BUNDES WEIT INFORMATIK NACHWUCHS FÖRDERN

www.bwinf.de

Informatik-Biber

AUFGABEN 2019

Der Wettbewerb zum digitalen Denken.

bwinf.de/biber

Herausgeber Wolfgang Pohl, BWINF

Der Aufgabenausschuss Informatik-Biber 2019

Hannes Endreß, Universität Leipzig
Ulrich Kiesmüller, Simon-Marius-Gymnasium Gunzenhausen
Wolfgang Pohl, BWINF
Kirsten Schlüter, Bayerisches Staatsministerium für Unterricht und Kultus
Michael Weigend, Holzkamp-Gesamtschule Witten

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

Wilfried Baumann, Österreichische Computer Gesellschaft
Robert Czechowski, BWINF
Christian Datzko, Wirtschaftsgymnasium und Wirtschaftsmittelschule Basel
Susanne Datzko, freischaffende Graphikerin, ETH Zürich
Gerald Futschek, Technische Universität Wien
Martin Guggisberg, Pädagogische Hochschule FHNW, SVIA*
Juraj Hromkovic, ETH Zürich, SVIA
Anna Husmann, BWINF
Ivana Kosirová, ETH Zürich, SVIA
Regula Lacher, ETH Zürich, SVIA
Lucio Negrini, Scuola universitaria professionale della Svizzera italiana (SUPSI)
Gabriel Parriaux, Haute École Pédagogique Vaud, SVIA
Jean-Philippe Pellet, Haute École Pédagogique Vaud, SVIA
Katharina Resch-Schobel, Österreichische Computer Gesellschaft
Florentina Voboril, Technische Universität Wien

* Schweiz. Verein für Informatik in der Ausbildung

Der Informatik-Biber

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

Einleitung

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

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

Der Informatik-Biber findet jährlich im November statt. An der 13. Austragung im Jahr 2019 beteiligten sich 2.308 Schulen und andere Bildungseinrichtungen mit 401.737 Schülerinnen und Schülern. Die Möglichkeit, auch in Zweierteams zu arbeiten, wurde gern genutzt.

Die Online-Teilnahme am Informatik-Biber 2019 war mit Desktops, Laptops und Tablets möglich. Weniger als die Hälfte der Antworteingaben waren multiple-choice. Verschiedene andere Interaktionsformen machten die Bearbeitung abwechslungsreich. In diesem Biberheft ist die Dynamik der Aufgabenbearbeitung nicht vorführbar. Darum geben Handlungstipps in den Aufgabenstellungen und Bilder von Lösungssituationen davon eine Vorstellung. Der Umgang mit dem Wettbewerbssystem selbst konnte in den Wochen vor der Austragung online geübt werden.

Der Informatik-Biber 2019 wurde in fünf Altersgruppen durchgeführt. In den Klassenstufen 3 bis 4 waren innerhalb von 30 Minuten 9 Aufgaben zu lösen, jeweils drei in den Schwierigkeitsstufen leicht, mittel und schwer. In den Klassenstufen 5 bis 6 waren innerhalb von 35 Minuten 12 Aufgaben zu lösen, jeweils vier in den Schwierigkeitsstufen leicht, mittel und schwer. In den Klassenstufen 7 bis 8, 9 bis 10 und 11 bis 13 waren innerhalb von 40 Minuten 15 Aufgaben zu lösen, jeweils fünf in den Schwierigkeitsstufen leicht, mittel und schwer.

Die 35 Aufgaben des Informatik-Biber 2019 sind auf Seite 6 gelistet, nach ungefähr steigender Schwierigkeit und mit einer informatischen Klassifikation ihres Aufgabenthemas. Ab Seite 7 folgen die Aufgaben nach ihrem Titel alphabetisch sortiert. Im Kopf sind die zugeordneten Altersgruppen und Schwierigkeitsgrade vermerkt. Eine kleine Flagge gibt an, aus welchem Bebras-Land die Idee zu dieser Aufgabe stammt. Der Kasten am Aufgabenende enthält Erläuterungen zu den Lösungen und Lösungswegen sowie eine kurze Darstellung des Aufgabenthemas hinsichtlich seiner Relevanz in der Informatik.

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

Wir laden die Schülerinnen und Schüler ein, auch 2020 wieder beim Informatik-Biber mitzumachen, und zwar in der Zeit vom 9. bis 20. November. Weitere Informationen werden über die Website bwinf.de und per E-Mail an die Koordinatorinnen und Koordinatoren bekannt gegeben.

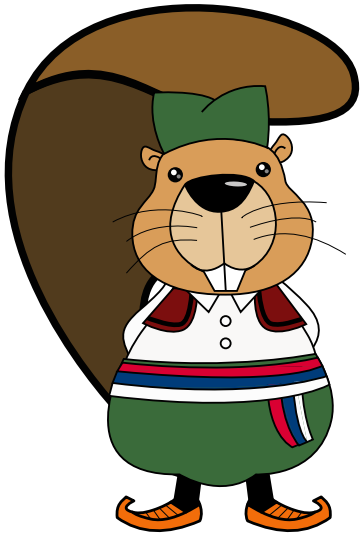
Bebras: International Challenge on Informatics and Computational Thinking

Der belgische Biber

Der deutsche Informatik-Biber ist Partner der internationalen Initiative Bebras. 2004 fand in Litauen der erste Bebras Challenge statt. 2006 traten Estland, die Niederlande und Polen der Initiative bei, und auch Deutschland veranstaltete im Jahr der Informatik als „EI: Spiel blitz!“ einen ersten Biber-Testlauf. Seitdem kamen viele Bebras-Länder hinzu. Zum Drucktermin sind es weltweit 61, und weitere Länderteilnahmen sind in Planung. Insgesamt hatte der Bebras Challenge 2019 international etwa drei Millionen Teilnehmerinnen und Teilnehmer.

Die Bebras-Community erarbeitet jedes Jahr auf einem internationalen Workshop anhand von Vorschlägen der Länder eine größere Auswahl möglicher Aufgabenideen. Die Ideen zu den 35 Aufgaben des Informatik-Biber 2019 stammen aus 18 Ländern: Belgien, Deutschland, Indien, Irland, Japan, Kanada, Südkorea, Litauen, Österreich, Pakistan, Russland, Schweiz, Slowakei, Slowenien, Thailand, Tschechien, Ukraine und Vietnam.

Der koreanische Biber



Der serbische Biber

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

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

Bundesweite Informatikwettbewerbe

Bundesweite Informatikwettbewerbe

Informatik-Biber

Jugendwettbewerb Informatik

Bundeswettbewerb Informatik

Informatik-Olympiade

Bei jungen Menschen das Interesse für Informatik wecken, Begabungen entdecken und fördern: das ist das Ziel der Bundesweiten Informatikwettbewerbe (BWINF), an denen im Jahr 2019 etwa 420.000 junge Menschen teilnahmen. Der Informatik-Biber ist das BWINF-Einstiegsformat; außerdem werden noch drei weitere Wettbewerbe 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 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 Wettbewerbsplattform 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.

Internationale Informatik-Olympiade
Die Jüngeren unter den BwInf-Finalisten und einige ausgewählte Teilnehmende der zweiten Runde 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 Twitter und Instagram, das Informatik-Jugendportal Einstieg Informatik mit seiner Community und die BWINF-Website. Die mehr als 35 Jahrgänge von Bundeswettbewerbs-Teilnehmenden bilden ein wachsendes Netzwerk, vor allem im BwInf Alumni und Freunde e.V. Nach der 1. Runde lernen sich viele Teilnehmende bei Informatik-Workshops von Hochschulen und Unternehmen kennen.

