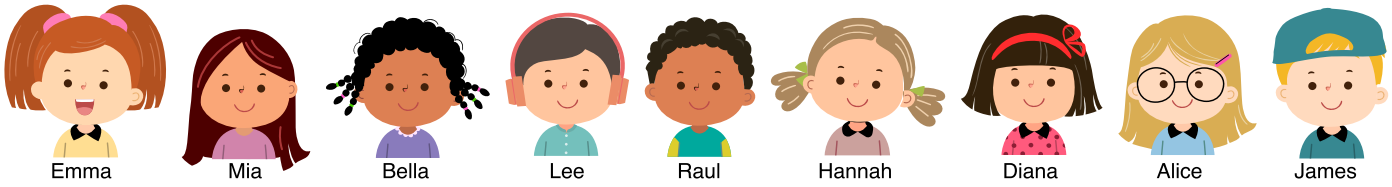
Nebeneinander

Ava hat die Schule gewechselt. Heute trifft sie ihre Freunde von der alten Schule online. Ihre Freunde sitzen alle nebeneinander in einer Reihe.

Jedes Kind sitzt vor einer eigenen Kamera. In jedem Kamerabild sind aber auch die Kinder zu sehen, die direkt daneben sitzen.

So sieht Ava ihre Freunde auf ihrem Bildschirm:

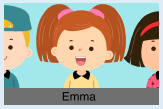
Wie sitzen die Freunde nebeneinander in der Reihe?



So ist es richtig:

Die Kamerabilder verraten, wer neben wem sitzt.

Im Kamerabild rechts unten auf dem Bildschirm ist links neben James kein Kind zu sehen. Deshalb sitzt James an einem Ende der Reihe. Das Kind, das rechts neben James zu erkennen ist ...

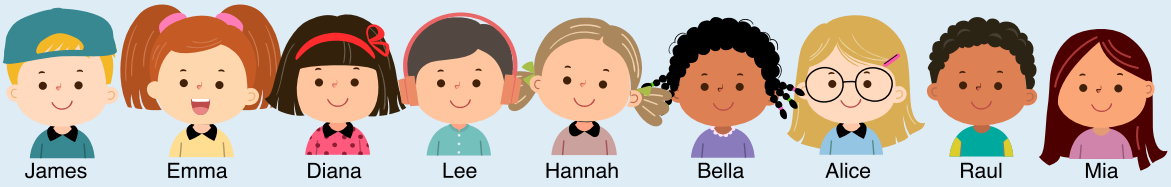


... ist im Bild links oben auf dem Bildschirm zu sehen: Emma. Das Kind, das rechts neben Emma zu erkennen ist, …

… ist im Bild links unten zu sehen: Diana. Das Kind, das rechts …

… ist im Bild oben mitte zu sehen: Mia. Rechts neben Mia ist kein Kind zu sehen. Deshalb sitzt Mia am anderen Ende der Reihe.

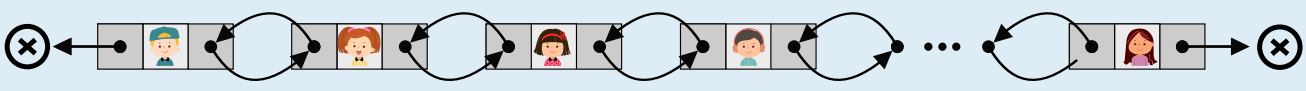
Insgesamt zeigen die Kamerabilder, dass die neun Freunde so nebeneinander sitzen:



Das ist Informatik!
Auf den Kamerabildern sieht man nicht nur ein Kind, sondern kann auch erkennen, welche Kinder links und rechts daneben sind. Informatikerinnen und Informatiker würden sagen: Die Bilder enthalten *Verweise* auf die Kinder daneben, und die Bilder sind (überwiegend) in zwei Richtungen miteinander *verkettet.*

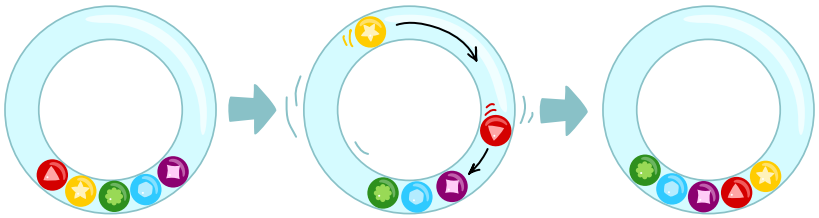
Die Kamerabilder in dieser Biberaufgabe bilden aus Informatik-Sicht eine *doppelt verkettete* Liste. Eine Liste ist allgemein eine Datenstruktur mit Knoten und Verweisen zwischen den Knoten; die Knoten enthalten Daten bzw. *Datenelemente.* In einer doppelt verketteten Liste enthält jeder Knoten ein Element, einen Verweis auf den vorherigen Knoten und einen Verweis auf den nächsten Knoten. In dieser Aufgabe zeigt jedes Kamerabild (Knoten) das Kind (Element) und Verweise auf die Kamerabilder und damit auch die Elemente rechts und links vom Kind.

Um die richtige Antwort zu finden, durchläufst du die Kinder-Liste und folgst den Verweisen, um die Kinder in die richtige Reihenfolge zu bringen. Im Gegensatz zu der zirkulär verketteten Liste in der Biberaufgabe „Olivers Rassel“ kannst du in beide Richtungen laufen. Es gibt jedoch einen Anfang *(Kopf)* und ein Ende der Liste. In unserer Liste ist James am Kopf und Mia am Ende der Liste. Ihre Verweise nach links bzw. nach rechts sind leer.



Ein Beispiel für eine doppelt verkettete Liste ist dein Browser-Verlauf: Du kannst beliebig rückwärts oder dann wieder vorwärts springen. Die aktuell betrachtete Seite (von der aus du im Verlauf zurückgehst), ist am Kopf der Liste.

Olivers Rassel



Oliver hat eine durchsichtige Rassel mit bunten Kugeln.
Wenn er die Rassel schüttelt, bewegen sich einige Kugeln.
Danach liegen die Kugeln anders in der Rassel als vorher.
Oliver schüttelt die Rassel noch einmal.

Wie liegen die Kugeln nun? Fülle die leeren Plätze in der Rassel.

So ist es richtig:

Nach dem Schütteln können die Kugeln anders liegen. Aber ihre Reihenfolge ändert sich nicht: Wenn man von einer bestimmten Kugel aus zur nächsten Kugel geht, ist das immer dieselbe – auch wenn man einmal rundherum durch die Rassel gehen muss.

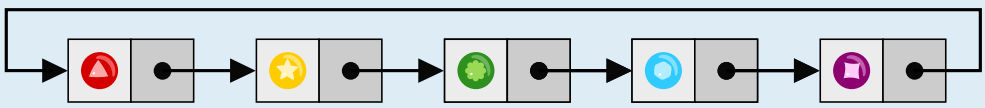
Schauen wir uns die Reihenfolge im ersten Bild oben an: Nach der roten Kugel kommt die gelbe. Wenn wir das mit einem Pfeil schreiben – → – können wir die gesamte Reihenfolge so aufschreiben: → → → → →

Nach der violetten kommt also wieder die rote.
So schliesst sich der Kreis in Olivers Rassel.

Mit Hilfe der Reihenfolge finden wir leicht die richtige Antwort:
Nach der violetten Kugel kommt die rote und danach die gelbe.
Die grüne Kugel liegt schon da, und danach kommt die blaue :

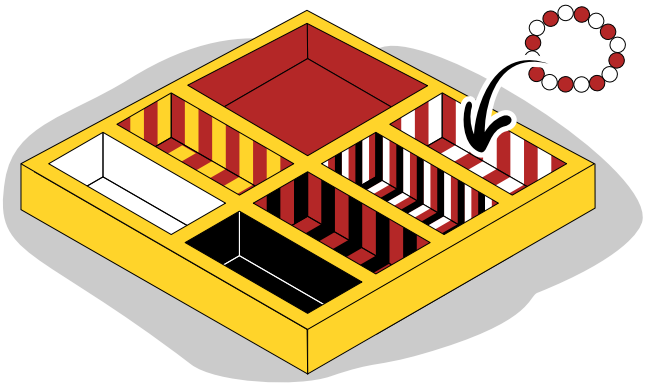
Das ist Informatik!
In Olivers Rassel bleibt die Reihenfolge der Kugeln immer gleich. Egal wo eine Kugel liegt, die nächste Kugel (in einer bestimmten Richtung) ist immer dieselbe. Aber die Kugeln in der Rassel haben noch eine Eigenschaft: Wenn man bei einer Kugel anfängt und zur nächsten Kugel geht und von dort wieder zur nächsten und so weiter, dann kommt man irgendwann wieder bei der Anfangs-Kugel an. Das ist so ähnlich wie bei Perlen an einer Kette, die jemand am Arm oder um den Hals trägt.

Auch in der Informatik kennt man Ketten. Wenn man Datenelemente in einer festen Reihenfolge bearbeiten will, können sie in einer *verketteten Liste* gespeichert werden. In einer verketteten Liste wird jedes Datenelement in einem eigenen *Knoten* gespeichert. Zusätzlich enthält jeder Knoten einen Verweis auf den nächsten Knoten in der Liste. Enthält der letzte Knoten einen Verweis auf den ersten Knoten, ist die Kette *zirkulär.* Die Kugeln in Olivers Rassel bilden so eine zirkuläre verkettete Liste, mit den Kugeln als Knoten und der Nachbarschaft innerhalb der Rassel (gegen den Uhrzeigersinn gesehen) als Verweise:



Verkettete Listen in Computerprogrammen kommen im Alltag selten zum Vorschein – aber gelegentlich doch: Wenn die Musik-App auf dem Smartphone Songs einer Wiedergabeliste der Reihe nach spielt und am Ende wieder von vorne anfängt, bilden auch die Songs eine zirkuläre verkettete Liste.

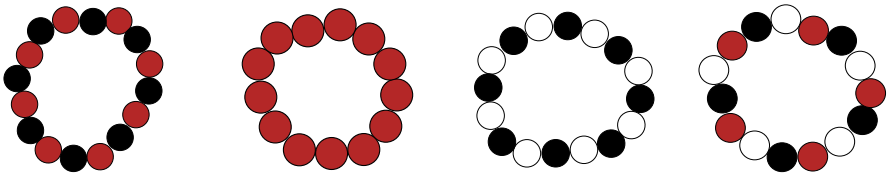
Passendes Fach



Viktoria hat Armbänder mit farbigen Perlen. Für die Bänder hat sie eine Schachtel mit sieben Fächern. Sie legt ihre Armbänder nur in Fächer mit passendem Farbmuster. Hier ist ein Beispiel für ein Band, das zu einem Fach passt:

Welches Armband passt zu keinem Fach?

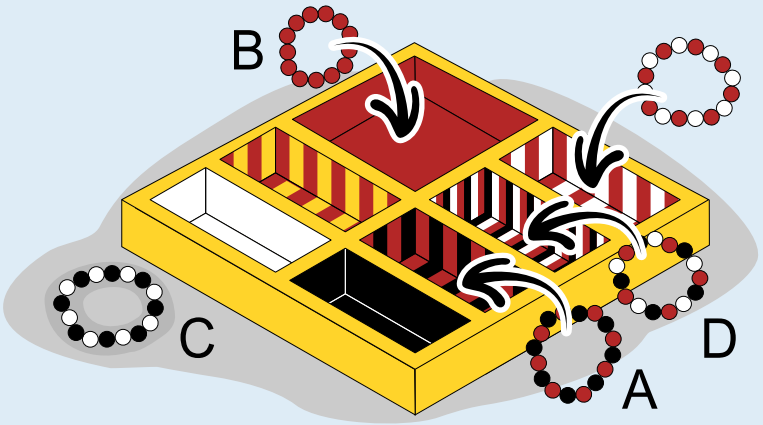
A) **B)** **C)** **D)**



Antwort C ist richtig:

Dieses Bild zeigt, zu welchen Fächern die Armbänder A, B und D passen. Das Farbmuster des Fachs passt jeweils zum Farbmuster des Armbands.

Es gibt aber kein Fach mit dem Farbmuster schwarz und weiß wie bei Armband C.

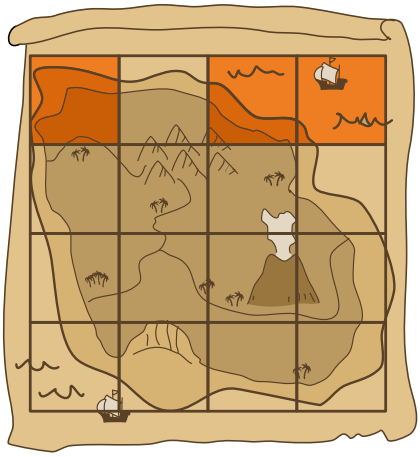


Das ist Informatik!
Viktoria ist ordentlich. Sie legt ihre Armbänder in eine Schachtel mit Fächern. Das Farbmuster eines Fachs muss zu dem Armband passen, das in dieses Fach gelegt wird. So kann Viktoria mit einem Blick jedes Band aus dem richtigen Fach nehmen. Wäre Viktoria weniger ordentlich und würde ihre Armbänder ungeordnet in eine Schublade legen, müsste sie in der Schublade kramen, wenn sie ein bestimmtes Band herausnehmen will. Noch schlechter wäre, wenn in der Schublade auch andere Dinge lägen oder wenn die Bänder in eine große Kiste kämen, die viel mehr Dinge enthalten als eine Schublade.

Computer verwalten viele Daten und legen sie in Speichern mit sehr viel Platz ab. Wenn die Daten ungeordnet in die Speicher kämen, wäre das wie bei einer riesigen Wühlkiste. Es würde dann sehr lange dauern, bestimmte Daten wiederzufinden. Deshalb müssen Informatikerinnen und Informatiker wie Viktoria sein, nämlich sehr ordentlich – zumindest wenn es um die Computerspeicher geht, um die sie sich kümmern. Für Daten gibt es viele verschiedene Ablage-Methoden, mit denen ein Computerprogramm die benötigten Daten schnell finden kann. Am schnellsten geht es mit *Hashing:* Wenn ein Datenelement gespeichert werden soll, wird ein Wert für das Element berechnet und an eine Stelle im Speicher gelegt, die man mit diesem Wert direkt finden kann. Das ist wie bei Viktoria und der Schachtel mit den Farbmustern in dieser Biberaufgabe.

Pip der Pirat

Pip der Pirat sucht einen Schatz auf einer einsamen Insel. Pip hat einen Plan der Insel. Der Plan ist in 16 Quadrate eingeteilt.



Plapper, Pips Papagei, weiß, in welchem Quadrat der Schatz ist. Pip kann Plapper nach einer beliebigen Menge von Quadraten fragen. Plapper sagt ihm dann, ob der Schatz in irgendeinem dieser Quadrate ist oder nicht.

Ein Beispiel: Pip fragt nach den drei Quadraten, die im Bild markiert sind. Wenn Plapper „ja“ sagt, weiß Pip, dass der Schatz in irgendeinem dieser drei Quadrate ist – aber nicht, in welchem.

Pip will so schnell wie möglich sicher wissen, in welchem Quadrat der Schatz ist.

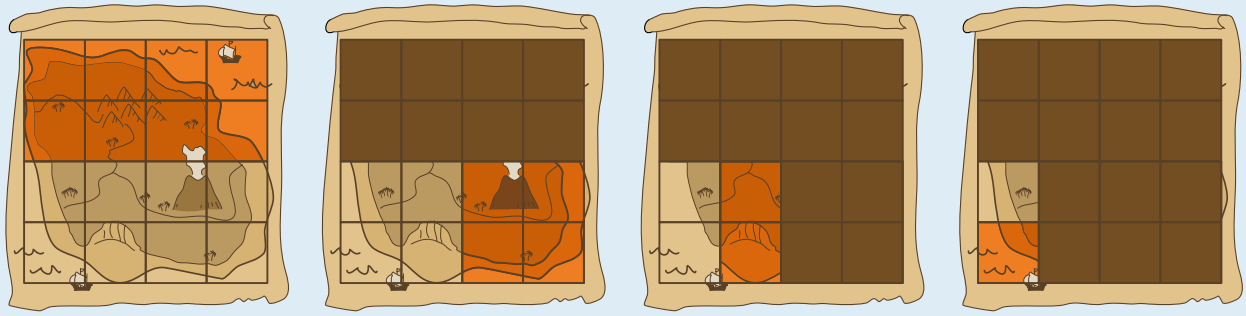
Wie oft muss Pip dazu mindestens nach Quadraten fragen?