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

Aufgabenliste

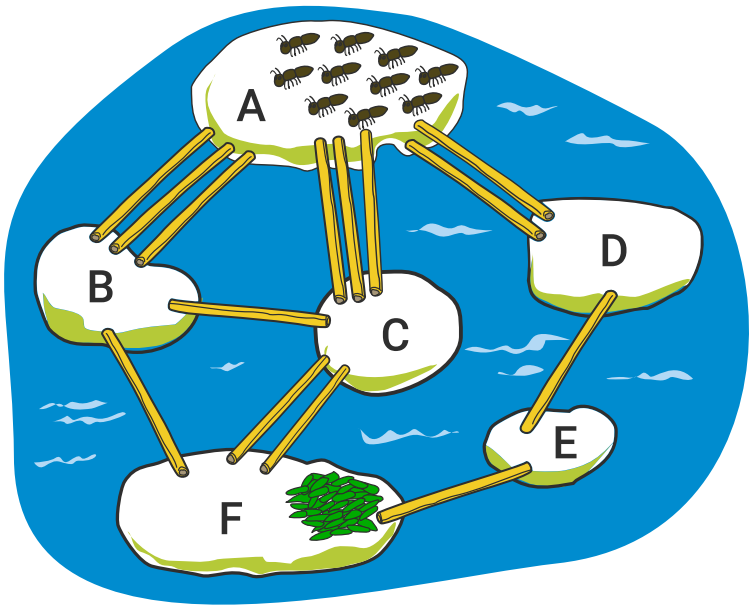
Das sind die 35 Aufgaben des Informatik-Biber 2019, grob geordnet nach steigender Schwierigkeit und gelistet mit einer Klassifikation ihres informatischen Inhalts.

Titel	Thema	Seite
Zum Strand	Programmieren, bedingte Anweisung	65
Lutscher	Datenstrukturen, Queue/Schlange	29
Bibertaler	Kodierung, Binärzahlen	12
Schneemann-Hüte	Digitales Denken, Abstraktion, Informationsverlust	44
Kratzbilder	Anwendungen, Computergrafik, Ebenen	22
Rauchzeichen	Kodierung, Fehlerkorrektur	36
Lenas Nachricht	Kodierung, Kryptographie	28
Superstar	Modellierung, Graphen, Soziale Netzwerke	51
Aufräumen	Robotik, Steuerung, Sensorik	10
Teller-Ordnung	Algorithmik, Sortieren, Einfügen, Effizienz	52
Stempel	Programmieren, Software Engineering, Reverse Engineering	48
Obstspieße	Algorithmik, Heiratsproblem, Matching	32
Julias Turm	Modellierung, Zustandsübergangsdiagramm	19
Raumfahrt	Modellierung, Automaten, DEA	37
Leckeres Holz	Algorithmik, Set Cover, NP-vollständig	26
Rangoli	Digitales Denken, Dekomposition	34
Türme	Algorithmik, Sortieren, Bubble-Sort, Effizienz	56
Zeichenroboter	Programmieren, Grundbausteine, Turing-vollständig	63
Formenspiel	Theoretische Informatik, Formale Sprachen, Wortproblem	13
Kanalsystem	Algorithmik, Sortieren, Sortiernetze	20
Schiebeparkplatz	Robotik, autonomes Fahren	43
Überwacht	Anwendungen, Bildverarbeitung, Überwachung	57
Sägewerk	Programmieren, Reaktives Programmieren	39
Ameisen im Fluss	Algorithmik, Graphen, Flussalgorithmen	7
Mondzauber	Kodierung, Binäre Logik, XOR	30
Wackelig	Rekursion (Strukturen, Programmierung)	61
Stern-Mobiles	Rekursion (Strukturen, Programmierung)	49
Insel-Falle	Algorithmik, Flussdiagramm, Schadprogramme	17
Kugelbahn	Kodierung, Binäres Zählen, Hardware	24
Schatzkarte	Modellierung, Abstraktion, Graphen	41
Hände schütteln	Algorithmik, Laufzeit, O-Notation	15
Video speichern	Kodierung, Videokompression	59
Trainingstour	Algorithmik, Graphen, Hamiltonpfad, NP-vollständig	54
Schneepflug-Roboter	Algorithmik, Graphen, Steinerbäume, NP-vollständig	46
Anproben	Algorithmik, Binäre Suche, Sortieren	8

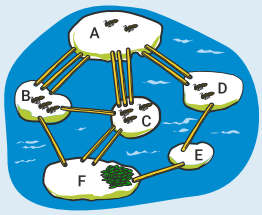
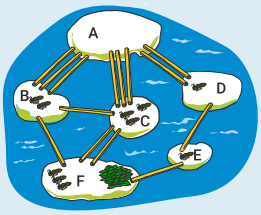
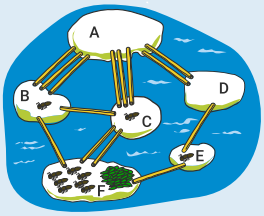
Ameisen im Fluss

Zehn Ameisen sind auf Stein A. Sie wollen zum Stein F, dort gibt es Futter. Die Ameisen können über die Strohhalme zu den anderen Steinen laufen. Aber jeder Strohhalm kann höchstens eine Ameise gleichzeitig tragen. Eine Ameise braucht eine Minute, um von einem Stein über einen Strohhalm zum nächsten Stein zu laufen.

Wie viele Ameisen können nach drei Minuten höchstens beim Futter auf Stein F sein?



7 ist die richtige Antwort:
Die Bilder zeigen, wie viele Ameisen nach ein, zwei bzw. drei Minuten höchstens über die Strohhalme gelaufen sein können:

nach einer Minute	nach zwei Minuten	nach drei Minuten
		

Das ist Informatik!
Die Frage in dieser Biberaufgabe lässt sich auch als Auftrag formulieren: Bestimme die größte Zahl an Ameisen, die nach 3 Minuten beim Futter sein können. Es gibt viele Möglichkeiten, wie die Ameisen über die Strohhalme laufen können. Zur Bearbeitung dieses Auftrags werden sie allein nach einem Merkmal oder Kriterium unterschieden, nämlich nach der Anzahl der Ameisen, die nach 3 Minuten beim Futter sind. Bezüglich dieses Kriteriums wird die beste, also optimale Möglichkeit gesucht. Der Auftrag entpuppt sich also als Optimierungsaufgabe, mit dem schon genannten Optimierungskriterium.
Die Informatik beschäftigt sich intensiv mit vielerlei verschiedenen Optimierungsaufgaben. Eine Optimierungsaufgabe ist, den maximalen Fluss von Mengen (zum Beispiel Güter, Verkehrsteilnehmer, Wasser, …) durch Netzwerke (zum Beispiel Eisenbahn- oder Straßenverbindungen, Leitungssysteme, …). Dabei können die einzelnen Verbindungen der Netzwerke (Schienenstrecken, Straßen, Leitungen, …) unterschiedliche Kapazitäten haben. Auch die Strohhalme in dieser Biberaufgabe bilden ein Fluss-Netzwerk, mit einzelnen Verbindungen zwischen zwei Steinen und der Anzahl der Strohhalme als Kapazitäten. Die Menge, deren Fluss zu optimieren ist, sind die Ameisen. Du hast hier also ein ganz spezielles Fluss-Problem gelöst.
https://de.wikipedia.org/wiki/Flüsse_und_Schnitte_in_Netzwerken