4 ist die richtige Antwort.

Um nach vier Fragen sicher zu wissen, in welchem Quadrat der Schatz ist, kann Pip diese Strategie benutzen: Er fragt jeweils genau nach der Hälfte der Quadrate, in denen der Schatz nach den bisherigen Fragen noch sein kann.

Zu Beginn kann der Schatz noch in allen 16 Quadraten sein. In Frage 1 fragt Pip also nach 8 Quadraten davon. Wenn Plapper „ja“ sagt, kann der Schatz in diesen 8 Quadraten sein; wenn Plapper „nein“ sagt, kann der Schatz noch in den anderen 8 Quadraten sein. In Frage 2 fragt Pip dann nach 4 Quadraten aus den übrigen 8, in Frage 3 nach 2 Quadraten aus den übrigen 4 und in Frage 4 nach einem Quadrat aus den übrigen 2. Je nach Antwort auf Frage 4 weiß Pip nun genau, in welchem Quadrat der Schatz ist.

Die Bilder zeigen einen möglichen Verlauf. Pip teilt die übrigen Quadrate jeweils in zusammenhängende Hälften auf. Die angefragten Quadrate sind hellrot markiert. Plapper sagt immer „nein“, und nach vier Fragen weiß Pip, dass der Schatz in dem Quadrat liegt, das im letzten Bild nicht markiert ist.

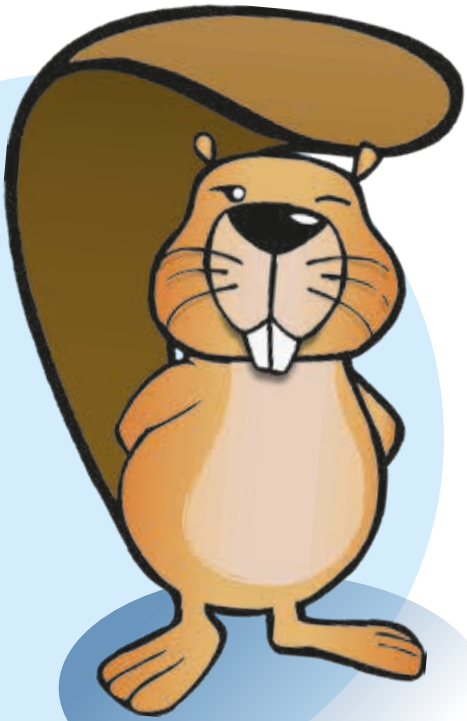


Mit weniger Fragen kann Pip nicht auskommen. Würde er die übrigen Quadrate nicht in zwei gleich große Hälften einteilen, könnte der Schatz im größeren Teil liegen. Für das Abfragen dieses Teils müsste Pip mindestens genau so oft fragen wie für eine Hälfte.

Das ist Informatik!
Die Methode, die Pip bei der Schatzsuche anwendet, heißt in der Informatik binäre Suche. Der Begriff *binär* kommt vom lateinischen Wort „binus“ (deutsch: ein Paar, zwei, zweifach). Bei der binären Suche nach einem Objekt in einer Menge wird die Menge durch eine Reihe von Fragen immer wieder neu halbiert, also in zwei Teile geteilt – deshalb „binär“. Gut halbieren lässt sich eine Menge, wenn die Objekte darin geordnet werden können, z. B. nach Größe; das trifft unter anderem für jede Zahlenmenge zu. Dann gibt es in der Menge ein mittleres Objekt, und man kann das mittlere Objekt mit dem gesuchten vergleichen. Wenn das mittlere Objekt nicht schon das gesuchte ist, weiß man immerhin, in welcher Hälfte der Menge sich das gesuchte Objekt befindet, und durchsucht diese Hälfte wieder binär.

Auf diese Weise kommt man sehr schnell beim gesuchten Objekt an. Bei 1.000 Objekten werden etwa 10 Suchschritte benötigt, bei 1.000.000 Objekten etwa 20. Allgemein kann man sagen: Bei n Objekten werden etwa $\log(n)$ Schritte benötigt; die Funktion log ist der „Zweier-Logarithmus“ oder der Logarithmus zur Basis 2. Weil die binäre Suche so schnell ist, wird sie in Computerprogrammen häufig für die Suche in geordneten Daten verwendet.

In dieser Biberaufgabe ist der Suchraum die Menge der Quadrate auf dem Insel-Plan. Die Quadrate kann man ordnen, indem man sie etwa von oben nach unten und von links nach rechts durchnummeriert. Aber in diesem Fall würde es auch funktionieren, die Menge der Quadrate, in denen der Schatz noch sein kann, beliebig zu halbieren. Dann ist es nur etwas aufwendiger sich zu merken, welche Quadrate-Menge für den nächsten Suchschritt noch in Frage kommt und neu halbiert werden muss.



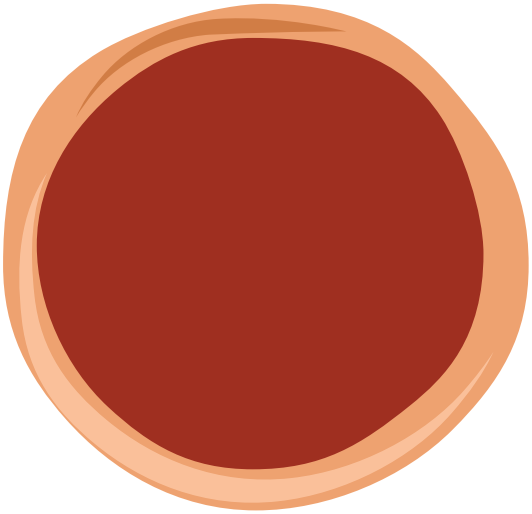
Pizza-Party

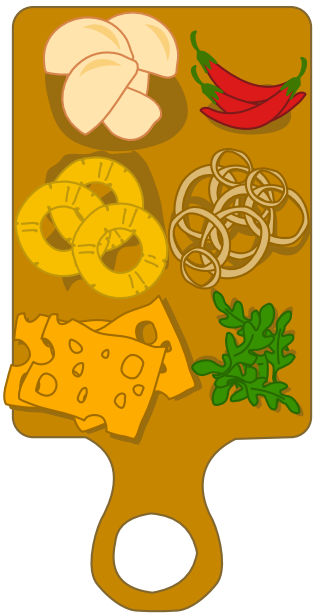
John feiert eine Pizzaparty. Er weiß, welche Beläge sich seine Gäste auf der Pizza wünschen.

Alice	
Bob	
Cem	
Dana	

John möchte eine Pizza mit ingesamt drei Belägen backen. Er wählt drei Beläge so aus, dass er möglichst viele Wünsche seiner Gäste erfüllt.

Mit welchen drei Belägen erfüllt John die meisten Wünsche?





So ist es richtig:

Um die richtige Antwort zu finden, könnte man alle Kombinationen von drei Belägen ausprobieren und für jede Kombination zählen, wie viele Wünsche sie erfüllt. Es gibt aber 20 solcher Kombinationen, sodass dieses Vorgehen mühsam und fehleranfällig ist.

Man muss aber gar nicht über Kombinationen nachdenken. Die Kombination, die die meisten Wünsche erfüllt, besteht aus den drei Belägen, die einzeln am meisten gewünscht werden. Diese Beläge kann man herausfinden, indem man die Tabelle aus der Aufgabenstellung anders organisiert. Für jeden Belag gibt es nun eine eigene Spalte. In der Spalte steht eine 1, wenn der Belag von einem Gast gewünscht wird, und sonst eine 0. Dann addiert man die Zahlen in jeder Spalte.

						
Alice	1	1	0	0	1	0
Bob	0	1	1	0	0	1
Cem	0	1	0	1	1	0
Dana	0	1	0	0	1	1
Summe	1	**4**	1	1	**3**	**2**

Pilze sind am beliebtesten (4), dann kommen Käse (3) und Zwiebeln (2). Wenn John diese beliebtesten drei Beläge auf die Pizza legt, erfüllt er die meisten Wünsche, nämlich insgesamt 9. Mit keiner anderen Zusammenstellung von Belägen kann er mehr Wünsche erfüllen.

Das ist Informatik!
Das Pizza-Belag-Auswahl-Problem in dieser Biberaufgabe ist ein *Optimierungsproblem.* John sucht nicht irgendeine Belag-Kombination für seine Pizza, sondern die beste – oder, besser gesagt: eine beste, denn es wäre ja denkbar, dass es mehrere Belag-Kombination gibt, welche die meisten Wünsche erfüllen.

Um die beste Lösung eines Optimierungsproblems zu finden, muss man
- Daten erfassen (hier: die Wünsche der Freunde),
- neue Daten berechnen (die Häufigkeiten der Wünsche),
- Bedingungen berücksichtigen (es soll nur eine Pizza mit genau drei Belägen gebacken werden) und
- eine Entscheidung fällen (die Auswahl der Beläge).

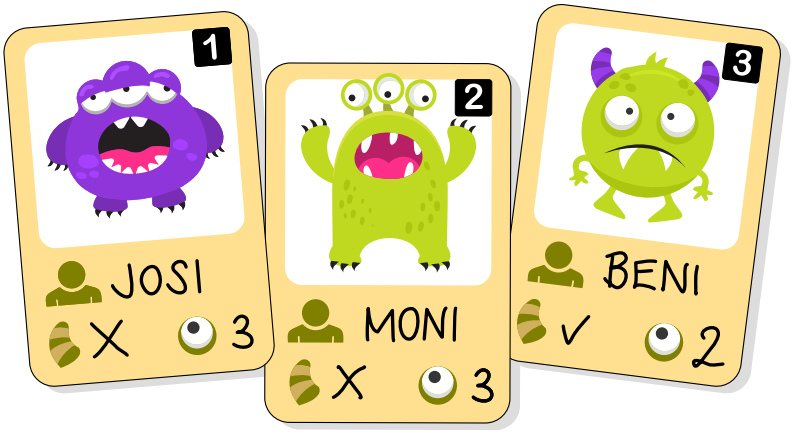
Für die Entscheidung bei der Lösung eines Optimierungsproblems muss man mögliche Antworten bzw. Lösungen miteinander vergleichen können. Dazu benötigt man ein *Maß* dafür, wie gut eine Lösung ist. In dieser Biberaufgabe ist die Anzahl der erfüllten Wünsche das Optimierungsmaß: Je mehr Wünsche erfüllt sind, desto besser. Der Optimierungsalgorithmus konnte die Lösung mit dem besten Wert für das Optimierungsmaß direkt auswählen, durch Addition der Wünsche für die einzelnen Beläge. In diesem Fall war es also nicht nötig, unterschiedliche Lösungen miteinander zu vergleichen.

Optimierungsalgorithmen werden in der Informatik für viele Zwecke entwickelt, zum Beispiel für Produktionsprozesse in der Industrie oder zur Berechnung von Routen für die Auslieferungsfahrzeuge von Paket- oder anderen Lieferdiensten. Dabei kann es ganz unterschiedliche Optimierungsmaße geben, etwa die Geschwindigkeit des Produktionsprozesses oder die Länge der Route. Eines ist aber immer gleich: Man muss sich überlegen, wie man möglichst geschickt aus der Menge der möglichen Lösungen eine beste Lösung auswählt. So leicht wie in dieser Biberaufgabe geht das nur selten.

Ricca-Karten 1

Barbara sammelt Karten mit Monstern, den „Riccas“.
Auf jeder Karte sind die Eigenschaften eines Riccas angegeben: der Name , die Anzahl der Augen und ob das Ricca Hörner hat .

Hier sind drei Ricca-Karten, auf denen die Werte der Eigenschaften stehen:



Für jede Eigenschaft ist festgelegt, welche Werte erlaubt sind: Barbara bekommt vier Karten für ein weiteres Ricca. Aber nur auf einer Karte haben alle Eigenschaften erlaubte Werte.

Auf welcher?

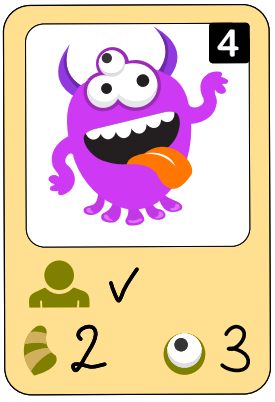
Symbol	Eigenschaft	erlaubte Werte
	Name	Text (mehrere Buchstaben)
	hat Hörner	✓ oder X
	Anzahl der Augen	Zahlen

A)

B)

C)

D)



Antwort B ist richtig!

Die Tabelle zeigt, welche Werte die Eigenschaften auf den vier Karten haben:

	A)	B)	C)	D)
	LORI	LORI	LORI	✓
	2	✓	2	2
	✓	3	3	3

Nur auf der Karte von Antwort B haben alle Eigenschaften erlaubte Werte, nämlich den Text LORI bei  , einen ✓ bei  und eine Zahl bei  .

Auf den Karten von den Antworten A, C und D steht bei  jeweils eine Zahl, nämlich 2. Das Ricca hat zwar zwei Hörner, aber die Eigenschaft  soll nur angeben, ob das Ricca Hörner hat oder nicht. Deshalb sind nur die Werte ✓ und X erlaubt.

Auf der Karte von Antwort A ist bei  der Wert ✓angegeben. Das Ricca hat Augen, aber die Eigenschaft  soll die Anzahl der Augen angeben. Deshalb ist nur eine Zahl erlaubt.

Das ist Informatik!
Die Eigenschaften und deren Werte auf den Karten beschreiben die Riccas. Die Bilder enthalten jedoch viel mehr Informationen über die Riccas, als auf den Karten notiert ist. So ist zum Beispiel nicht nur zu sehen, ob ein Ricca Hörner hat oder nicht, sondern auch wie viele Hörner es hat. Es ist wichtig, sich gut zu überlegen, welche Werte man für die Eigenschaften erlaubt.

Wenn man zum Beispiel herausfinden will, wie viele Hörner die Riccas auf den Karten insgesamt haben, muss man sie auf den Bildern zählen. Damit das schneller geht, wäre es gut, für die Eigenschaft  Zahlen als Werte zu erlauben, so wie bei den Augen. Dann gibt die Eigenschaft  die Anzahl der Hörner an.

Auch bei Computer-Programmen werden als Daten oft Werte für Eigenschaften gespeichert, wie bei den Ricca-Karten in dieser Biberaufgabe. Zum Beispiel werden im Computer-Katalog einer Bücherei für jedes Buch unter anderem der Titel, die Autorin oder der Autor und die Anzahl der Ausleihen gespeichert. Je nachdem, welche Werte für diese Eigenschaften erlaubt sind, kann das Programm unterschiedliche Berechnungen oder Operationen ausführen. Die Informatik nennt die Menge der für eine Eigenschaft erlaubten Werte den *Datentyp* dieser Eigenschaft. Wenn die Anzahl der Ausleihen den Datentyp Zahl hat, kann das Computer-Programm der Bücherei leicht herausfinden, welche Bücher am häufigsten ausgeliehen wurden.

Ricca-Karten 2

Barbara sammelt Karten mit Monstern, den „Riccas“. Hier sind ihre Ricca-Karten:

Auf jeder Karte sind die Eigenschaften eines Riccas angegeben, zum Beispiel der Name ( ) oder ob das Ricca Zähne hat ( ). Die Eigenschaften haben Werte.
Auf Karte 2 zum Beispiel hat den Wert MONI, und hat den Wert ✓: Das Ricca auf der Karte heißt also MONI und hat Zähne.

Barbara erkennt, dass es nur drei verschiedene Arten von Werten auf den Karten gibt. Sie nennt sie „Typen“:

Text	mehrere Buchstaben hintereinander
Zahlen	also 1, 2, 3 usw.
ja/nein	die Zeichen ✓ und X, die „ja“ und „nein“ bedeuten

Ordne den Eigenschaften die richtigen Typen zu.

ja/nein ja/nein Text Zahlen Zahlen

So ist es richtig:

				
Text	Zahlen	Zahlen	ja/nein	ja/nein

Diese Tabelle enthält für alle sechs Karten die Werte der fünf Eigenschaften:

Karte					
1	JOSI	3	2	X	✓
2	MONI	3	2	X	✓
3	KILI	2	2	✓	✓
4	BENI	2	2	X	✓
5	LORI	2	5	X	✓
6	PHIL	2	2	X	✓

Aus der Tabelle können wir nun ablesen:

- Alle Werte der Eigenschaft  sind vom Typ Text.
- Alle Werte der Eigenschaften  und  sind vom Typ Zahlen.
- Alle Werte der Eigenschaften  und  sind vom Typ ja/nein.