Anproben

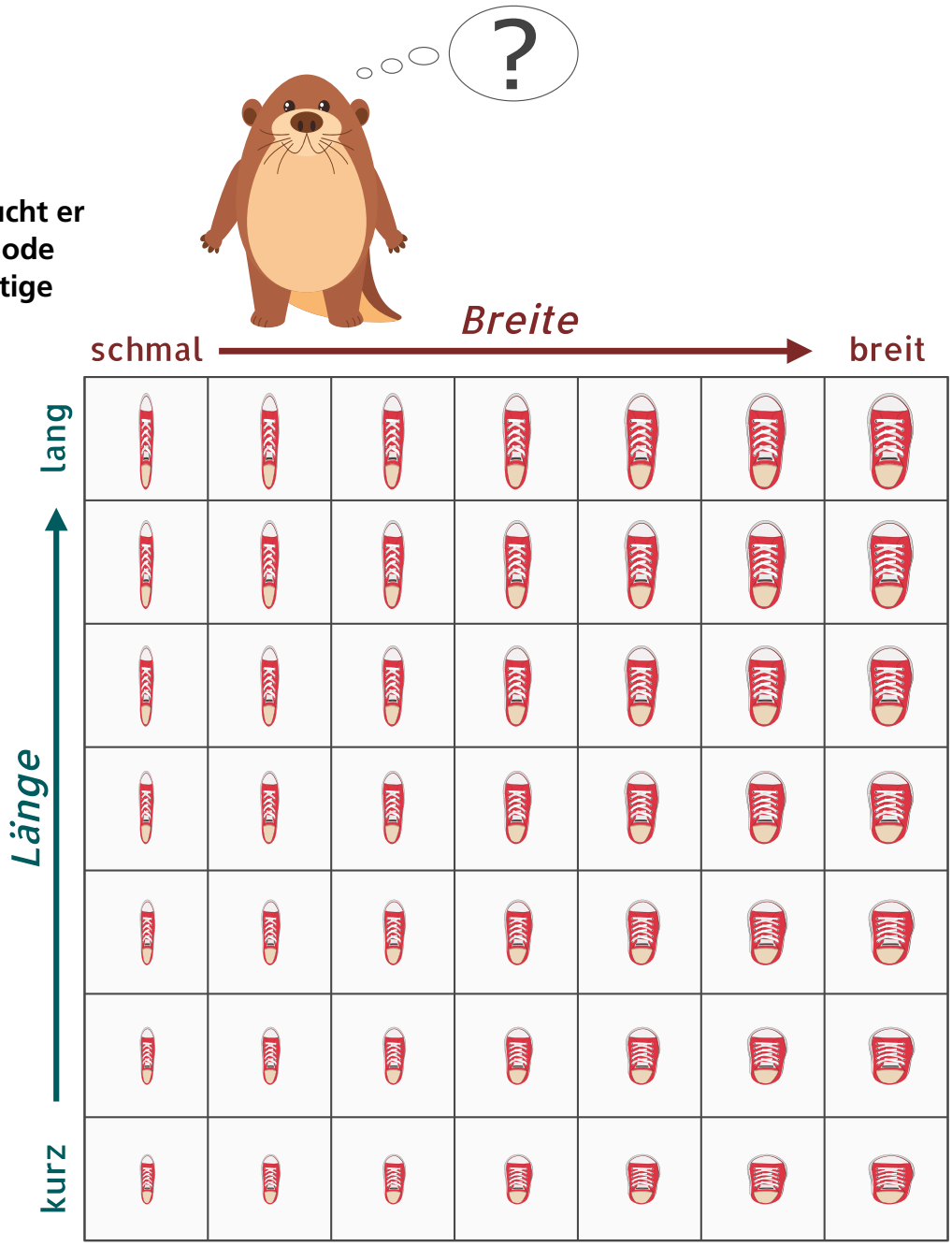
Der Biber braucht neue Schuhe. Im Geschäft gibt es seine Lieblingsschuhe in sieben Längen und sieben Breiten. Schuhe in allen 49 Größen sind im Regal, nach Länge und Breite sortiert.

Weil der Biber die richtige Größe nicht weiß, muss er sie durch Anprobieren herausfinden. Bei jeder Anprobe merkt der Biber, ob der Schuh passt oder ob er einen kürzeren, längeren, schmaleren oder breiteren Schuh braucht. Länge und Breite müssen stimmen!

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

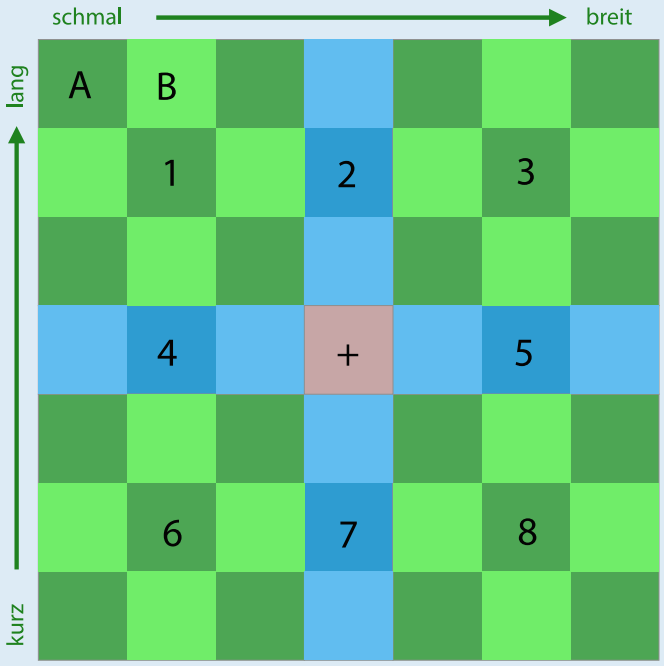
Doch dem schlauen Biber ist die schnellste Methode eingefallen, die richtige Größe in jedem Fall nach möglichst wenigen Anproben zu wissen.

Wie viele Anproben braucht er mit der schnellsten Methode höchstens, bis er die richtige Größe weiß?



2 ist die richtige Antwort:
Natürlich kann der Biber Glück haben und direkt bei der ersten Anprobe den Schuh in der richtigen Größe erwischen. Aber auf Glück will er sich nicht verlassen und geht nach dieser Methode vor:
Zuerst probiert er den Schuh in der Mitte an (Position + im Bild). Bei der Anprobe prüft er Länge und Breite des Schuhs.

- Wenn Länge und Breite stimmen, hat der den Schuh mit der richtigen Größe gefunden.
- Wenn der Schuh zu kurz und zu breit ist, ist der passende Schuh in Bereich 1.
- Wenn der Schuh zu kurz ist aber die richtige Breite hat, ist der passende Schuh in Bereich 2.
- Wenn der Schuh zu kurz und zu schmal ist, ist der passende Schuh in Bereich 3.
- Und so weiter.



Nehmen wir an, der Schuh mit der richtigen Größe ist in Bereich 1. Der Biber wählt für die zweite Anprobe den Schuh in der Mitte von Bereich 1.

Nun gibt es wieder mehrere Möglichkeiten:
Wenn der Schuh passt, hat er die richtige Größe gefunden.

- Wenn der Schuh immer noch zu kurz und zu breit ist, weiß der Biber, dass der Schuh an Position A die richtige Größe hat.
- Wenn der Schuh zu kurz ist, aber die passende Breite hat, weiß der Biber, dass der Schuh an Position B die richtige Größe hat.
- Und so weiter.

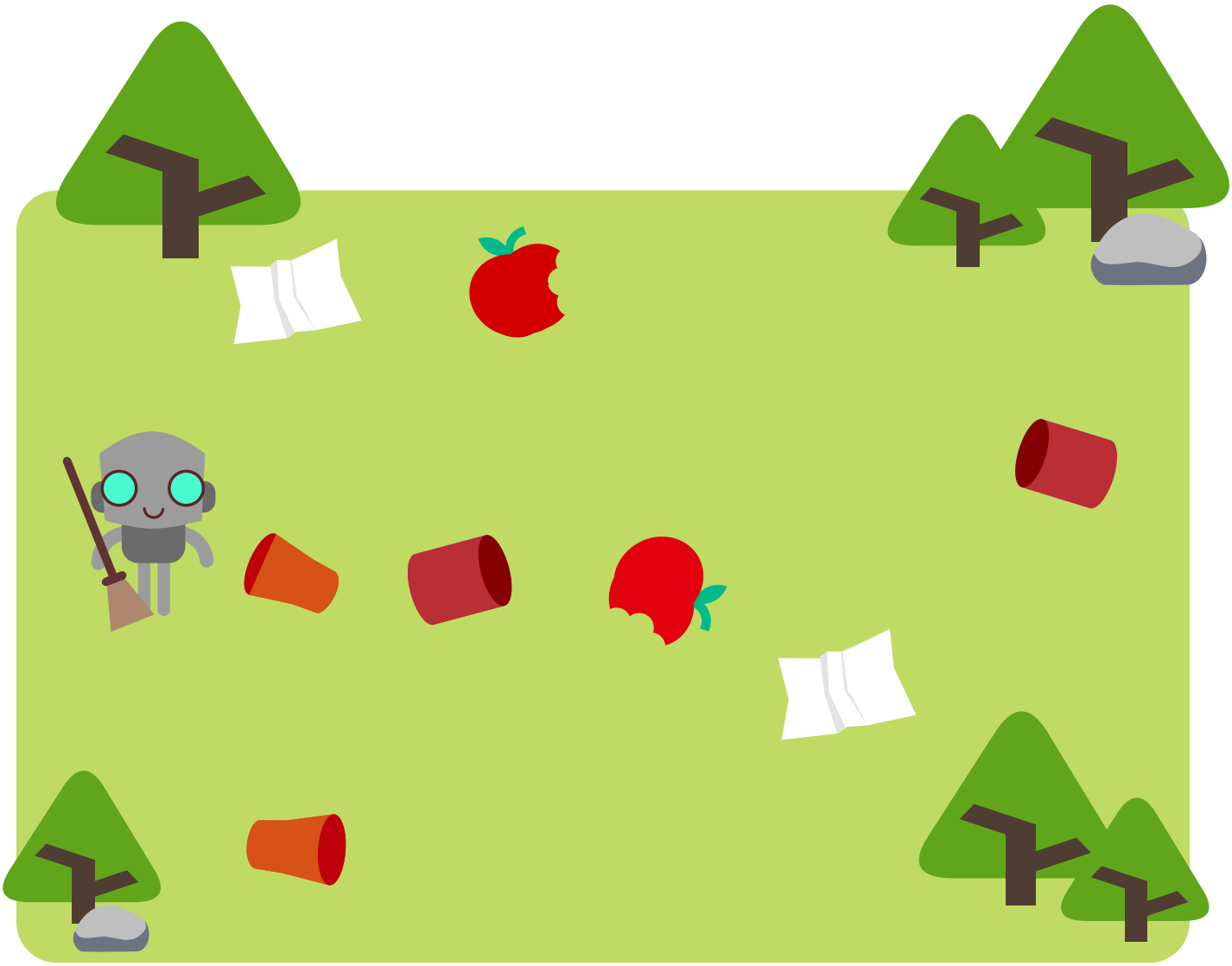
Weil in jedem nummerierten Bereich das mittlere Regalfach in jeder Richtung nur ein Nachbarfach hat, sind keine weiteren Anproben notwendig. Der Biber braucht also in jedem Fall höchstens zwei Anproben, um die richtige Größe zu wissen.

Das ist Informatik!
Die Methode, die der Biber bei der Anprobe anwendet, heißt in der Informatik binäre Suche. Der Begriff *binär* kommt vom lateinischen Wort *bis* (zweimal). Bei der binären Suche nach einem Objekt in einer Folge sortierter Objekte wird deren mittleres Objekt mit dem gesuchten verglichen. Dann weiß man, in welcher Hälfte der Folge sich das gesuchte Objekt befindet und durchsucht diese Hälfte wieder binär. In jedem Schritt wird die Folge also in zwei Teile geteilt – deshalb „binär“. Auf diese Weise kommt man sehr schnell beim gesuchten Objekt an. Bei 1.000 Objekten werden etwa 10 Suchschritte benötigt, bei 1.000.000 Objekten etwa 20. Allgemein kann man sagen: Bei n Objekten werden etwa $\log n$ Schritte benötigt; die Funktion log ist der „Zweier-Logarithmus“.
In dieser Biberaufgabe ist die Suchfolge, nämlich die Schuhe im Regal, in zwei Dimensionen (Länge und Breite) sortiert. Deshalb kann der Biber die binäre Suche gleich auf beide Dimensionen anwenden. Dann teilt sich die Suchmenge in einem Schritt nicht in 2, sondern gleich in 8 Teile auf.
Weil die binäre Suche so schnell ist, wird sie in Computerprogrammen häufig für die Suche in sortierten Daten verwendet.

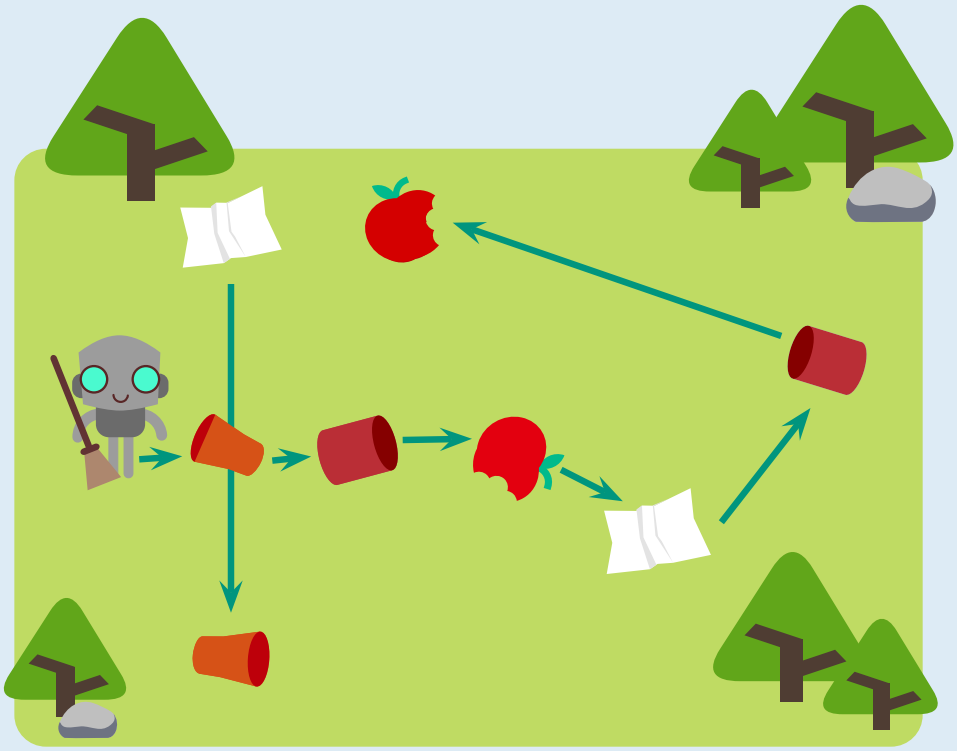
Aufräumen

Nach dem Freilichtkonzert im Park sammelt der Roboter den Müll auf, den das Publikum auf dem Rasen zurückgelassen hat.
Der Roboter geht zu dem nächstgelegenen Ding, das auf dem Rasen herumliegt, und sammelt es auf. Dann geht er zu dem Ding, das jetzt am nächsten liegt, und sammelt es auf. So macht der Roboter immer weiter, bis er allen Müll aufgesammelt hat.

Welches Ding sammelt der Roboter als letztes auf?



So ist es richtig:



Das Bild zeigt den Weg, den der Roboter beim Müllsammeln geht. Als letztes sammelt er den Becher unten links auf.

Das ist Informatik!

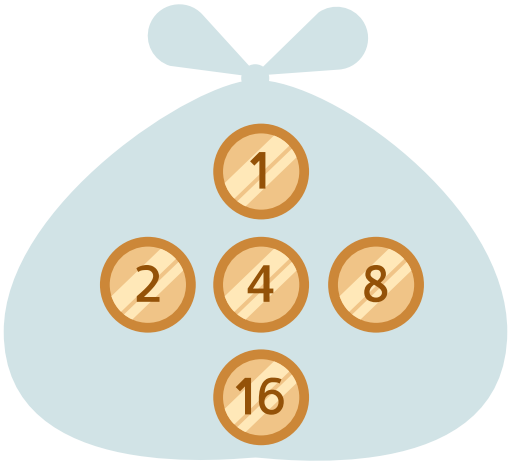
Das Verhalten des Roboters wird durch ein Computerprogramm gesteuert. Das Programm wird auf einem Computer ausgeführt, der in den Roboter eingebaut ist. Ein Computerprogramm besteht aus Befehlen, die in einer Sprache geschrieben sind, die der Computer versteht. Ein Roboter hat Kameras und Sensoren, mit denen er Dinge in seiner Umgebung wahrnehmen und die Entfernung zu diesen Dingen berechnen kann. Das Programm für den Computer des Roboters verarbeitet die Daten der Sensoren und ermöglicht, dass der Roboter sich autonom bewegen kann. Das heißt, er muss nicht von einem Menschen ferngesteuert werden.

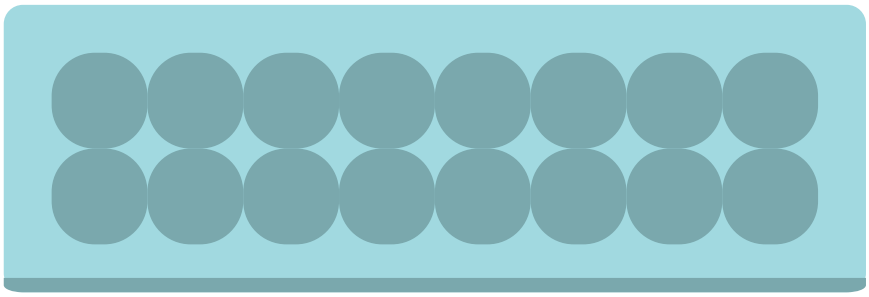
Der Müllsammelroboter in der Aufgabe hat einen ziemlich harmlosen Job. Aber ein Roboter kann auch Arbeiten erledigen, die für Menschen gefährlich sind. Roboter können nach einer Katastrophe nach Überlebenden suchen oder giftige oder radioaktive Materialien einsammeln.