Das ist Informatik!
Barbara hat festgestellt, dass auf ihren Ricca-Karten verschiedene Typen von Werten verwendet werden. Diese Typen zu unterscheiden, kann bei der Organisation der Karten helfen. Weil Barbara weiß, dass alle Werte der Eigenschaft  Zahlen sind, kann sie die Anzahlen der Beine der Riccas miteinander vergleichen und die Karten nach der Anzahl der Beine sortieren.

Eine Ricca-Karte ist wie ein Datensatz, der im Computer gespeichert ist. Die Elektronik von Computer-Speichern kann aber nur 1en und 0en speichern. Auch Computer-Programme verwenden deshalb „Wert-Typen“, die in der Informatik *Datentypen* heißen, um festzulegen, wie der Computer diese 1en und 0en interpretieren und manipulieren soll. Außerdem kann mit Hilfe von Datentypen geprüft werden, ob und wie gespeicherte Daten miteinander kombiniert werden können.

Die Wert-Typen aus dieser Biberaufgabe entsprechen wichtigen Datentypen aus der Programmierung:

Text wird in Programmiersprachen häufig als „string“ bezeichnet: Daten dieses Typs sind Folgen von Zeichen (Buchstaben, Ziffern und andere Zeichen).

Zahlen entspricht dem Typ „integer“: Daten dieses Typs sind ganze Zahlen, also … -2, -1, 0, 1, 2, … .

ja/nein entspricht dem Typ „boolean“: Dieser Datentyp kennt nur zwei verschiedene Werte, die häufig `true` und `false` (wahr und falsch) heißen.

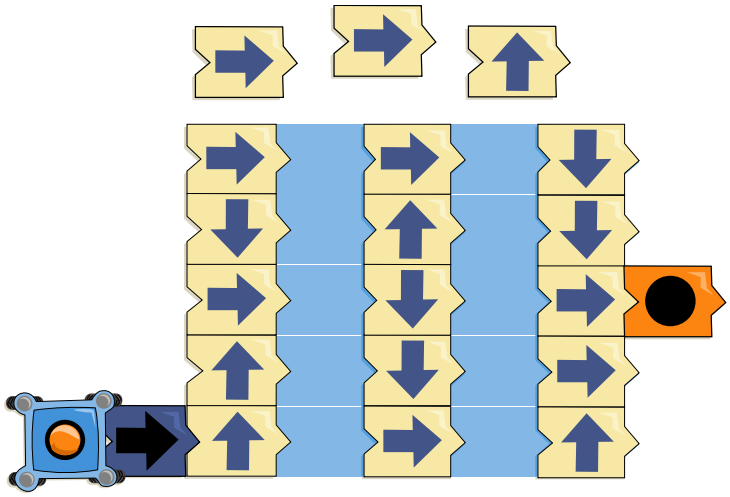
Robertas Roboter

Robertas Roboter kann nur über besondere Kärtchen fahren. Mit den Kärtchen kann man einen Wegeplan legen.

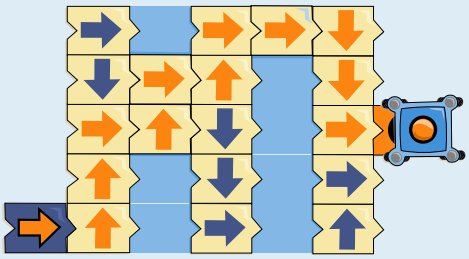
Der Roboter beginnt immer auf dem Start-Kärtchen und soll das Ziel-Kärtchen erreichen. Von einem Kärtchen mit Pfeil fährt er in Pfeilrichtung zum nächsten Kärtchen.

Unten siehst du einen Wegeplan. Darauf fährt der Roboter zunächst nach rechts, dann hoch und noch einmal hoch. Dann würde er gerne nach rechts fahren, aber dort ist kein Kärtchen! Der Wegeplan ist noch nicht fertig. Es gibt noch drei Kärtchen und zehn Stellen, an die sie gelegt werden können.

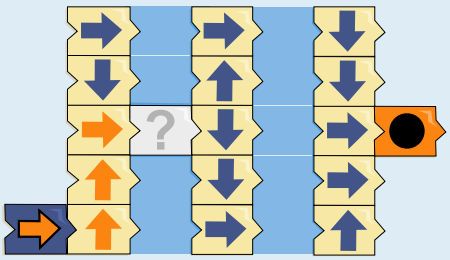
Lege die drei Kärtchen so, dass Robertas Roboter das Ziel erreicht.



So ist es richtig:

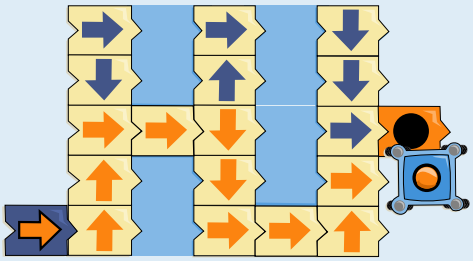


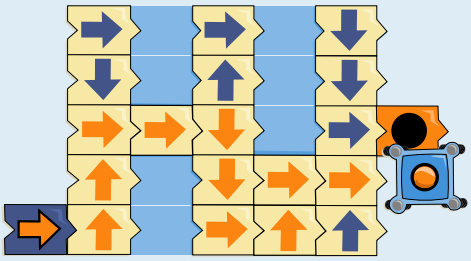
Wie bereits in der Aufgabe angedeutet, wird Robertas Roboter zunächst nach rechts, zweimal hoch und einmal wieder nach rechts fahren. Damit muss ein Kärtchen an die mit ? markierte Stelle gelegt werden:



Die beiden Arten von Kärtchen, die zur Verfügung stehen, sind „hoch“  oder „rechts“  . Falls  gewählt würde, würde der Roboter also nach rechts fahren, danach zweimal runter und wieder rechts. Nun könnte man entweder wieder  oder  und darüber  wählen.

In beiden Fällen würde der Roboter unter dem Ziel landen:





Also muss  gewählt werden. Als zweites Kärtchen kann nun nur noch  gewählt werden, weil das einzige  bereits verwendet wurde. Dann würde der Roboter also nach rechts, hoch, nach rechts und von dort hoch und nach rechts fahren. An dieser Stelle ist das letzte verfügbare Kärtchen  zu legen, so dass der Roboter also nach rechts, zweimal runter und einmal nach rechts fahren kann und im Ziel landet.

Es gibt also nur eine richtige Lösung.

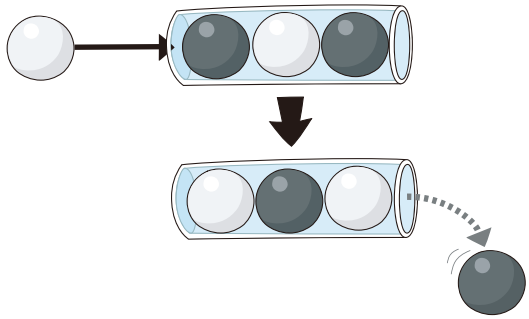
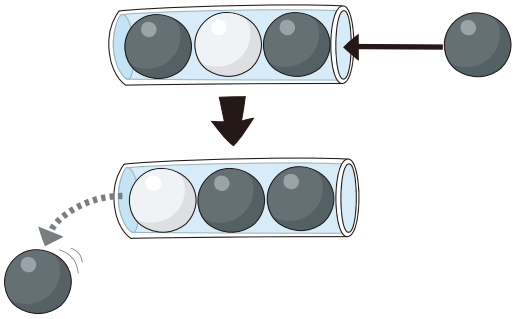
Man kann sich diese Lösung übrigens auch schneller rückwärts überlegen: Zum Ziel kommt man nur von den obersten drei möglichen Positionen in der rechten hellblauen Spalte. Dorthin kommt man jedoch nur ganz oben, damit muss ganz oben rechts schon mal  liegen. Dorthin kommt man jedoch nur von den obersten beiden Kärtchen in der Mitte, also muss man in der linken hellblauen Spalte noch einmal  legen, um dort hinzukommen.

Das ist Informatik!
In der Regel wird man dieses Problem so lösen wie in der Antworterklärung beschrieben: Man wird mal einen Weg ausprobieren, und wenn er nicht zum Ziel führt, wird man zurückgehen und einen anderen Weg ausprobieren. Dies nennt man in der Informatik eine *Tiefensuche,* weil zunächst ein Weg vollständig ausprobiert wird und dann erst ein nächster probiert wird. Weil man dazu oftmals einen oder mehrere Schritte zurückgehen muss, um eine andere Lösung auszuprobieren, macht man *Backtracking.* In jedem Fall jedoch löst man das Problem vorwärts. Manche Probleme jedoch lassen sich besser rückwärts lösen, indem man vom Ziel her schaut, wie man dort hinkommen könnte. Eigentlich ist diese Art von Problem nicht gut geeignet, dass man es rückwärts löst, denn während man vorwärts immer weiß, welches das nächste Kärtchen ist, kann es rückwärts sein, dass man von mehreren Kärtchen kommt (der Pfeil zeigt vorwärts und rückwärts muss man für alle Kärtchen drum herum schauen, ob sie in Richtung des Kärtchens zeigen). Aber in diesem speziellen Fall ist gleichwohl schnell zu erkennen, welche Lösung die einzig richtige sein kann.

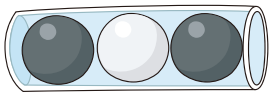
Röhre

In eine durchsichtige Röhre passen genau drei Kugeln. Die Röhre ist an beiden Seiten offen.

Wenn in die volle Röhre von einer Seite eine Kugel hinein geschoben wird, fällt auf der anderen Seite eine Kugel heraus. Hier sind zwei Beispiele:

Weiße Kugel von links	Schwarze Kugel von rechts
	

Die Röhre ist mal wieder voll:

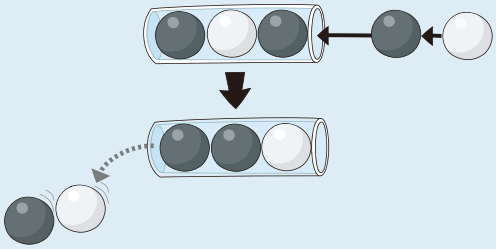


Nun werden nacheinander vier Kugeln in die Röhre geschoben: Zuerst eine schwarze und dann eine weiße Kugel von rechts, danach eine schwarze und dann eine weiße Kugel von links:

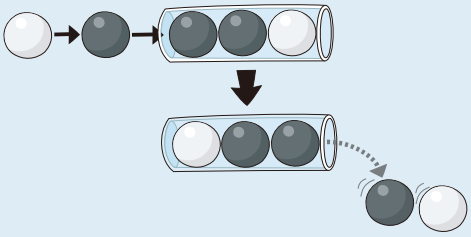
Welche drei Kugeln sind am Ende in der Röhre?

So ist es richtig:

Es werden zwei Kugeln von rechts in die Röhre geschoben, zuerst eine schwarze und dann eine weiße Kugel. Daher fallen links zwei Kugeln raus, zuerst die schwarze, dann die weiße.



Danach werden zwei Kugeln von links in die Röhre geschoben, zuerst eine schwarze und dann eine weiße Kugel. Daher fallen diesmal rechts zwei Kugeln raus, zuerst die weiße, dann die schwarze.



Am Ende sind also (von links nach rechts) eine weiße, eine schwarze und noch eine schwarze Kugel in der Röhre.

Man kann die richtige Antwort auch einfacher bestimmen: Da zuerst von rechts zwei Kugeln in die Röhre geschoben werden und danach von links zwei Kugeln, ist die ursprünglich rechts in der Röhre liegende Kugel wieder rechts. Links davon sind (von rechts nach links) die beiden zuletzt in die Röhre geschobenen Kugeln.

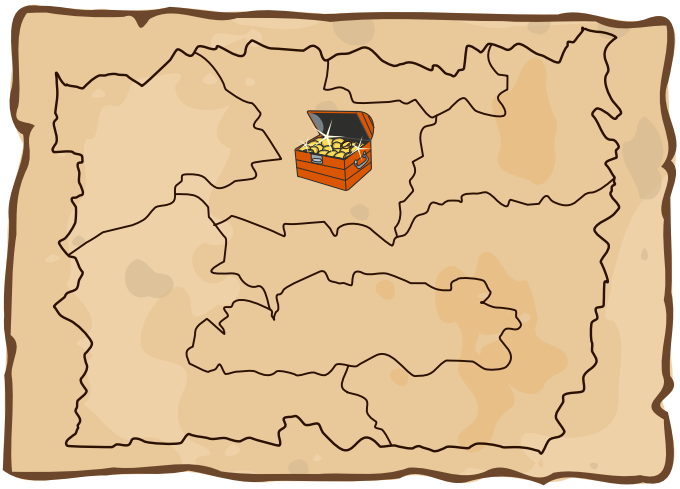
Das ist Informatik!
Die Röhre in dieser Biberaufgabe ist ein spezielle Form der Warteschlange, die in der Informatik *Deque* genannt wird (kurz für engl. *Double-Ended Queue, „Warteschlange mit zwei Enden“).* Im Unterschied zu einer herkömmlichen Warteschlange können wir an beiden Enden einer Deque ein Element hinzufügen und von beiden Enden der Deque ein Element entfernen. Eine Deque ist eine *Datenstruktur.*

Deques sind vielfältig einsetzbar, z. B. als verallgemeinerter Ersatz sowohl von *Stapeln* (vorne hinzufügen und vorne entfernen) als auch einfachen *Warteschlangen* (hinten hinzufügen und vorne entfernen). Man kann sich eine Deque wie ein Rangiergleis vorstellen: vorne und hinten können Waggons an- oder abgehängt werden, aber nicht einfach ein Waggon aus der Mitte herausgenommen werden. Bei einem einfachen Kopfgleis hingegen könnte man nur an einer Seite Waggons an- und abhängen (wie bei einem Stapel) und auf einem so genannten Ablaufberg bewegen sich die Waggons immer nur in einer Richtung (wie bei einer Warteschlange).

So wie die Röhre dieser Aufgabe nur bis zu drei Bälle enthalten kann, hat auch eine Deque praktisch eine begrenzte Kapazität. Wenn einer bereits vollkommen ausgelasteten Deque ein Element hinzugefügt wird, muss am anderen Ende Platz gemacht werden. Dies geschieht, indem dort ein Element entfernt wird und die entsprechende Information verloren geht. Speicherplatz ist eine endliche Ressource und die Menge der Daten, die man speichern kann, hängt davon ab. Gleichwohl werden Deques häufig als hypothetisch unbegrenzt verwendet, insbesondere wenn es eher um theoretische Fragestellungen als um praktische Anwendungen geht.

Schatzkarte

Die Landkarte zeigt das Reich der Biberkönigin mit seinen sieben Provinzen. In einer Provinz hat die Königin ihren Schatz versteckt.



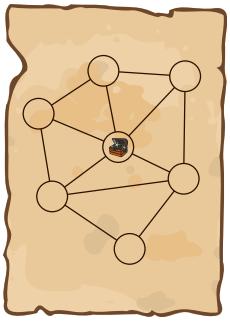
Die Königin will die Lage des Schatzes geheim halten. Deshalb hat sie eine besondere Schatzkarte gezeichnet.

Für jede Provinz ist darin ein Kreis eingezeichnet. Eine Linie zwischen zwei Provinz-Kreisen zeigt, dass die beiden Provinzen aneinander angrenzen. Der Kreis für die Provinz mit dem Schatz ist markiert.

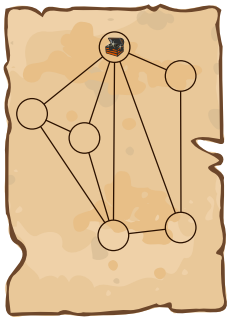
Um mögliche Räuber zu verwirren, hat die Königin zusätzlich vier falsche Schatzkarten gezeichnet.

Welche ist die richtige Schatzkarte?