Bibertaler

Bibertaler gibt es nur als Münzen, und zwar 1, 2, 4, 8 und 16 Bibertaler.
Die Biber bezahlen immer den genauen Betrag, mit so wenigen Münzen wie möglich.
Ein Biber kauft Holz und muss 13 Bibertaler bezahlen.

Mit welchen Münzen bezahlt der Biber?



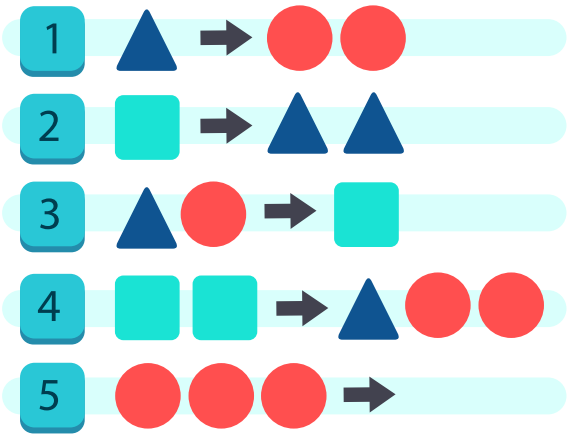


So ist es richtig:
Der Biber bezahlt mit diesen Münzen: 8, 4 und 1. So bezahlt er den genauen Betrag, denn 8 + 4 + 1 = 13 Bibertaler. Mit weniger Münzen kann er nicht bezahlen. Es gibt nämlich keine einzelne Münze und auch keine Kombination von zwei Münzen, mit der man 13 Bibertaler bezahlen kann. Die einzige Kombination von zwei Münzen, die mindestens 13 Bibertaler wert ist, besteht aus zwei 8er-Münzen – aber 8 + 8 = 16, und das ist nicht der genaue Betrag.

Das ist Informatik!
Die Münzen in dieser Biberaufgabe sind so gewählt, dass zwei Münzen von gleichem Wert zusammen immer den Wert der nächstgrößeren Münze haben. Angefangen bei der Münze mit dem Wert 1 gibt es also die Münzen 1, 2, 4, 8 und 16. Das entspricht den ersten fünf Stellen des binären Zahlensystems oder, salopp gesagt, Zweiersystems; die weiteren Stellen sind dann 32, 64, 128, 256 und so weiter.
Während im Zehnersystem an jeder Stelle (1, 10, 100 …) eine der Ziffern 0 bis 9 stehen kann, kann im Zweiersystem an jeder Stelle nur die Ziffer 0 oder die Ziffer 1 stehen. Die Darstellung einer beliebigen Zahl wie der 13 ist dabei immer eindeutig: ein Stellenwert ist entweder verwendet (1) oder nicht (0).
Weil die Bausteine, aus denen Computerspeicher zusammengesetzt sind, ein Bit mit seinen zwei Werten (an oder aus, wahr oder falsch, 0 oder 1) realisieren, ist das Zweiersystem in der Informatik für die Darstellung von Zahlen und allen anderen Informationen von ganz besonderer Bedeutung.

Formenspiel

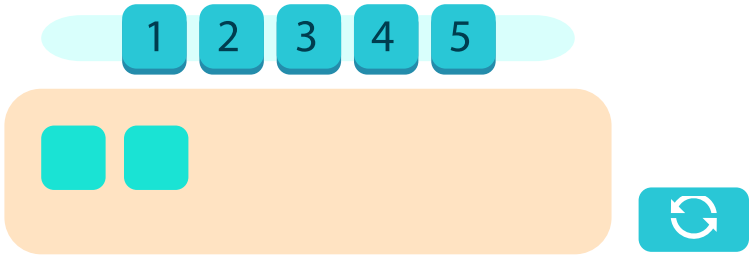
Ganz unten findest du ein Spiel.
Auf dem Spielfeld liegt immer eine Reihe von Formen.



Wenn du einen der Knöpfe über dem Spielfeld drückst, wird auf dem Spielfeld das erste Auftreten (von links) einer oder mehrerer Formen durch eine oder mehrere andere Formen ersetzt.

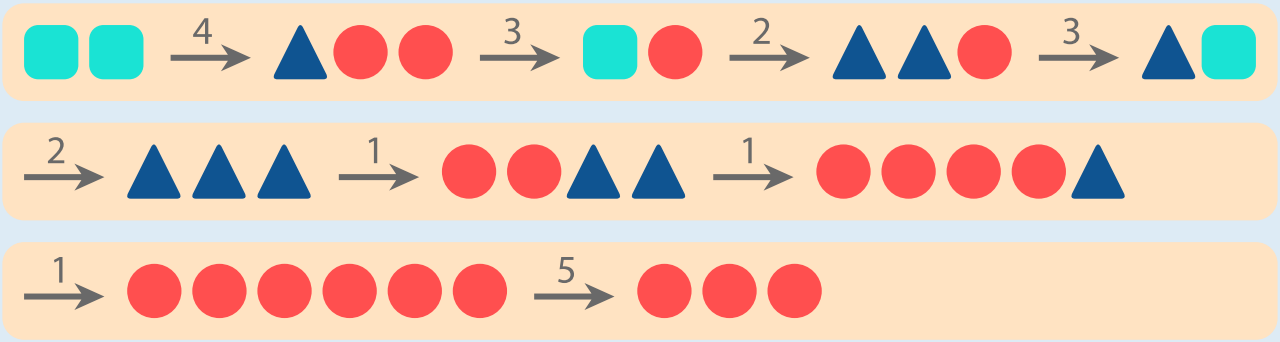
Ein Beispiel: Wenn du Knopf 1 drückst, wird das erste Dreieck durch zwei Kreise ersetzt. Die anderen Knöpfe funktionieren analog. Aber: Wenn du Knopf 5 drückst, werden die ersten drei Kreise durch nichts ersetzt; sie werden also entfernt.

Drücke die Knöpfe so, dass zum Schluss genau drei Kreise auf dem Spielfeld liegen:



So ist es richtig:
Auf den ersten Blick sind die Knöpfe 1 und 2 verführerisch. Mit ihnen lassen sich reichlich Kreise produzieren – aber leider nur genau acht, und aus denen lassen sich mit Knopf 5 nur fünf und zwei Kreise produzieren. Zu einer Lösung kommt man, wenn man vom Ziel her denkt: Drei Kreise lassen sich mit Knopf 1 aus sechs Kreisen produzieren. Sechs Kreise wiederum lassen sich mit Knopf 1 aus drei Dreiecken produzieren. Diese wiederum lassen sich aus den zu Beginn vorhandenen zwei Quadraten durch diese Knopffolge produzieren: 4, 3, 2, 3, 2.

Insgesamt kommt man also so zum Ziel:



Das ist Informatik!

Zur Beantwortung dieser Biberaufgabe muss Folgendes herausgefunden werden: Gibt es, ausgehend von den Startsymbolen (den zwei Quadraten) eine Folge von Ersetzungen, die zu den Zielsymbolen (den drei Kreisen) führt?

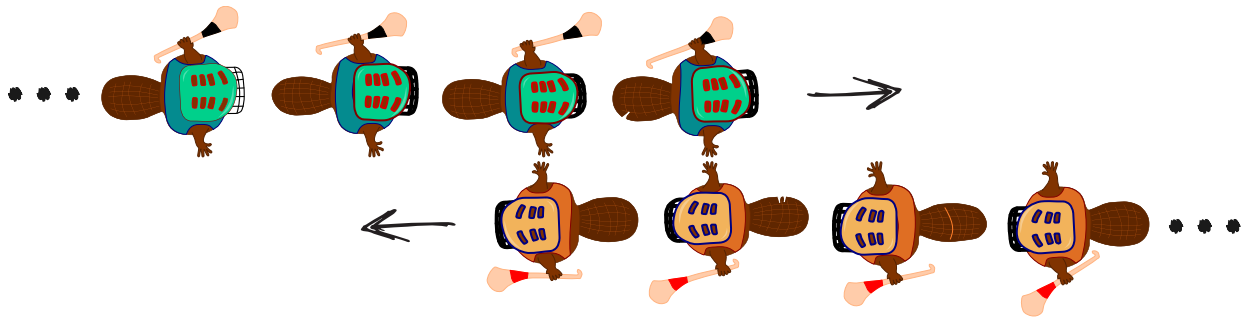
Formen bzw. Symbole und Ersetzungen bilden zusammen ein Ersetzungssystem. In der theoretischen Informatik werden solche Ersetzungssysteme unter bestimmten Bedingungen auch als Grammatiken bezeichnet. Die Symbolfolgen, die, wie hier im Formenspiel, durch Anwendung der Ersetzungen (die Informatik spricht auch von Regeln) einer Grammatik erzeugt werden können, sind dann die Wörter einer Sprache. Für jede Grammatik und die durch sie erzeugte Sprache lässt sich das Wortproblem als Frage formulieren: Gehört ein gegebenes Wort zu der von der Grammatik erzeugten Sprache? Genau diese Frage haben wir oben für die drei Kreise gestellt.

Auch in der Praxis der Informatik spielt das Wortproblem eine wichtige Rolle. Zum Beispiel werden Programmiersprachen durch Grammatiken beschrieben. Ein Wort der (Programmier)Sprache ist dann ein Programm-Quellcode, der keine syntaktischen bzw. strukturellen Fehler mehr hat: alle Kommata oder Semikolons, alle Klammern, alle mathematischen oder anderen Zeichen und so weiter sitzen an der richtigen Stelle. Bevor ein Compiler den Quellcode in ein ausführbares Programm übersetzen kann, muss er zunächst prüfen, ob der Quellcode syntaktisch korrekt ist – er muss also das Wortproblem lösen.

Hände schütteln

Biber spielen gerne das irische Spiel Hurling.

Am Schluss einer Partie Hurling stellen sich die Teams gegenüber auf, jeweils in einer Reihe. Dann gehen die Teams aneinander vorbei, schütteln sich nach und nach die Hände und sagen "Danke für das Spiel!".

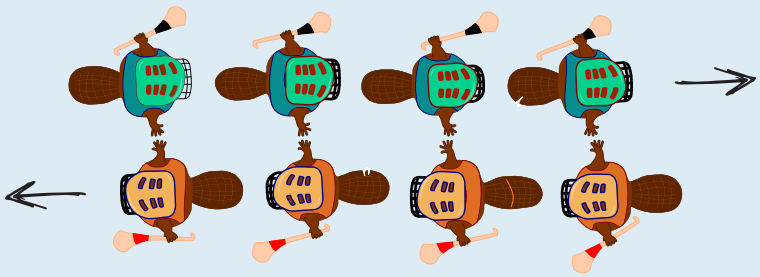


Das Händeschütteln geht so:
Die Teams gehen um einen Spieler vor, so dass die ersten Spieler beieinander stehen und sich die Hände schütteln.
Dann gehen die Teams wieder um einen Spieler vor, so dass die ersten Spieler und die zweiten Spieler beieinander stehen und sich die Hände schütteln (siehe Bild).
Dies geht so weiter, bis auch die beiden letzten Spieler sich die Hände geschüttelt haben.

Beim Hurling gibt es 15 Spieler pro Team. Dass die Teams um einen Spieler vor gehen und sich alle Spieler, die beieinander stehen, die Hände schütteln, dauert 1 Sekunde.

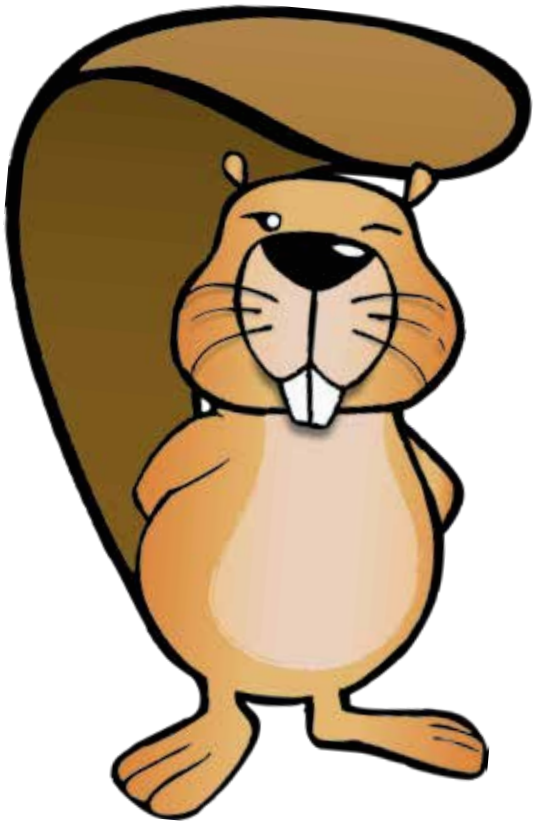
Wie viele Sekunden dauert das Händeschütteln der beiden Teams insgesamt?

29 ist die richtige Antwort:
Wenn der erste Spieler den letzten Spieler erreicht und ihm die Hände geschüttelt hat, sind genau so viele Sekunden vergangen, wie jede Mannschaft Spieler hat. Bei Mannschaften mit vier Spielern sind das vier Sekunden, und die Spieler stehen dann so:

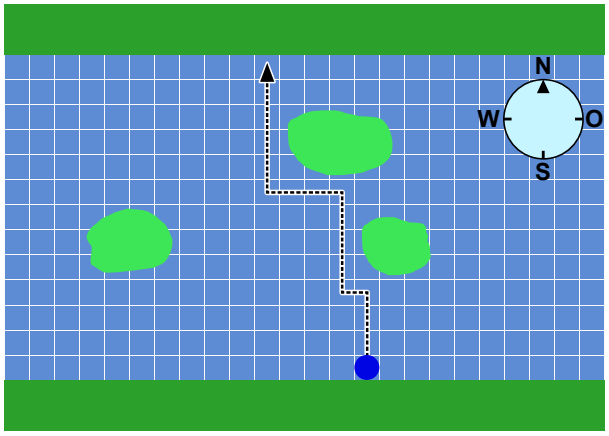


Die letzten Spieler haben dann dem ersten Spieler der jeweils anderen Mannschaft die Hände geschüttelt. Sie müssen danach noch allen weiteren Spielern der anderen Mannschaft die Hände schütteln. Bei vier Spielern pro Mannschaft dauert das noch einmal vier weniger eins gleich drei Sekunden.
Allgemein dauert das Händeschütteln in Sekunden also zweimal die Anzahl der Spieler einer Mannschaft weniger eins. Bei 15 Spielern pro Mannschaft dauert es folglich: (2 x 15) – 1 = 29 Sekunden.

Das ist Informatik!
Für Hurling-Teams mit 15 Spielern konnten wir genau berechnen, wie lange ihr Händeschüttel-Algorithmus abläuft. 29 Sekunden sind für die Zuschauer gut auszuhalten. Doch wie sieht diese „Laufzeit" z.B. bei Eishockey-Teams mit insgesamt 22 Spielern aus? Ist das Hurling-Händeschütteln dann immer noch brauchbar, oder würde es zu lange dauern? Es wäre gut, eine allgemeine Einschätzung der Laufzeit eines Algorithmus zu haben, anstatt Laufzeiten für bestimmte Werte einzeln zu überlegen. Die Informatik befasst sich intensiv mit allgemeinen Einschätzungen von Algorithmen-Laufzeit. Solche Laufzeitanalysen liefern einen mathematischen Ausdruck, der eine Variable n für die Größe der Eingabe enthält. Für das Hurling-Händeschütteln erhalten wir einen solchen Ausdruck, wenn wir im vorletzten Satz der Antworterklärung „Anzahl der Spieler einer Mannschaft" durch n ersetzen: $2n - 1$. Damit lässt sich auch für andere Spielerzahlen die Händeschüttel-Laufzeit berechnen: für 22 Spieler 43 Sekunden, für 40 Spieler 79 Sekunden usw.
Hinter dem Laufzeitausdruck $2n - 1$ steckt eine lineare Funktion. Damit gehört der Hurling-Händeschüttel-Algorithmus zur Klasse der Algorithmen mit linearer Laufzeit, die man auch als $O(n)$ bezeichnet. Ein Händeschüttel-Algorithmus, bei dem jeder jedem anderen nacheinander die Hand gibt, wäre weniger schnell. Er gehörte zur Klasse $O(n^2)$, und die Hurling-Teams würden damit in der Größenordnung von $15^2 = 225$ Sekunden lang Hände schütteln, das sind fast vier Minuten. Gähn! Es lohnt sich also darüber nachzudenken, ob man Dinge parallel, also gleichzeitig erledigen kann – wie die Hurler beim Händeschütteln.