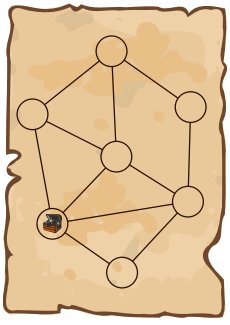
A)



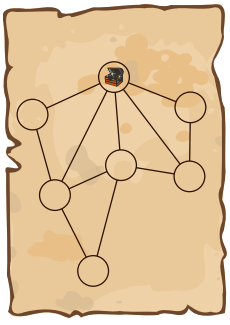
B)



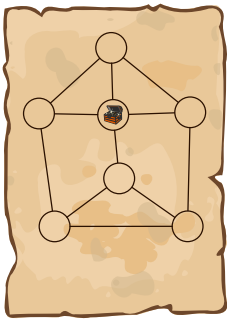
C)



D)

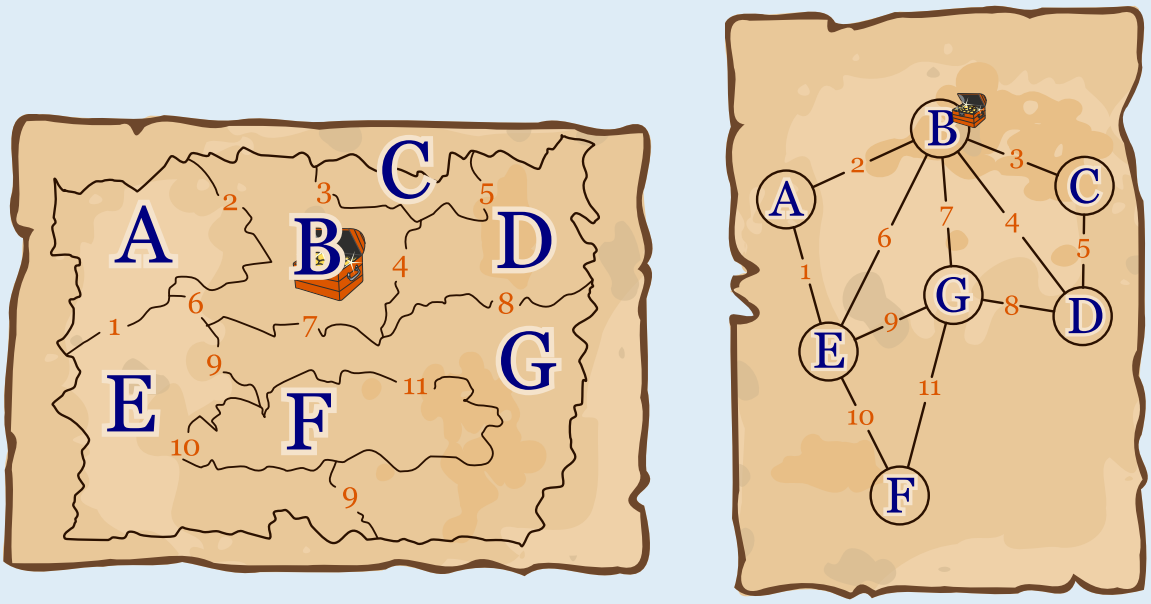


E)



Antwort D ist richtig:

In der Landkarte links sind die Provinzen mit den Buchstaben A bis G gekennzeichnet. Wenn zwei Provinzen aneinander angrenzen, ist ihre gemeinsame Grenze mit einer Zahl bezeichnet; insgesamt mit den Zahlen 1 bis 11. Die Grenze zwischen E und G hat zwei Abschnitte, doch es handelt sich um dieselbe gemeinsame Grenze. Daher sind beide Grenzabschnitte mit der Zahl 9 gekennzeichnet. Indem wir die Buchstaben A bis G und die Zahlen 1 bis 11 in die Schatzkarte D rechts übertragen, erkennen wir, dass diese die richtige Schatzkarte ist.



- Schatzkarte A ist falsch: Die drei Provinzen A, C und F grenzen jeweils nur an zwei andere Provinzen an. Daher muss es auf der richtigen Schatzkarte drei Kreise geben, von denen jeweils genau zwei Linien ausgehen. Es gibt aber nur einen Kreis, von dem genau zwei Linien ausgehen.
- Schatzkarte B ist falsch: Sie hat nur sechs Kreise, es gibt aber sieben Provinzen.
- Schatzkarte C ist falsch: Die Provinz B mit dem Schatz grenzt an die fünf Provinzen A, C, D, E und G an. Also müssen fünf Linien vom markierten Kreis ausgehen, es sind aber nur vier.
- Schatzkarte E sieht der Landkarte am ähnlichsten. Aber auch sie ist falsch, denn auch hier gehen nur vier Linien vom markierten Kreis aus.

Das ist Informatik!
Die Schatzkarten sind Diagramme, die nur ausgewählte Eigenschaften der Landkarte darstellen. In dieser Biberaufgabe ist die dargestellte Eigenschaft, dass zwei Provinzen aneinander grenzen. Sie wird durch eine verbindende Linie symbolisiert. Vernachlässigt werden die Form dieser Provinzen, die Größe, Lage, Richtungen und Entfernungen. Ein Beispiel: In der Karte grenzt C an die nördlichen Teile von B und D; im Diagramm ist das nicht erkennbar. Um ein Problem mit Hilfe von Computern lösen zu können, wird auch in der Informatik die Realität auf die für die Lösung des Problems wichtigen Eigenschaften reduziert. Informatikerinnen und Informatiker sagen: Zur *Modellierung* eines Problems wird *abstrahiert.*

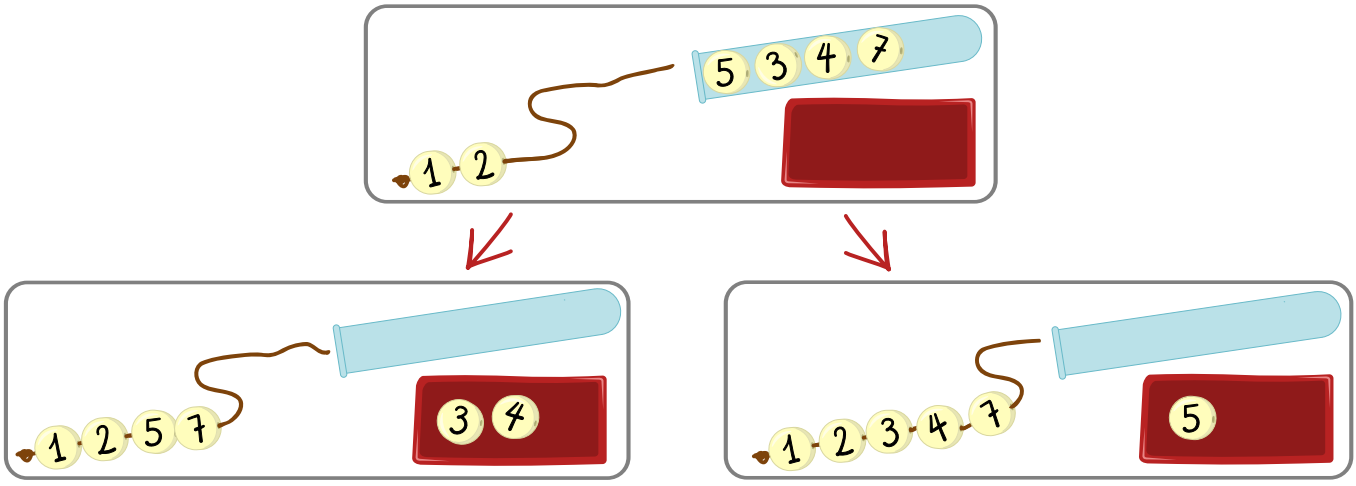
Übrigens: Die Schatzkarten-Diagramme sind so genannte Graphen. Ein Graph besteht aus *Knoten und Kanten.* Die Knoten stehen für bestimmte Einheiten (hier: die Provinzen), die Kanten repräsentieren Beziehungen zwischen den Einheiten (hier: aneinander grenzen). Graphen können auch noch weitere Eigenschaften haben. Die Graphentheorie, ein Teilgebiet von Mathematik und Informatik, stellt Wissen und Methoden zu Graphen zur Verfügung. Modelliert man ein Problem mit Hilfe von Graphen, kann man darauf zurückgreifen.

Scotts Armbänder

Scott hat Perlen mit Nummern und macht Armbänder daraus. Die Perlen kommen nacheinander aus einem Glasröhrchen. Für jede Perle entscheidet Scott: Entweder verwendet er die Perle fürs Armband und fädelt sie auf die Schnur, oder er legt sie weg.

Scott kann eine Perle nur dann verwenden, wenn
- die Schnur leer ist oder
- die Nummer der Perle größer ist als die Nummer der letzten Perle auf der Schnur.

In diesem Beispiel ist zunächst 2 die letzte Perle auf der Schnur:

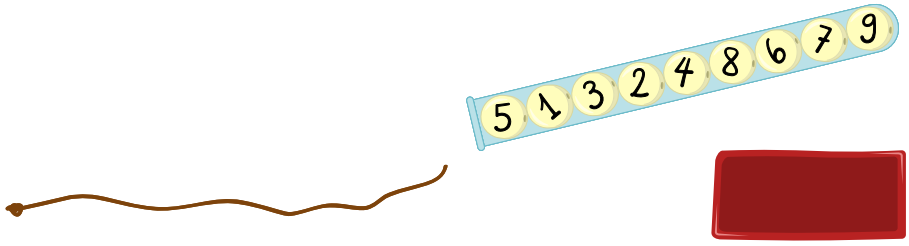


Die nächste Perle 5 kann Scott verwenden, aber auch weglegen. Wenn er

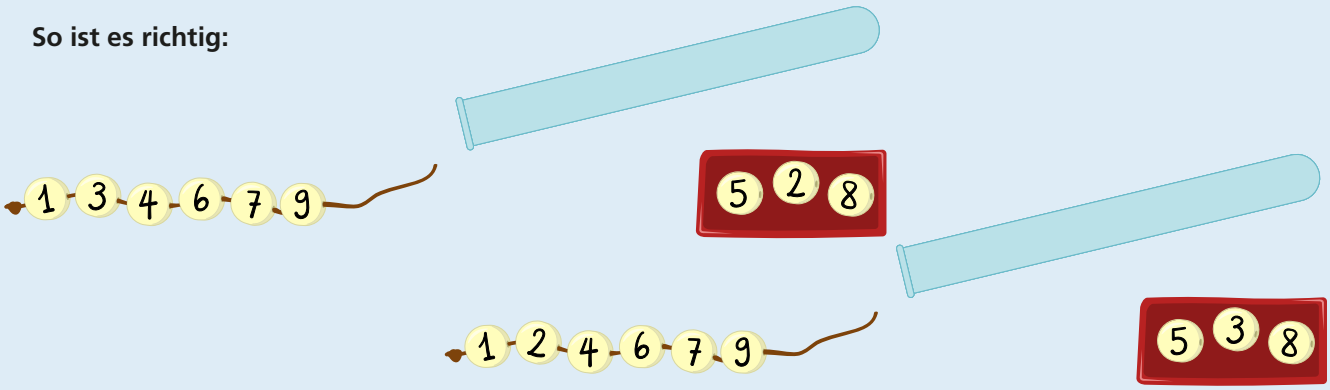
- 5 verwendet, kann er ein Armband mit vier Perlen machen: 1 2 5 7
- weglegt, kann er ein Armband mit fünf Perlen machen: 1 2 3 4 7.

Scott bekommt ein neues Röhrchen mit Perlen. Er will daraus ein Armband mit möglichst vielen Perlen machen.

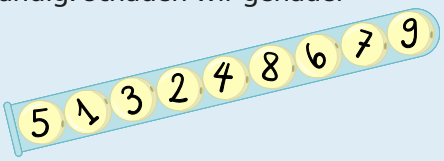
Wie kann Scott das schaffen?
Entscheide für jede Perle, ob sie verwendet oder weggelegt wird.

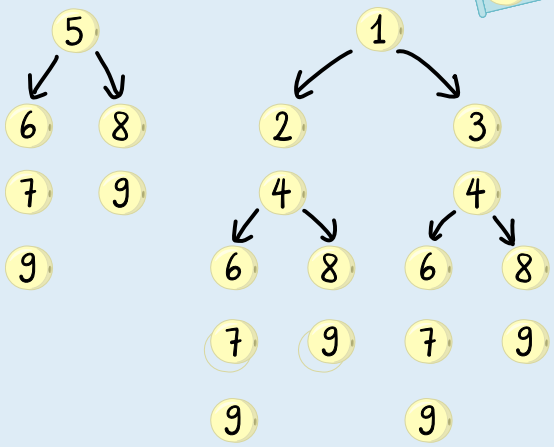


So ist es richtig:



Wir könnten für dieses Röhrchen alle Armbänder auflisten, die Scott aus den Perlen machen kann, und sehen, welche die meisten Perlen haben. Das ist allerdings aufwändig. Schauen wir genauer hin und betrachten nur noch die Nummern auf den Perlen – denn am Ende sind Scotts Armbänder Folgen von aufsteigend sortierten Nummern.

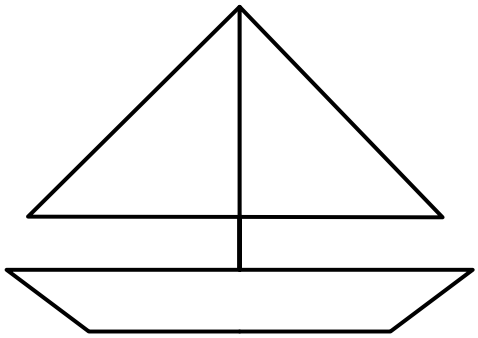




Verwendet Scott die erste Nummer 5 aus dem Röhrchen, kann er danach nur noch 6, 7, 8 und 9 verwenden und die Armbänder 5679 und 589 machen. Wenn Scott die 5 weglegt und die nächste Nummer 1 als erste fürs Armband verwendet, kann er mehr Armbänder mit mehr Perlen machen. Dieser Entscheidungsbaum zeigt, wie Scott die Perlen verwenden und welche Armbänder er machen kann: Würde Scott auch die 1 weglegen, kann er nur noch Armbänder machen, die weniger Perlen haben als möglich. Verwendet Scott z. B. die 2 zuerst, kann er u. a. noch die Armbänder 24679 und 2489 machen. Diese sind aber in den Armbändern 124679 bzw. 12489 enthalten, die Scott machen kann, wenn er zuerst die 1 verwendet. Armbänder mit möglichst vielen Perlen beginnen also mit 1 und bestehen aus 6 Perlen: 124679 oder 134679.

Das ist Informatik!
Jedes Röhrchen in dieser Biberaufgabe enthält eine *Folge* von nummerierten Perlen, die wir als Folge von Zahlen auffassen können. So wie Scott die Perlen verwendet, ist jedes seiner Armbänder eine *aufsteigende Teilfolge* der „Röhrchen-Folge“; *aufsteigend* bedeutet, dass jede Zahl in der Teilfolge (mit Ausnahme der ersten Zahl) größer ist als die Zahl davor. Um ein Armband mit möglichst vielen Perlen zu machen, muss Scott die *längste* aufsteigende Teilfolge bestimmen.
Bei langen Zahlenfolgen würde es sehr viel Zeit benötigen, alle möglichen aufsteigenden Teilfolgen zu bilden, um dann die längste zu bestimmen. Wenn das Röhrchen z. B. 20 Perlen enthält, liegt der Rechenaufwand schon in der Größenordnung von einer Million Arbeitsschritten.
Zum Glück hat man in der Informatik Algorithmen entwickelt, die die längste aufsteigende Teilfolge schneller bestimmen können. Dabei wird eine Technik angewendet, die **Dynamisches Programmieren** heißt. Die Größenordnung des Rechenaufwands liegt bei diesen schnellen Algorithmen für ein Röhrchen mit 20 Perlen unterhalb von hundert Arbeitsschritten.

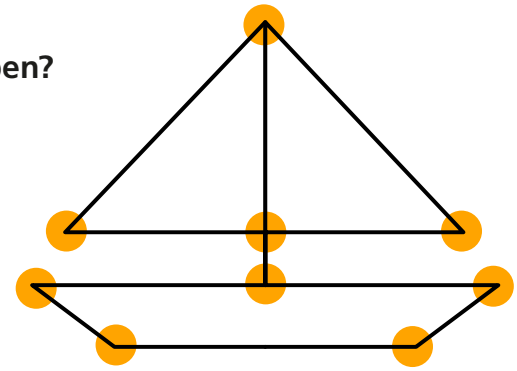
Segelboot



Sophie hat das Segelboot gezeichnet.