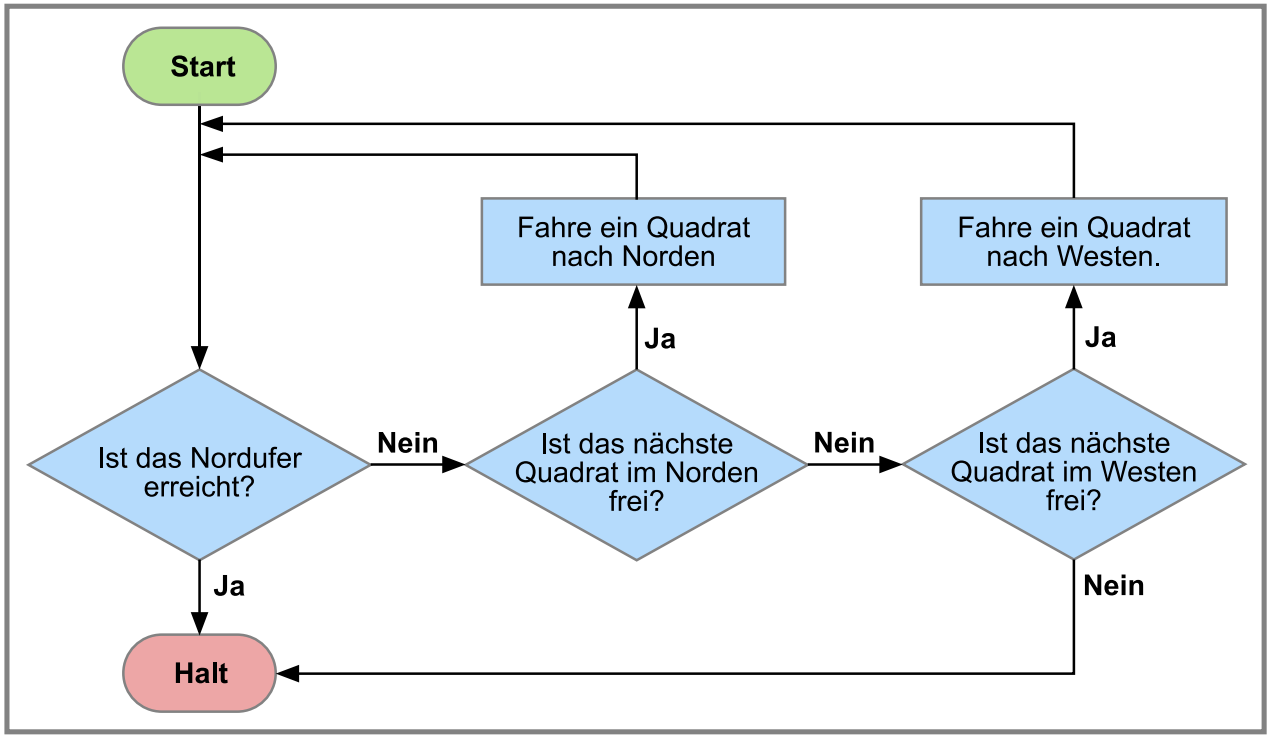


Insel-Falle



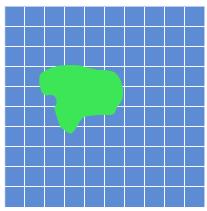
Ein autonomes Boot fährt vom Südufer des Flusses zum Nordufer. Dabei muss es die Inseln im Fluss umfahren.
Die digitale Landkarte des Boots ist in Quadrate eingeteilt. Das autonome Boot fährt immer von einem Quadrat zum nächsten Quadrat im Norden oder Westen. Es kann nur dann zum nächsten Quadrat fahren, wenn dieses völlig frei und nicht zum Teil von einer Insel belegt ist.

Das autonome Boot fährt also nach diesen Anweisungen:

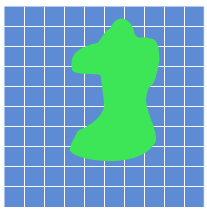


Piraten haben eine künstliche Insel angelegt, als Falle für das autonome Boot.
Das heißt: Je nachdem, wie sich das Boot der Insel nähert, kann es passieren, dass es stehen bleibt und nicht weiterfahren kann. Es ist gefangen.

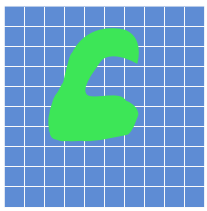
Welche dieser Inseln ist eine Falle für das autonome Boot?



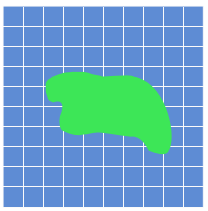
A)



B)

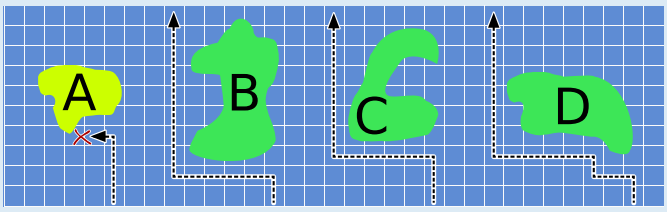


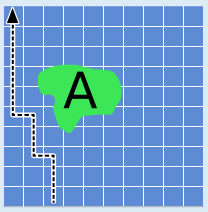
C)



D)

Antwort A ist richtig:





Wenn das autonome Boot Insel A auf der rechten Seite erreicht (wie im Bild links), weicht es zunächst nach Westen aus und bleibt dann stehen, ohne das Nordufer des Flusses zu erreichen. Es steckt in der Falle. Der Grund: Es kann weder nach Westen noch nach Norden weiterfahren, und für andere Bewegungsrichtungen gibt es keine Anweisungen. Wenn es von einem anderen Startpunkt – wie im Bild rechts – losfährt, wird es von der Insel A nicht gefangen.

Die anderen Inseln sind keine Gefahr für das autonome Boot. An den Südufern dieser Inseln gibt es kein freies Quadrat, dessen westliches *und* nördliches Nachbarquadrat belegt ist. Wenn ein autonomes Boot sich von Süden einer dieser Inseln nähert, kann es immer nach Westen ausweichen und die Insel umfahren.

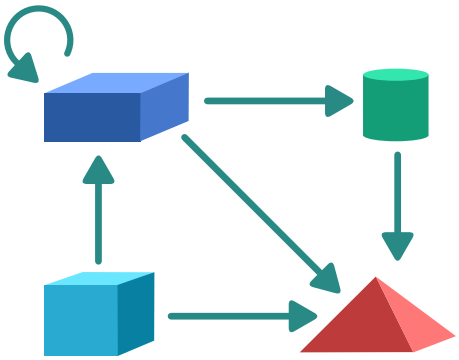
Das ist Informatik!

Die Steuerung des autonomen Boots wurde mit einem Flussdiagramm beschrieben. Ein Flussdiagramm (oder Programmablaufplan, PAP) definiert einen Algorithmus oder Arbeitsablauf. Die Rechtecke des Flussdiagramms enthalten Anweisungen. Die Rauten enthalten jeweils einen Test, d.h. eine Frage, die mit Ja oder Nein beantwortet werden kann. Die Pfeile zeigen an, in welcher Reihenfolge die Anweisungen ausgeführt werden. Die Ovale zeigen Start und Ende des Ablaufs an.

Die Steuerung des autonomen Boots funktionierte perfekt mit den natürlichen rund geformten Inseln des Flusses. Allerdings berücksichtigte der Algorithmus nicht, dass Inseln auch anders geformt sein können und unter bestimmten Bedingungen das autonome Boot an einer Insel anhalten muss und das Nordufer nicht erreichen kann. Das ist eine Schwäche dieses Algorithmus, eine Sicherheitslücke, die von den Piraten ausgenutzt wurde.