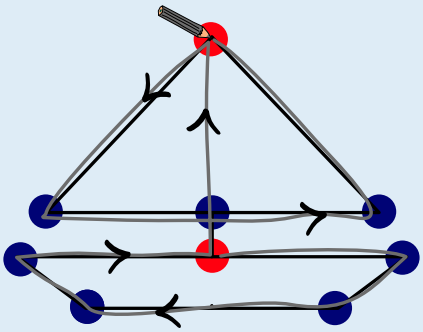
Sie hat
- das Segelboot in einem Zug gezeichnet, den Stift also niemals angehoben, und
- jede Linie nur genau einmal gezeichnet.

An welchen der unten markierten Punkte kann Sophie mit dem Zeichnen begonnen haben?



So ist es richtig:

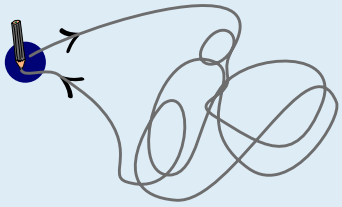
Sophie kann an beiden roten Punkten beginnen. Wenn sie beim unteren roten Punkt beginnt, zeichnet sie zuerst den Rumpf des Bootes nach, danach den Mast und das Segel und kommt schließlich beim oberen roten Punkt an (siehe Bild). Beginnt sie beim oberen roten Punkt, kann sie genau umgekehrt vorgehen.

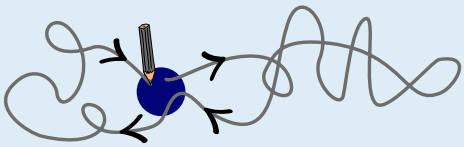


Die beiden Möglichkeiten kann man durch Ausprobieren finden. Man kann aber auch systematisch bestimmen, an welchem Punkt Sophie beginnen kann. Man kann zwischen zwei Arten von Punkten unterscheiden:
- Von den blauen Punkten geht jeweils eine **gerade** Anzahl Linien aus.
- Von den roten Punkten geht jeweils eine **ungerade** Anzahl Linien aus.

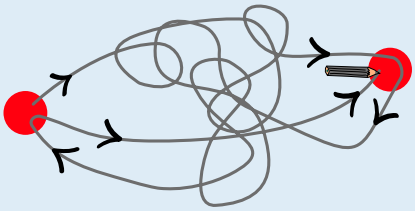
Sophie kann entweder bei einem blauen oder bei einem roten Punkt beginnen. Wir probieren beide Alternativen aus:

Sophie beginnt bei einem blauen Punkt und zeichnet die erste Linie nach. Um alle von diesem Punkt ausgehenden Linien nachzuzeichnen, muss sie zu diesem Punkt zurückkehren - einmal oder mehrmals. Das ist sicher dann möglich, wenn es nur blaue Punkte gibt: Bei den weiteren blauen Punkten gibt es für jede Linie, über die Sophie beim Nachzeichnen zu diesem Punkt kommt, immer eine noch unbenutzte Linie, über die Sophie den Punkt auch wieder verlassen kann.





Sophie beginnt bei einem roten Punkt. Wegen der ungeraden Anzahl an ausgehenden Linien muss Sophie diesen Punkt irgendwann über die letzte nicht nachgezeichnete Linie verlassen und kann dann nicht mehr zu diesem Punkt zurück. Sie muss also einen anderen Endpunkt finden. Hier bietet sich der zweite rote Punkt an: Wegen der ungeraden Anzahl an Linien kann Sophie diesen Punkt irgendwann nicht mehr verlassen. Es gibt also einen Zusammenhang zwischen den zwei roten Punkten: Sophie muss in einem roten Punkt beginnen und im anderen roten Punkt enden.



Sophie konnte das Segelboot so – also in einem Zug und jede Linie genau einmal – nachzeichnen, weil es genau zwei rote Punkte hat. In der richtigen Antwort müssen beide roten Punkte angegeben sein, obwohl es auch schon eine gute Leistung ist, genau einen richtig zu erkennen.

Das ist Informatik!
Die Zeichnung vom Segelboot besteht aus Punkten und aus Linien, die einige der Punkte verbinden. In der Informatik sind Strukturen aus Dingen, von denen manche miteinander verbunden sind, als *Graph* bekannt. Ein Graph besteht aus *Knoten* und *Kanten.* Die Segelboot-Zeichnung ist also die Darstellung eines Graphen: Die Punkte sind die Knoten, und die Linien sind die Kanten, die jeweils zwei Knoten miteinander verbinden. Ein Knoten kann mehrere Kanten haben, die von ihm ausgehen, also ihn mit anderen Knoten verbinden. Ein Weg in einem Graphen ist durch eine Folge von Knoten bestimmt und geht über Kanten, die jeweils zwei im Weg aufeinanderfolgende Knoten miteinander verbinden.

In dieser Biberaufgabe will Sophie einen Weg durch den Segelboot-Graphen finden, der durch alle Knoten geht und genau einmal über jede Kante. Die Knoten dürfen dabei beliebig oft besucht werden. Damit das Problem lösbar ist, darf es keine Knoten geben, die man gar nicht erreichen kann. Ein Graph, der diese Voraussetzung erfüllt, ist *zusammenhängend.* Das Haus vom Nikolaus ist ein bekanntes Rätsel, bei dem man einen solchen Weg in einem zusammenhängenden Graph finden muss.

Der Schweizer Mathematiker Leonhard Euler (1707-1783) hat massgeblich dazu beigetragen, diese Art von Problemen zu verstehen und zu lösen. In Anlehnung an seine Forschungsarbeit ist ein Weg, wie ihn Sophie durch den Segelboot-Graphen finden will, in der Informatik als *Eulerweg* bekannt. Ein bekannter Algorithmus, der festellt, ob es in einem zusammenhängenden Graphen einen Eulerweg gibt, überprüft, ob ein Graph Knoten mit einer *ungeraden Anzahl* ausgehender Kanten hat. Einen Eulerweg gibt es in genau zwei Fällen:

- Wenn es *genau zwei* solcher Knoten gibt, dann startet der Eulerweg in einem dieser Knoten und endet im anderen Knoten.
- Hat ein Graph nur Knoten mit einer geraden Anzahl ausgehender Kanten, dann kann man von *jedem Knoten* aus einen Eulerweg konstruieren. Alle diese Eulerwege haben eine kreisartige Form, weil sie im selben Knoten starten und enden. In der Informatik redet man dann von einem *Eulerkreis.*

Eulerwege und Eulerkreise haben zahlreiche Anwendungen im alltäglichen Leben. Wenn man einen Spaziergang durch ein Dorf plant, kann man versuchen einen Eulerweg oder einen Eulerkreis zu konstruieren. Findet man einen, dann kann man alle Straßen nacheinander besuchen ohne eine Straße jemals doppelt zu benutzen.

Sonnige Tage

Tom sagt: „An sonnigen Tagen schwimmt in jedem Teich mindestens ein Biber.“

Kim antwortet: „Das ist nicht wahr. Am letzten Sonntag wurde deine Aussage widerlegt.“

Was ist am letzten Sonntag passiert? Fülle die Lücken so, dass Toms Aussage widerlegt wird:

Letzten Sonntag war es [ ] und [ ] .

sonnig	in allen Teichen schwammen Biber
nicht sonnig	im Teich mit dem Wasserfall schwamm kein Biber
	Biber Michael schwamm durch alle Teiche
	Biber Michael schwamm gar nicht

So ist es richtig:

Tom macht eine Aussage über alle sonnigen Tage. Um Toms Aussage zu widerlegen, muss man einen sonnigen Tag finden, bei dem der zweite Teil seiner Aussage nicht stimmt. Da Tom keine Aussage über nicht sonnige Tage macht, kann man ihn mit einer Aussage über nicht sonnige Tage nicht widerlegen. Die erste Lücke muss also mit „sonnig“ gefüllt werden.

Der zweite Teil von Toms Aussage wiederum ist eine Aussage über alle Teiche. Um diese Aussage zu widerlegen, brauchen wir eine Aussage über (mindestens) einen Teich, in dem der zweite Teil dieser Aussage – nämlich dass ein Biber im Teich schwimmt – nicht wahr ist. Eine solche Aussage ist „im Teich mit dem Wasserfall schwamm kein Biber“; die zweite Lücke muss also mit dieser Aussage gefüllt werden. Mit den anderen Möglichkeiten, die zweite Lücke zu füllen, kann Toms Aussage nicht widerlegt werden:

„Letzten Sonntag war es sonnig und in allen Teichen schwammen Biber.“: Diese Aussage stimmt mit Toms Aussage überein.

„Letzten Sonntag war es sonnig und Biber Michael schwamm durch alle Teiche.“: Irgendwann letzten Sonntag ist also in jedem Teich mindestens ein Biber geschwommen (nämlich Biber Michael), so wie Tom es für sonnige Tage behauptet. Auch diese Aussage stimmt mit Toms Aussage überein.

„Letzten Sonntag war es sonnig und Biber Michael schwamm gar nicht.“: Diese Aussage reicht nicht, um Toms Aussage zu widerlegen. Auch wenn Biber Michael gar nicht geschwommen ist, könnten in jedem Teich andere Biber geschwommen sein.

Das ist Informatik!
In dieser Biberaufgabe diskutiert Kim den Wahrheitsgehalt von Toms Aussage. Tom macht eine Aussage über alle sonnigen Tage, und für diese wiederum eine Aussage über alle Teiche. Die gesamte Aussage kann nur falsch sein, wenn es mindestens einen sonnigen Tag gibt, an dem in mindestens einem Teich kein Biber schwamm. Daher reicht ein einziges Gegenbeispiel aus, um seine Aussage zu widerlegen.

Auch die klassische Logik beschäftigt sich mit der Wahrheit von Aussagen: Aussagen können *wahr* oder *falsch* sein. *Logische Operatoren* wie „und“, „oder“ und „nicht“ werden verwendet, um (einfachere) Aussagen zu komplexeren Aussagen zu verbinden. *Quantoren* wie „alle“ und „mindestens ein“ werden verwendet, um Aussagen über Mengen von Objekten zu machen.

Der Umgang mit Logik ist fundamental in der Informatik. Computersysteme sind mit logischen Bausteinen wie „und“, „oder“ und „nicht“ aufgebaut, die statt „wahr“ und „falsch“ die Werte 1 und 0 (elektrische Spannung bzw. keine Spannung) verwenden. Ebenso verwenden Programmiersprachen logische Ausdrücke, die in Anweisungen wie

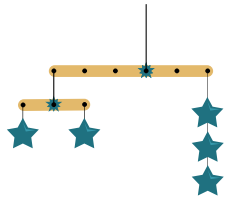
```
IF it is summer AND the sun is shining, THEN turn on the air conditioning
```

verwendet werden.

Stern-Mobiles

Stern-Mobiles sind Gebilde aus Fäden, Stäben und Sternen. An einem Faden kann eine Anzahl von Sternen hängen; oder ein Stab, an dessen beiden Enden jeweils wieder ein Stern-Mobile hängt.

Dies ist ein einfaches Stern-Mobile:



Mit Zahlen und Klammern kann man es so beschreiben:

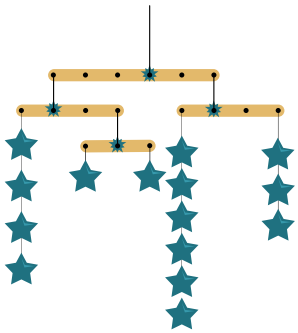
`(-3 (-1 1) (1 1)) (2 3)`

Die Zahlen geben jeweils an:
- entweder den Abstand eines Stab-Endes zum Faden, an dem der Stab hängt,
- oder eine Anzahl an Sternen.

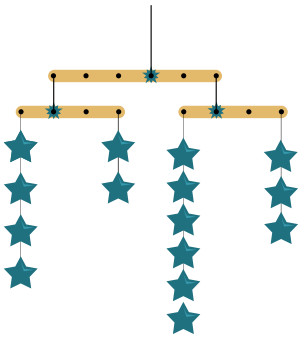
Die Klammern geben die Struktur des Stern-Mobiles an.

Welches der folgenden Stern-Mobiles kann man so beschreiben:
`(-3 (-1 4) (2 (-1 1) (1 1))) (2 (-1 6) (2 3))`

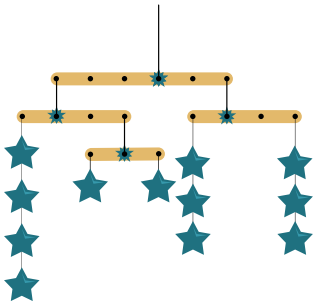
A)



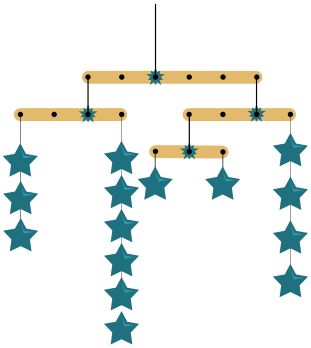
B)



C)



D)

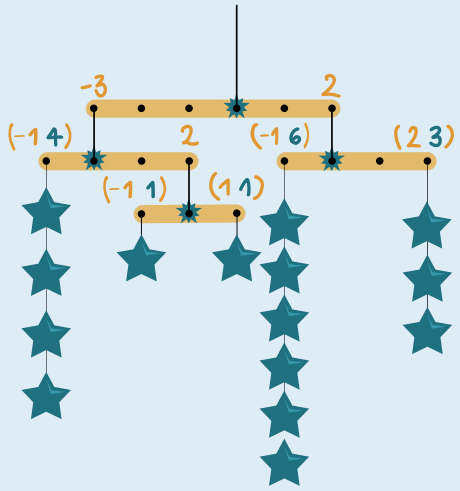


Antwort A ist richtig:

Aus dem Beispiel und seiner Beschreibung kann man Folgendes erkennen:
- Ein Stern-Mobile mit Stab wird durch zwei Klammerpaare beschrieben: `(A1 M1) (A2 M2)`. Die Klammerpaare sind von links nach rechts so angeordnet wie die Teil-Mobiles, die am Stab des Stern-Mobiles hängen.
- `A1` und `A2` geben jeweils einen Abstand zum Aufhängefaden des Stabs an; an diesen Stellen hängen die Stern-Mobiles `M1` und `M2`.
- Ein einfaches Stern-Mobile (ein Faden mit Sternen) wird durch eine Zahl S beschrieben. Das ist die Anzahl der Sterne, die an dem Faden hängt.

Konkret gilt für das Beispiel:
- Die beiden Klammerpaare `(-3 …)` `(2 3)` beschreiben ein Stern-Mobile mit Stab.
- Das rechte Klammerpaar `(2 3)` bedeutet, dass mit Abstand 2 nach rechts vom Aufhängefaden ein einfaches Stern-Mobile mit 3 Sternen hängt.
- Das linke Klammerpaar `(-3 (-1 1) (1 1))` bedeutet, dass mit Abstand 3 vom Aufhängefaden (und zwar nach links, deshalb -3) ein Stern-Mobile mit Stab hängt; an dem Stab hängen wiederum zwei einfache Stern-Mobiles mit jeweils einem Stern.

Der Ausdruck aus der Frage `(-3 (-1 4) (2 (-1 1) (1 1))) (2 (-1 6) (2 3))` beschreibt also ein Stern-Mobile mit Stab und zwei Teil-Mobiles:
- Die Teil-Mobiles am obersten Stab hängen links mit Abstand 3 und rechts mit Abstand 2.
- Am Stab des linken Teil-Mobiles hängt links (Abstand 1) ein Faden mit 4 Sternen und rechts ein Stern-Mobile mit Stab, mit jeweils einem Stern links und rechts (jeweils mit Abstand 1).
- Am Stab des rechten Teil-Mobiles hängt links (Abstand 1) ein Faden mit 6 Sternen und rechts (Abstand 2) ein Faden mit 3 Sternen.

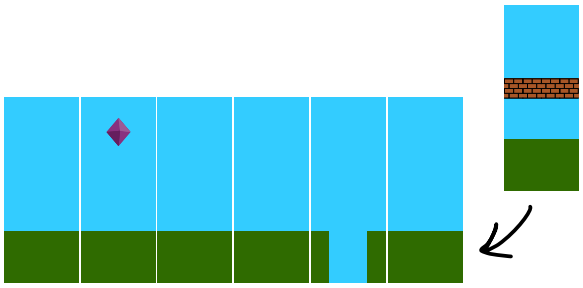


Das ist genau das Mobile aus Antwort A.
Beim Mobile aus Antwort B hat das linke Teil-Mobile selbst kein Teil-Mobile.
Beim Mobile aus Antwort C gibt es keinen Faden mit 6 Sternen.
Beim Mobile aus Antwort D ist alles spiegelverkehrt.

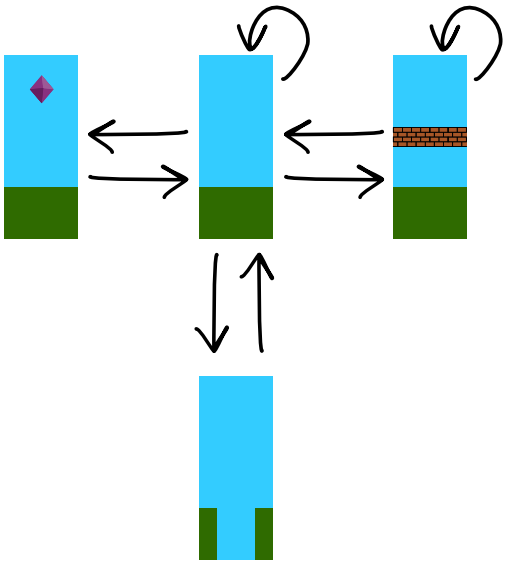
Das ist Informatik!
Ein Stern-Mobile hat eine interessante Struktur: An jedem Stab hängen nämlich stets wieder (etwas kleinere) Stern-Mobiles. Dabei ist ein Faden mit einem oder mehreren Sternen auch ein (ganz einfaches) Stern-Mobile. Ein Stern-Mobile ist also:
- entweder ein Faden mit einigen Sternen oder ...
- ... ein Faden mit einem Stab, an dessen Enden je ein Stern-Mobile hängt.

Diese *Definition* benennt Stern-Mobiles als mögliche Bestandteile eines Stern-Mobiles. Strukturen, die kleinere Exemplare des gleichen Strukturtyps als Bestandteil haben, nennt man *rekursiv.* In der Computerprogrammierung können rekursive Strukturen mit sehr kurzen Programmen bearbeitet werden. Die Programme haben dabei eine ähnlich rekursive Programmstruktur wie die rekursive Definition der Strukturen: Sie bearbeiten entweder den Basisfall (bei den Stern-Mobiles: Faden mit Sternen) oder rufen sich selbst auf, um Teilstrukturen zu bearbeiten, die nicht dem Basisfall entsprechen.

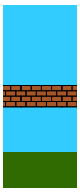
Superbebras



Im Computerspiel „Superbebras“ besteht der Hintergrund aus einer Folge von Kacheln. Der Computer fügt ständig eine neue Kachel auf der rechten Seite der Folge hinzu und entfernt gleichzeitig eine Kachel auf der linken Seite. Das sieht dann wie eine Bewegung aus.



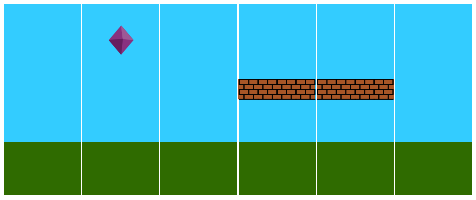
Um die nächste neue Kachel auszuwählen, verwendet der Computer das Diagramm auf der rechten Seite. Im Diagramm sucht er die vorige neue Kachel (auf der rechten Seite) und die Pfeile, die davon ausgehen. Dann wählt er zufällig eine Kachel aus, auf die einer dieser Pfeile zeigt.

Zum Beispiel kann der Computer nach der Kachel  entweder wieder die Kachel  oder die Kachel  auswählen.

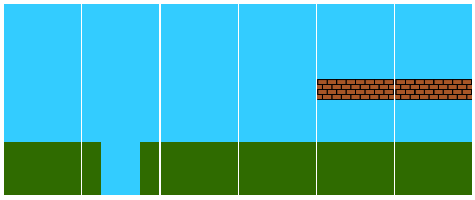
In einem Superbebras-Hintergrund muss also die Reihenfolge der Kacheln zu obigem Diagramm passen.

Eines dieser Bilder ist KEIN Superbebras-Hintergrund. Welches?

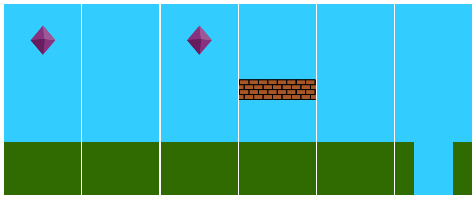
A)



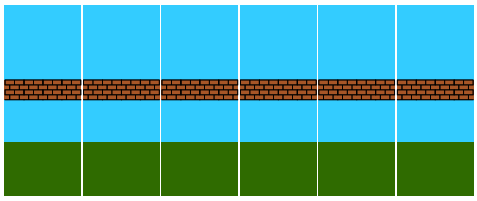
B)



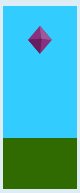
C)

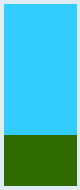


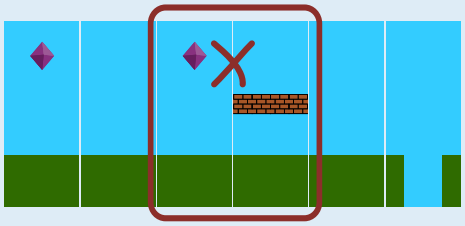
D)



Antwort C ist richtig:

Das Bild aus Antwort C passt nicht zum Diagramm. Auf eine Kachel mit einem Diamanten 

kann gemäß Diagramm nur eine  Kachel folgen.

Das ist im Bild aus Antwort C nicht der Fall: 

Alle anderen Bilder passen zum Diagramm.

Es gibt mehrere Möglichkeiten festzustellen, ob ein Bild zum Diagramm passt oder nicht. Ein einfaches Vorgehen besteht darin, die Kacheln des Bildes von links nach rechts durchzugehen und immer zu schauen, ob es für zwei nebeneinander liegende Kacheln einen Pfeil im Diagramm gibt.

Man kann aber auch gezielter vorgehen: Man geht die Kacheln im Diagramm nacheinander durch, beginnend mit den Kacheln, von denen nur ein Pfeil ausgeht. Ein Pfeil ist eine Bedingung: A ⟶ B bedeutet „nach Kachel A muss Kachel B kommen“. Dann prüft man, ob das Bild diese Bedingung erfüllt.

Das ist Informatik!
Einige Computerspiele – wie endlose Läuferspiele – haben einen Hintergrund, der sich horizontal bewegt und die Illusion erzeugt, dass sich der Spieler durch eine Fantasiewelt bewegt. Manchmal ist der Hintergrund kein konstantes Bild, sondern wird automatisch vom Computer erzeugt. Dies nennt man *prozedurale Generierung.* Üblicherweise werden kleinere Objekte kombiniert, um eine große Vielfalt an Hintergründen zu erzeugen.

Die Kombination der Objekte kann jedoch nicht völlig zufällig erfolgen. Sie muss bestimmten Regeln folgen, um Situationen zu vermeiden, mit denen der Spieler nicht zurechtkommt. In der Aufgabe werden diese Regeln durch ein Diagramm aus Kacheln und Pfeilen dargestellt. Solche Diagramme, in denen Objekte durch Pfeile verbunden sind, kennt die Informatik als *gerichtete Graphen* (genauer gesagt: als deren grafische Darstellung). Die Objekte werden dann als Knoten und die Pfeile als gerichtete Kanten bezeichnet. Gerichtete Graphen werden in der Informatik zur Modellierung von Zusammenhängen zwischen Objekten verwendet.

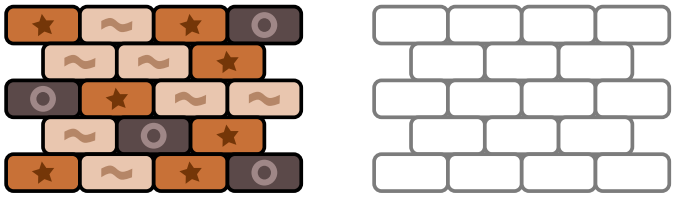
Endliche Automaten gehen noch einen Schritt weiter: Sie haben Zustände, einen Startzustand, Endzustände und Regeln, wie man von einem Zustand zum nächsten wechselt. Das ist ähnlich wie bei einem gerichteten Graphen, nur dass es für spezielle Zustände besondere Knoten gibt. Auch ein solcher Automat kann grafisch beschrieben werden, nämlich durch sein Zustandsübergangsdiagramm. Endliche Automaten werden in der Informatik zur Modellierung einfacher Abläufe oder zur Beschreibung einfach konstruierbarer Folgen eingesetzt – wie den Kachelfolgen in dieser Biberaufgabe.

Umstapeln

Bob hat einige Ziegelsteine ordentlich gestapelt. Nun soll er den Stapel woanders neu aufbauen. Damit auch der neue Stapel ordentlich wird, arbeitet Bob nach dieser Methode, Stein für Stein, bis alle Steine umgestapelt sind:

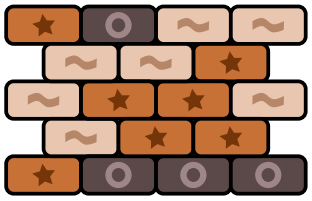
- Er nimmt irgendeinen Stein, auf dem kein anderer Stein liegt, vom Stapel und
- legt ihn auf den neuen Stapel, entweder auf den Boden oder auf einen oder zwei andere Steine, keinesfalls aber unter einen anderen Stein.

Links siehst du Bobs ersten Stapel. Rechts ist der Platz für den neuen Stapel.

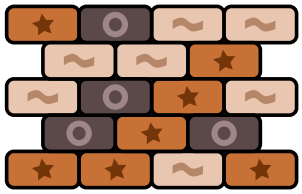


Wie kann Bobs neuer Stapel aussehen, wenn er fertig ist?

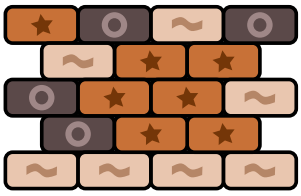
A)



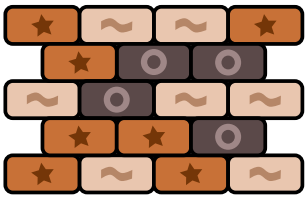
B)



C)

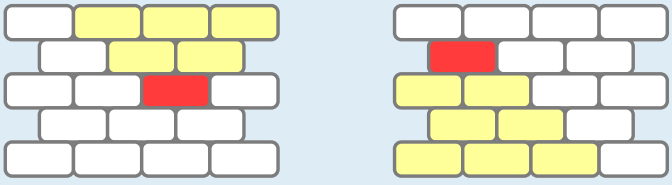


D)



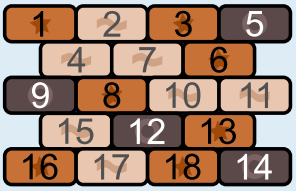
Antwort D ist richtig:

Zunächst stellen wir eine grundsätzliche Überlegung an. Im Bild links ist der rote Stein die Spitze eines umgedrehten Dreiecks, das ansonsten aus den sechs gelben Steinen besteht. Das sind genau die Steine, die vor dem roten Stein umgestapelt werden müssen. Zu jedem Stein im ersten Stapel gehört so ein „vorher weg"-Dreieck. Im Bild rechts sind die gelben Steine wiederum die, die vor dem roten Stein auf den neuen Stapel gelegt werden müssen. Dies ist das „vorher drauf"-Dreieck, das es für jeden Stein gibt.



Nummeriert man die Steine in der Reihenfolge, in der sie umgestapelt werden, gelten diese beiden Umstapel-Bedingungen: Jeder Stein im ersten Stapel hat eine größere Nummer als der anderen Steine in seinem „vorher weg“-Dreieck. Analog hat jeder Stein im neuen Stapel eine größere Nummer als die anderen Steine in seinem „vorher drauf“-Dreieck.

Das Bild unten zeigt, wie der Stapel von **Antwort D** aus dem ersten Stapel durch Umstapeln nach Bobs Methode erzeugt werden kann. Die Steine sind in der Reihenfolge nummeriert, in der sie umgestapelt werden, und die Umstapel-Bedingungen gelten:



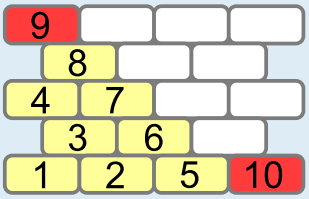
erster Stapel

neuer Stapel aus Antwort D

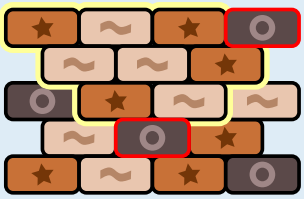
Nun bleibt zu zeigen, dass die Stapel der Antworten A bis C nicht durch Umstapeln nach Bobs Methode erzeugt werden können.

Am leichtesten ist das für **Antwort C** zu erkennen: In der unteren Reihe liegen vier  . Ein zweiter  kann frühestens als dritter Stein umgestapelt werden. Unter den ersten beiden Steinen, die umgestapelt werden, ist also einer kein  , muss aber in die untere Reihe des neuen Stapels gelegt werden. Es ist also nicht möglich, dass im neuen Stapel vier  in der unteren Reihe liegen.

Im Stapel aus **Antwort A** liegen drei  in der unteren Reihe. Ein Stein in der unteren Reihe des neuen Stapels muss spätestens als 10. Stein umgestapelt worden sein (Bild links). Wenn man aus dem ersten Stapel den  aus der vierten Reihe von oben nehmen will, muss Bob vorher alle neun Steine von dessen „vorher-weg“-Dreiecks umstapeln (Bild rechts). Dieser  kann also frühestens als 10. Stein umgestapelt werden und ist dann der zweite  im neuen Stapel. Ein dritter  kann deshalb frühestens als 11. Stein umgestapelt werden. Es ist also nicht möglich, dass im neuen Stapel drei  in der unteren Reihe liegen.



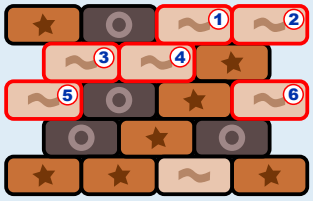
Stapel-Reihenfolge neuer Stapel



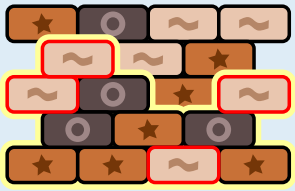
„vorher-weg“-Dreieck erster Stapel

Nun betrachten wir noch den Stapel von **Antwort B:** Der vierte  , der auf diesen Stapel gelegt wird, kann dort an den im Bild unten links markierten sechs Positionen liegen. Wenn man alle diese Positionen betrachtet und jeweils alle Möglichkeiten, welche drei  vorher auf den Stapel gelegt werden können, stellt man fest: Es müssen mindestens 3  vor dem vierten  auf den Stapel gelegt werden, nämlich u. a. wenn der vierte  auf Position 3 gelegt wurde. (Das Bild unten Mitte zeigt, welche Steine in diesem Fall vor dem vierten  auf den neuen Stapel gelegt werden

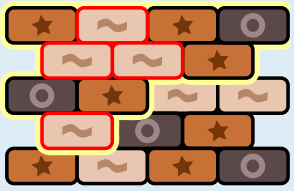
müssen.) Es ist aber nicht möglich, aus Bobs erstem Stapel so viele vor dem vierten umzustapeln: In Bobs erstem Stapel ist der Stein links in der zweiten Reihe von unten derjenige, der als vierter umgestapelt werden kann, so dass vorher möglichst viele andere Steine umgestapelt werden können. Dieser wird dann vor den anderen Steinen in seiner Reihe und vor denen in der unteren Reihe umgestapelt. Vor ihm können also höchstens 2 umgestapelt werden (siehe Bild rechts). Das reicht also nicht.



mögliche Positionen vierter im neuen Stapel



vierter auf Position 3, mit vorher-drauf-Steinen

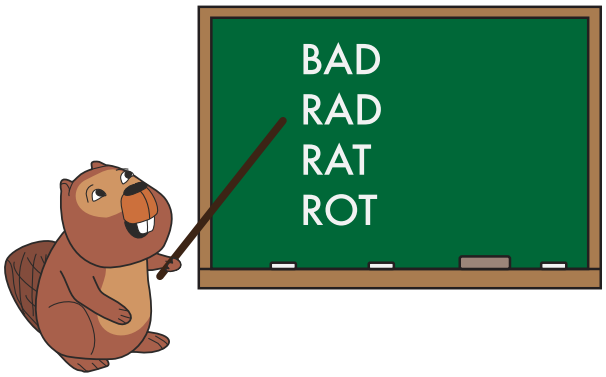


maximale Menge der vorher-weg-Steine

Das ist Informatik!
Weil Bob so ordentlich nach seiner Methode arbeitet, kann er die Steine nicht in beliebiger Reihenfolge umstapeln. Aus der Lage der Steine in seinem ersten Stapel ergeben sich viele Abhängigkeiten zwischen jeweils zwei Steinen bezüglich der Reihenfolge. Zum Beispiel gilt für den Stein links in der zweiten Reihe von oben, dass sowohl der Stein ganz links in der obersten Reihe als auch der Stein rechts daneben vorher umgestapelt werden muss. Der Stein selbst muss wiederum vor sowohl dem Stein ganz links in der dritten Reihe von oben als auch dem Stein rechts daneben umgestapelt werden. Insgesamt entsteht so eine „Umstapel-Ordnung“. Sie legt für viele Stein-Paare fest, dass der eine Stein vor dem anderen umgestapelt werden muss – aber nicht für alle möglichen Stein-Paare.

In der Informatik muss man häufig mit Ordnungen umgehen, die genau so unvollständig sind wie die Umstapel-Ordnung in dieser Biberaufgabe. Wenn die zeitliche Abfolge von Arbeitsschritten in einem Produktionsprozess per Computer geplant wird, ist für einige Paare von Arbeitsschritten bekannt, welcher vor dem anderen ausgeführt werden muss – aber nicht für alle Paare. Wenn ein neues Softwarepaket auf einem Computer installiert wird, müssen vorher meist einige andere Pakete installiert werden, und für einige Paare dieser Pakete ist eine Installationsreihenfolge bekannt – aber nicht für alle. Dennoch muss es am Ende eine Reihenfolge geben, in der alles schön nacheinander passiert. Es gibt einen erfreulicherweise einfachen Algorithmus, der eine solche *topologische Sortierung* auf Grundlage einer unvollständigen Ordnung berechnen kann.

Wortketten



Herr Bibers Klasse lernt Lesen. Dabei arbeiten sie mit Wortketten: Das nächste Wort in einer Kette unterscheidet sich von seinem Vorgänger an genau einer Position. Hier ist ein Beispiel:
BAD — RAD — RAT — ROT

Herr Biber schreibt nun diese neun Wörter an die Tafel:
ROT, RAD, RAT, BAD, BAR, RAR, TAT, BAU, FAD.

Die Klasse soll alle Wörter benutzen und daraus drei Wortketten mit je drei Wörtern bilden. Herr Biber warnt: „Je nachdem, welche Wortkette man zuerst bildet, schafft man es nicht mehr, zwei weitere Wortketten zu bilden und alle Wörter zu benutzen."

Welche Wortkette soll die Klasse auf keinen Fall zuerst bilden?

A) ROT – RAD – TAT

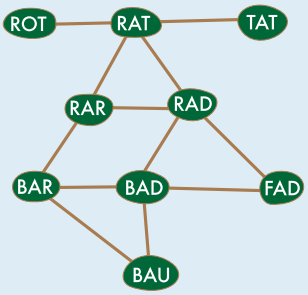
B) BAU – BAR – RAR

C) BAR – BAD – FAD

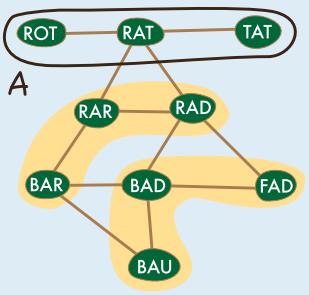
D) BAD – FAD – RAD

Antwort C ist richtig:

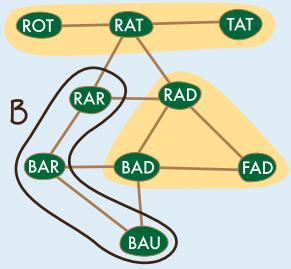
Wir können diese Aufgabe mithilfe einer Zeichnung lösen: Zunächst schreiben wir alle Wörter auf. Dann verbinden wir mit einer Linie je zwei Wörter, die sich an genau einer Position unterscheiden.



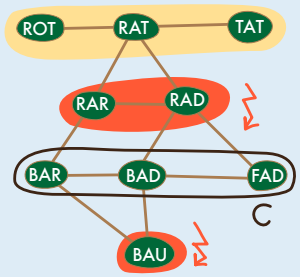
In Antwort A wird die Wortkette ROT — RAT — TAT gebildet. Aus den übrigen Wörtern lassen sich zum Beispiel die gelb markierten Wortketten bilden. Alle Wörter werden genau einmal benutzt. Alle Wortketten bestehen aus drei Wörtern.



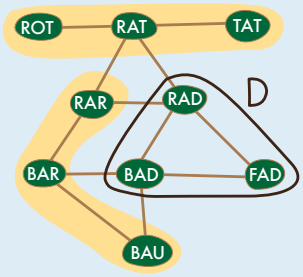
Die Wortkette in Antwort B ist BAU — BAR — RAR. Aus den übrigen Wörtern können wir die gelb markierten Wortketten bilden. Alle Wörter werden genau einmal benutzt. Alle Wortketten bestehen aus drei Wörtern.



In Antwort C wird die Wortkette BAR — BAD — FAD gebildet. Es fällt auf, dass BAU isoliert wird. Wir können mit den übrigen Wörtern die gelb markierte Wortkette bilden, aber die zwei rot markierten Wörter und das Wort BAU können nicht verbunden werden. Sie bleiben übrig.



Die Wortkette in Antwort D ist BAD — FAD — RAD. Aus den übrigen Wörtern können wir die gelb markierten Wortketten bilden. Alle Wörter werden genau einmal benutzt. Alle Wortketten bestehen aus drei Wörtern.



Die Wortkette in Antwort C darf die Klasse also auf keinen Fall zuerst bilden.

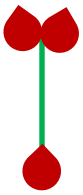
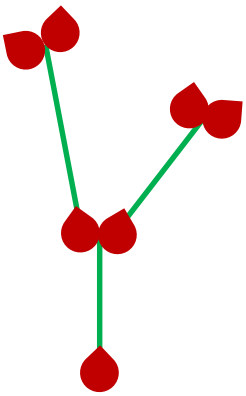
Das ist Informatik!
In den Zeichnungen, mit welchen wir den Lösungsweg erläutert haben, ist jedes Wort ein (grün hinterlegter) *Knoten.* Und eine Kante zwischen zwei Knoten (als Linie gezeichnet) zeigt, dass sich diese zwei Wörter an genau einer Position unterscheiden. Diese zwei Knoten sind dann *Nachbarn.* In der Wortkette BAD — RAD — RAT — ROT sind zum Beispiel BAD und RAD Nachbarn, aber BAD und RAT sind es nicht. Konstruktionen aus Knoten und Kanten sind in der Informatik allgemein als *Graphen* bekannt und werden eingesetzt, um mit den Kanten Zusammenhänge zwischen den durch die Knoten dargestellten Objekten darzustellen.

Die Wortkette BAD — RAD — RAT umfasst drei Knoten und zwei Kanten. Zwischen BAD und RAT besteht ein *Weg,* der die entsprechenden Knoten über den gemeinsamen Nachbarn RAD miteinander verbindet. Von jedem der neun Wörter aus können wir alle anderen Wörter erreichen. Dies kann direkt (d.h. wenn die entsprechenden Knoten Nachbarn sind) sein oder mithilfe eines längeren Wegs, der sich über mehrere Nachbarn erstreckt. Um diese Biberaufgabe zu lösen, werden drei solcher Wege benötigt. Dabei muss jeder Knoten genau einmal verwendet werdet werden, so dass, wenn man nur die Wege und keine weiteren Kanten betrachtet, der Graph am Ende in drei *Komponenten* zerlegt ist.

Graphen sind ein beliebtes Werkzeug der Informatikerinnen und Informatiker, um komplexe Sachverhalte zu modellieren. Programme können mithilfe von Graphen viele Aufgaben lösen. Navigationssysteme sind zum Beispiel Programme, die mithilfe sehr großer Graphen Autos schrittweise vom Startort bis zum Zielort führen. Mit jedem Schritt fährt das Auto über eine Kante von einem Knoten zu einem Nachbarn.

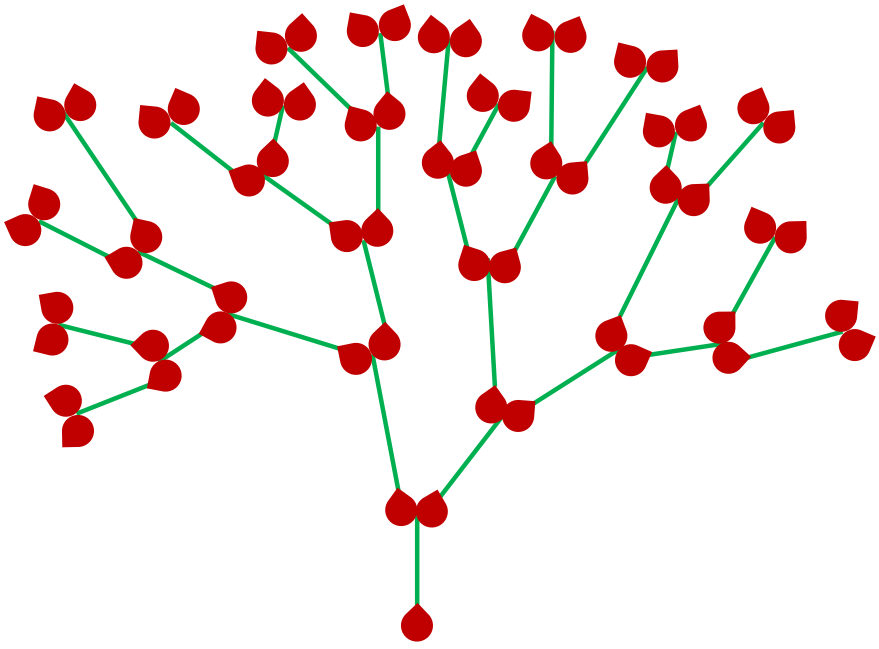
Wunderblume

Bei Sonnenaufgang wächst aus jeder neuen Knospe der Wunderblume ein Stiel. Den ganzen Tag lang wächst der Stiel weiter. Bei Sonnenuntergang hört der Stiel auf zu wachsen, und es kommen zwei neue Knospen heraus. So geht es Tag für Tag weiter, und die Wunderblume wird immer prächtiger.

Neue Knospe vor Sonnenaufgang	Tag 1 nach Sonnenuntergang	Tag 2 nach Sonnenuntergang
		

Wie viele Tage ist diese Wunderblume gewachsen?

A)	B)	C)	D)
5 Tage	11 Tage	16 Tage	32 Tage



Antwort A ist richtig:

Die Wunderblume hat Knospen, Stiele – und auch Äste. Ein Ast besteht aus den Stielen, an denen entlang man von der ersten Knospe zu einer äußeren Knospe geht, aus der noch kein Stiel gewachsen ist. Am Ende von Tag 1 gibt es nur einen Ast, und der ist 1 Stiel lang. Am Ende von Tag zwei gibt es schon zwei Äste, und die sind 2 Stiele lang. Durch das Wachstum der Wunderblume werden die Äste an jedem Tag um einen Stiel länger.

Man muss also nur einem Ast folgen, z. B. dem ganz rechts, und zählen, wie viele Stiele der Ast lang ist. Der Ast ist 5 Stiele lang, und so weiß man, dass die Wunderblume 5 Tage gewachsen ist.

Die anderen Antworten sind also falsch. Die Zahlen 11, 16 und 32 passen trotzdem zu dieser in 5 Tagen gewachsenen Wunderblume:
- Jeder Ast hat 11 Knospen.
- Die äußeren Knospen sind aus 16 Stielen herausgewachsen.
- Diese Wunderblume hat 32 äußere Knospen.

Das ist Informatik!
Nach nur fünf Tagen ist die Wunderblume schon sehr groß geworden. Sie hat viele Stiele und noch mehr schöne rote Knospen bekommen. Alleine von den äußeren Knospen gibt es nach 5 Tagen schon 32 Stück. Andersherum kann man feststellen, dass jede der 32 äußeren Knospen an einem Ast gewachsen ist, der nur 5 Stiele lang ist.

Diese Tabelle zeigt für die ersten 10 Tage, wie viele äußeren Knospen eine Wunderblume nach wie vielen Tagen hat und wie lang die Äste dann sind:

Tage	1	2	3	4	5	6	7	8	9	10
Länge der Äste	1	2	3	4	5	6	7	8	9	10
äußere Knospen	2	4	8	16	32	64	128	256	512	1024

Schon nach 10 Tagen hat die Wunderblume sehr viele Knospen. Die Äste sind aber noch recht kurz: Um nach 10 Tagen von der ersten Knospe aus eine der über tausend äußeren Knospen zu erreichen, muss man an nur 10 Stielen entlang gehen.

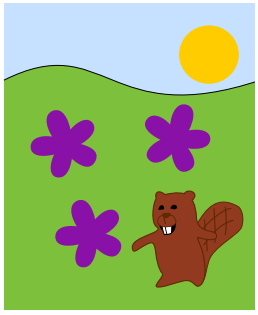
Damit Computerprogramme auch mit vielen Daten schnell umgehen können, wurden in der Informatik Datenstrukturen entwickelt, die der Wunderblume in dieser Biberaufgabe ähneln. In diesen Strukturen kann man sehr viele Daten speichern, aber dennoch jedes einzelne Datenelement vom Ausgangspunkt der Struktur aus in wenigen Schritten erreichen. Wenn eine solche Struktur genauso schön ausgewogen ist wie die Wunderblume, kann man in 5 Schritten $2 \times 2 \times 2 \times 2 \times 2 = 32$ Datenelemente erreichen, in 10 Schritten $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 1.024$ und in nur 20 Schritten mehr als eine Million. Die Anzahl der benötigten Schritte wächst also sehr viel langsamer als die Menge der damit erreichbaren Datenelemente. In der Informatik sagt man, dass die Anzahl der Schritte „logarithmisch“ wächst, im Verhältnis zur Anzahl der Datenelemente. Umgekehrt wächst die Anzahl der Datenelemente „exponentiell“ im Verhältnis zur Anzahl der Schritte.

Zeichnung

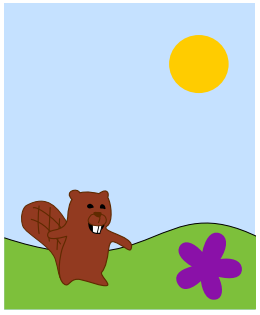
Bea malt ein Bild. Sie hat fünf Flaschen mit Farbe. Am Anfang sind alle Flaschen voll. Für große Flächen auf dem Bild braucht sie mehr Farbe als für kleine Flächen. Als Bea fertig ist, enthalten die Flaschen noch so viel Farbe:

Welches Bild hat Bea gemalt?

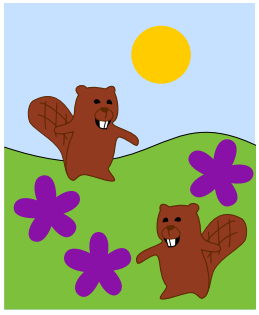
A)



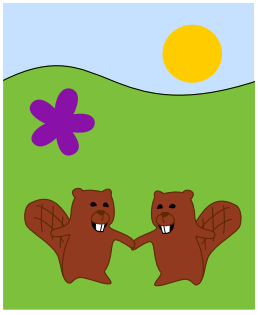
B)



C)



D)



Antwort D ist richtig:

Um die richtige Antwort zu finden, kannst du die Farbflächen auf dem Bild mit dem Inhalt der Farbflaschen vergleichen. Du erkennst an den Farbflaschen, dass am meisten Grün verwendet wurde und am zweitmeisten Blau. Das passt zu den Bildern A, C und D. Auf diesen Bildern ist die grüne Fläche am größten und die blaue Fläche am zweitgrößten. Bild B kannst du ausschließen, da dort die blaue Fläche viel größer ist als die grüne.

An den Farbflaschen kannst du außerdem erkennen, dass Bea genauso viel Gelb verwendet hat wie Violett. Auf den Bildern A und C ist die violette Gesamtfläche (3 Blüten) etwa 3-mal so groß wie die gelbe Fläche (Sonne). Nur in Bild D sind die gelben und violetten Flächen (Sonne und Blüte) etwa gleich groß. Auch die Verwendung der braunen Farbe passt zu Bild D: Bea hat genauso viel Braun verwendet wie Blau, und die braune Gesamtfläche (2 Biber) ist etwa gleich groß wie die blaue Fläche. Bea kann also nur Bild D gemalt haben.

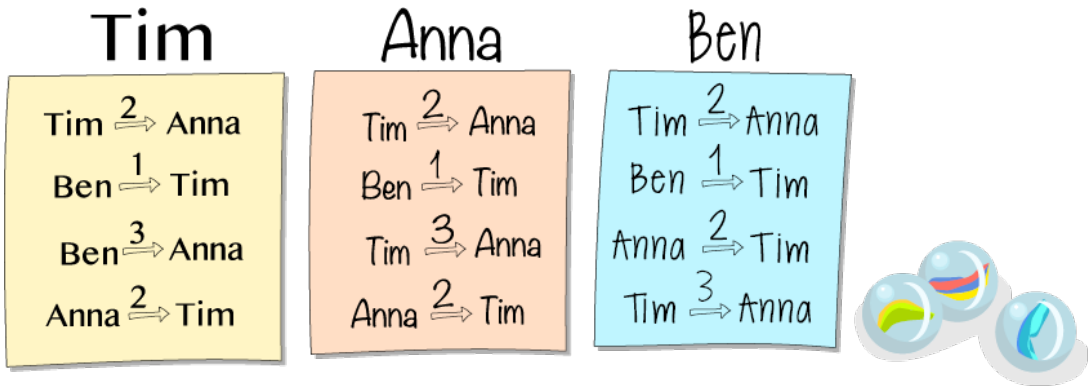
Das ist Informatik!
In dieser Biberaufgabe hast du das richtige Bild herausgefunden, indem du die Verwendung der Farben untersucht hast. Dabei hast du dir z. B. folgende Fragen gestellt: Welche Farben kommen vor? Welche Farbe wird am meisten verwendet? Welchen Farben werden nur wenig verwendet? Wird eine Farbe mehr verwendet als eine andere Farbe?

Zur Beantwortung dieser Fragen musstest du auf dem Bild verschiedene Flächen unterscheiden, die Größe von Flächen abschätzen, Flächenwerte miteinander vergleichen und Entscheidungen treffen. Solche Untersuchungen können auch automatisch von Computerprogrammen durchgeführt werden. Computerprogramme können dir helfen, auf deinem Handy gespeicherte Fotos zu finden. Durch eine Farbuntersuchung kann ein Computerprogramm Landschaftsbilder, die oben viel Blau und unten viel Grün enthalten, von Portraitfotos unterscheiden. Durch Farbuntersuchungen von Satellitenfotos können Computer ermitteln, wie dicht eine Gegend bebaut ist, oder ob in einem Naturschutzgebiet illegal Wald gerodet wurde.

In der Informatik gibt es einen wichtigen Bereich, der sich mit der automatischen Untersuchung von Bildern befasst: *Computer Vision.*

Zettel

Anna, Ben und Tim leihen sich ab und zu Murmeln untereinander aus. Zur Sicherheit trägt jeder auf einem eigenen Zettel ein, wer wem wie viele Murmeln ausgeliehen hat.



Nach einer Woche vergleichen die drei Freunde ihre Zettel. Sie haben den Verdacht, dass ein Fehler passiert ist: Auf genau einem Zettel scheint ein Eintrag falsch zu sein.

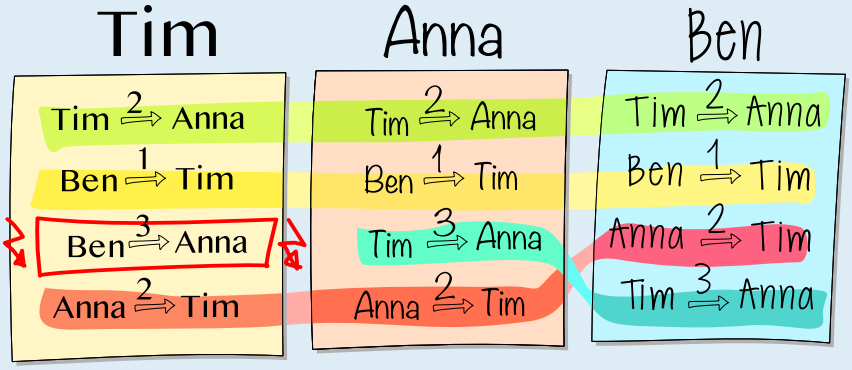
Wenn ja, wer hat den Fehler gemacht?

A)	B)	C)	D)
Anna	**Ben**	**Tim**	**Niemand hat einen Fehler gemacht**

Antwort C ist richtig:

Wenn niemand einen Fehler gemacht hätte, müssten auf jedem Zettel die gleichen Einträge stehen. Drei Einträge sind auf allen Zetteln gleich. Sie sind im Bild durch eine grüne, gelbe und rote Linie markiert. Nur bei einem Eintrag gibt es einen Unterschied:
Auf zwei Zetteln findet man den Eintrag
Tim $\overset{3}{\Rightarrow}$ Anna, und nur auf Tims Zettel steht Ben $\overset{3}{\Rightarrow}$ Anna

Wenn man annimmt, dass auf genau einem Zettel ein Eintrag falsch ist, muss Tim den Fehler gemacht haben.



Das ist Informatik!

In der Sprache der Informatik würde man sagen, dass die Kinder in unserer Geschichte sich entschieden haben, ihre Daten (die Murmel-Leih-Einträge) nicht in einer gemeinsamen Datenbank (Zettel) zu verwalten, sondern je eine eigene Datenbank zu führen. Deshalb gibt es nun mehrere, getrennte Datenbanken, die dennoch – so die Erwartung – denselben Inhalt abbilden. Aber warum diese, scheinbar überflüssige, mehrfache Datenhaltung? Sie ist sinnvoll, wenn man auf ein zentrales System verzichten möchte, etwa weil es ausfallen könnte oder nicht verlässlich oder nicht vertrauenswürdig arbeiten könnte. Insgesamt sehen wir hier ein Beispiel für ein *Peer-to-Peer-Netzwerk:* Jedes Kind ist ein *Peer,* und jedes Peer hat die gleichen Rechte und Pflichten (nämlich die korrekte Speicherung der Ausleih-Daten) wie alle anderen. Zum Abgleich der Daten können die Kinder miteinander kommunizieren, so wie das auch Computer in einem Netzwerk tun können. In einem Peer-to-Peer-Netz funktioniert der Abgleich nur, wenn die meisten Peers ehrlich sind. Bei Unstimmigkeiten stimmen sie darüber ab, welche der Datenbanken korrekt ist; in der Aufgabe würden Anna und Ben diese Abstimmung gewinnen.

Die Gleichstellung der Peers und die Verteilung der Daten gehören zu den Grundgedanken der *Blockchain-Technologie.* Diese geht noch einen Schritt weiter, indem sie einen *Konsensalgorithmus* implementiert. Dieser wäre nur dann problematisch, wenn ein böswilliger Akteur mehr Rechenleistung kontrollierte als alle anderen Teilnehmer zusammen. Das ist de facto unmöglich, wenn das Netzwerk groß genug ist. In dieser Biberaufgabe ist Blockchain kein Thema; sie zeigt aber dennoch, wie das Vertrauen in einer nicht von vornherein vertrauenswürdigen Umgebung gestärkt werden kann, indem die Kontrolle unter allen Teilnehmern verteilt wird.

Du hast das ganze Biberheft gelesen.
Zielgruppe: Informatik-interessiert.
Du solltest programmieren lernen:
jwinf.de
bwinf.de/python

Für Fortgeschrittene:
algo.bwinf.de

Hast du Lust auf mehr Informatik?

Mach mit beim Jugendwettbewerb Informatik!

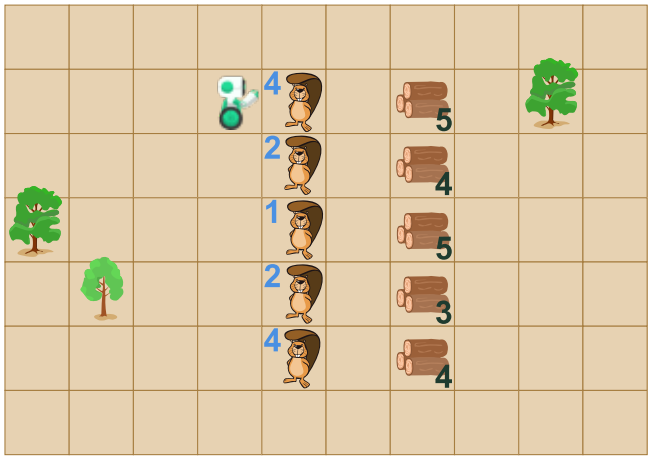
Informatik-Knobeln mit dem Biber macht Spaß, klar. Aber neben genauem Denken gehört auch das Programmieren zur Informatik. Entdecke und teste deine Algorithmen-Skills mit den Füchsen vom Jugendwettbewerb Informatik!

Im Jugendwettbewerb wird mit gut verständlichen grafischen Bausteinen programmiert. Alles, was du für die Teilnahme brauchst, kannst du auf der Trainingsplattform jwinf.de lernen. In 2025 startet die 1. Runde am 17. März. Hier ist eine Beispielaufgabe mit Lösung. Der Biber ist sicher, dass du sie verstehst.

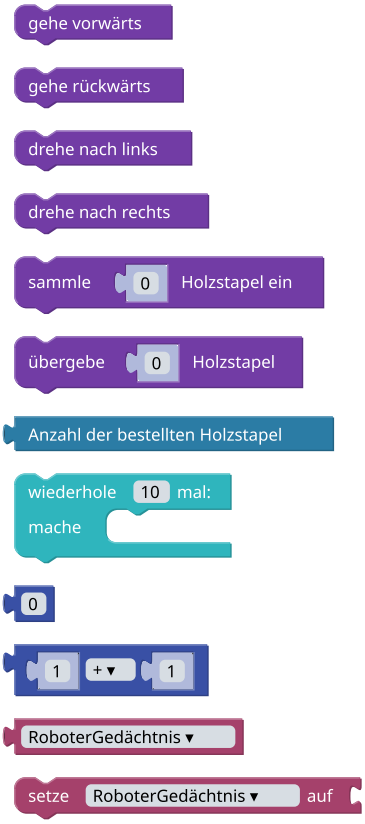
Fütterungszeit

Die Holzbestellung des Bibers, angegeben durch die Zahl neben seinem Kopf, liegt auf dem Weg verteilt. Der Roboter soll alle Holzstapel einsammeln und dem Biber bringen.

Hinweis: Der Roboter soll in jeder Zeile nur genau so viele Holzstapel einsammeln, wie auch bestellt worden. Du darfst nur 30 Bausteine verwenden.

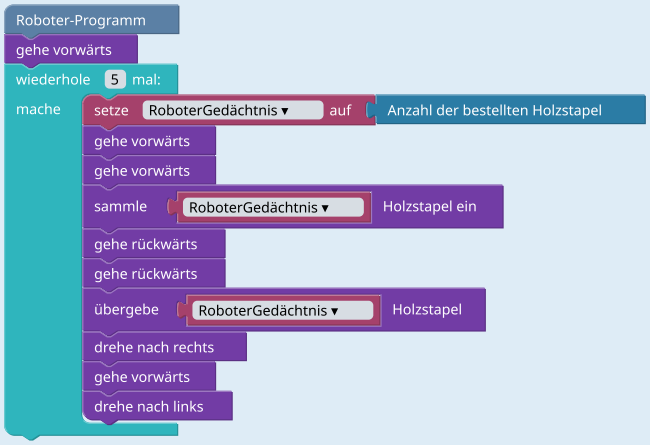


verfügbare Bausteine:

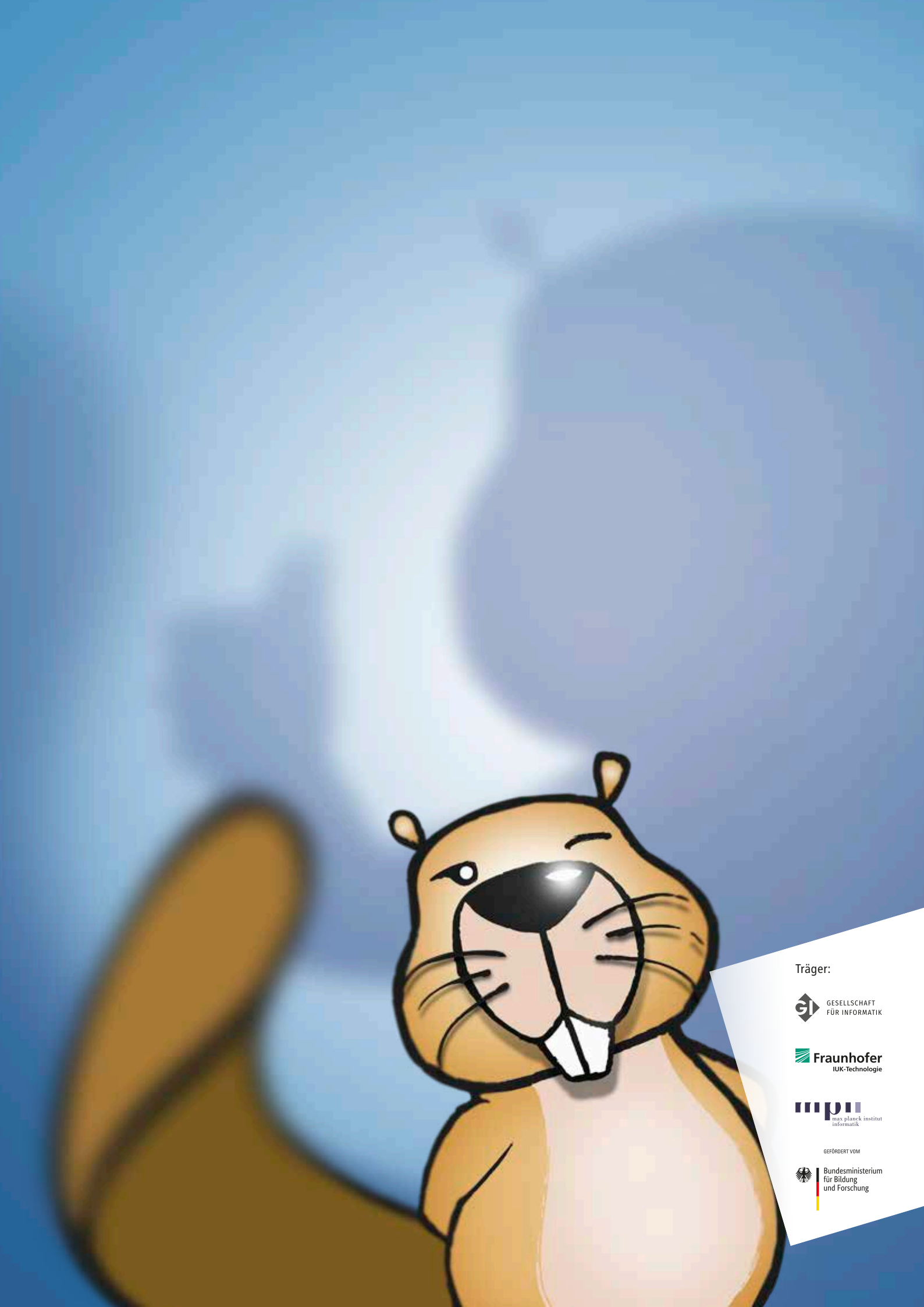


Programmiere den Roboter:

So ist es richtig:



Zur Online-Aufgabe:
jwinf.de/task/1688



Träger:

GESELLSCHAFT FÜR INFORMATIK

Fraunhofer IUK-Technologie

max planck institut informatik

GEFÖRDERT VOM

Bundesministerium für Bildung und Forschung