Informatikerinnen und Informatiker versuchen Software zu entwickeln, die unter allen denkbaren Bedingungen richtig funktioniert. Interaktive Webseiten zum Beispiel, die Eingabefelder enthalten, prüfen die eingegebenen Texte sorgfältig und weisen Texte zurück, die für den Computer gefährlich sein könnten, z.B. schädliche Skripte (Code Injection).

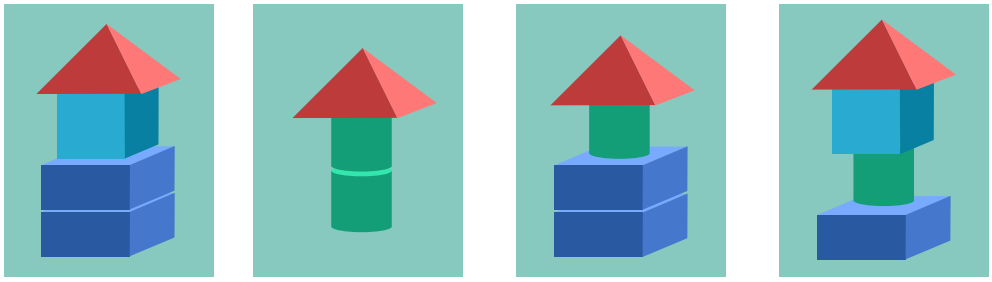
Julias Turm



In ihrem neuen Baukasten findet Julia vier Sorten Bauklötze. Sie erfindet Regeln, wie sie mit den Klötzen Türme bauen möchte. Das Bild zeigt Julias Regeln:
Julia setzt nur dann einen Klotz auf einen anderen, wenn ein Pfeil vom unteren zum oberen Klotz zeigt.

Ein Beispiel: Auf einen blauen Quader darf sie einen grünen Zylinder setzen, eine rote Pyramide, viele andere blaue Quader, aber keinen türkisen Würfel.
Wenn Julia einen Turm baut, fängt sie mit irgendeinem Klotz an und hält sich dann genau an ihre Regeln. Sie hört auf, wenn sie keine Lust mehr hat – oder wenn es nach den Regeln nicht mehr weitergeht.

Welchen Turm hat Julia gebaut?



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

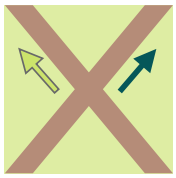
Antwort C ist richtig:
So baut Julia Turm C: Sie fängt mit einem blauen Quader an, setzt einen weiteren blauen Quader darauf, folgt dann dem Pfeil zum grünen Zylinder und abschließend dem Pfeil zur roten Pyramide. Damit ist Schluss, denn von der roten Pyramide zeigt kein Pfeil zu einem nächsten Klotz.
Die anderen Türme hat Julia nicht gebaut, denn sie verletzen die Regeln.
Turm A: Ein türkiser Würfel sitzt auf einem blauen Quader, obwohl es in den Regeln keinen Pfeil vom Quader zum Würfel gibt – nur umgekehrt.
Turm B: Ein grüner Zylinder sitzt auf einem anderen, obwohl es in den Regeln keinen "Kreispfeil" von Zylinder zu Zylinder gibt.
Turm D: Ein türkiser Würfel sitzt auf einem grünen Zylinder, obwohl es in den Regeln keinen Pfeil vom Zylinder zum Würfel gibt.

Das ist Informatik!
Julias Turmbau-Regeln orientieren sich daran, welcher Klotz gerade oben auf dem Turm sitzt. Der oberste Klotz ist also entscheidend für den aktuellen Zustand des Turms. Die Regeln legen dann die möglichen nächsten Zustände des Turms fest. Das Regelbild in dieser Biberaufgabe ist also ein Zustandsübergangsdiagramm für Bauklotz-Türme.
Zustandsübergangsdiagramme werden in der Informatik häufig verwendet. Sie eignen sich gut zur Beschreibung ganz unterschiedlicher Dinge: das Verhalten von Softwaremodulen oder auch ganzer Programme, einfache sprachliche Strukturen, das Zusammenspiel von Hardware-Bauteilen und vieles mehr. Mit Hilfe einer solchen formalen Beschreibung kann dann auch getestet werden: ob die Software sich wie gewünscht verhält – oder ob der Turm richtig gebaut ist.

Kanalsystem

Die blauen und die grünen Biber haben zusammen ein Kanalsystem gebaut.
Es hat sechs Eingänge (unten) und sechs Ausgänge (oben), und es gibt Kreuzungen.

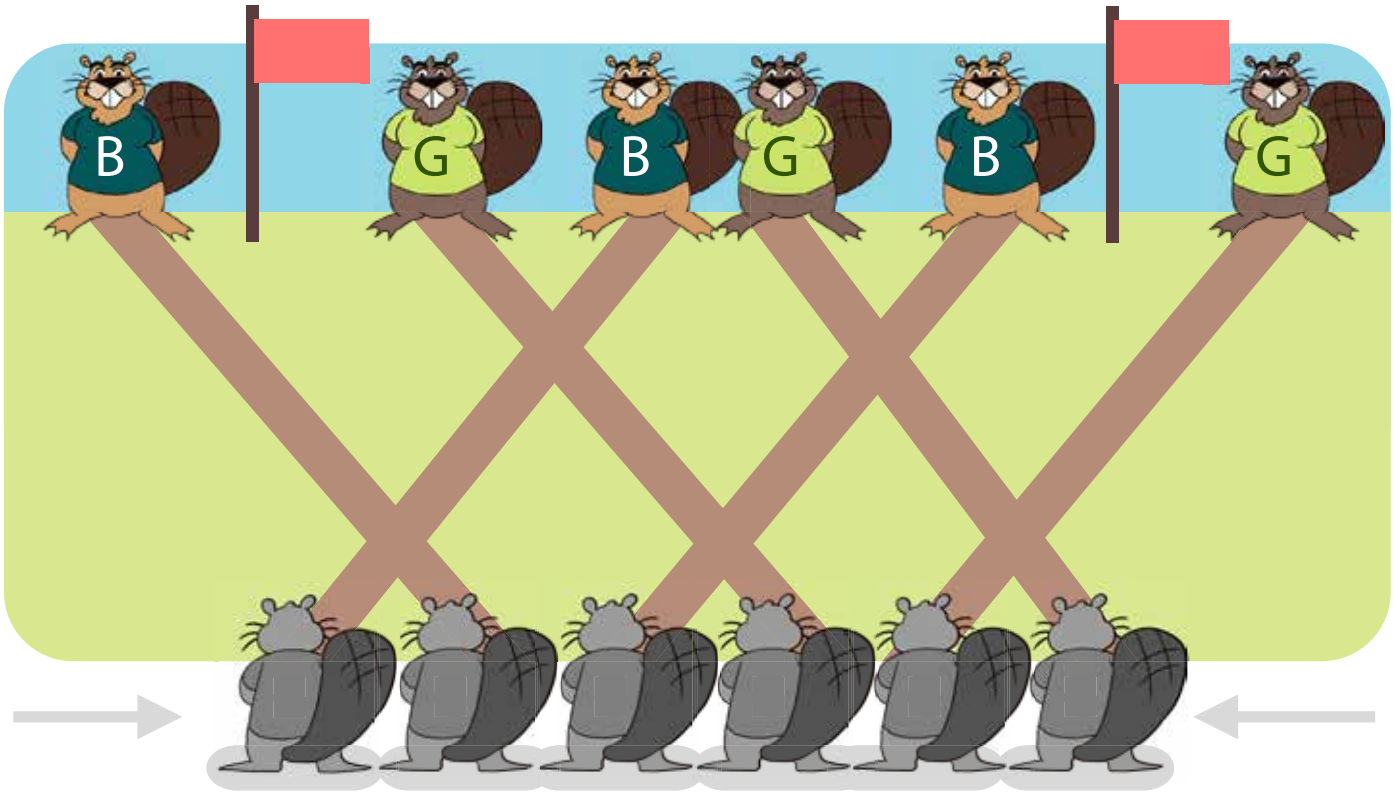
Für die Kreuzungen gibt es eine Regel:
Wenn sich ein grüner und ein blauer Biber treffen,
schwimmt der grüne nach links und der blaue nach rechts.



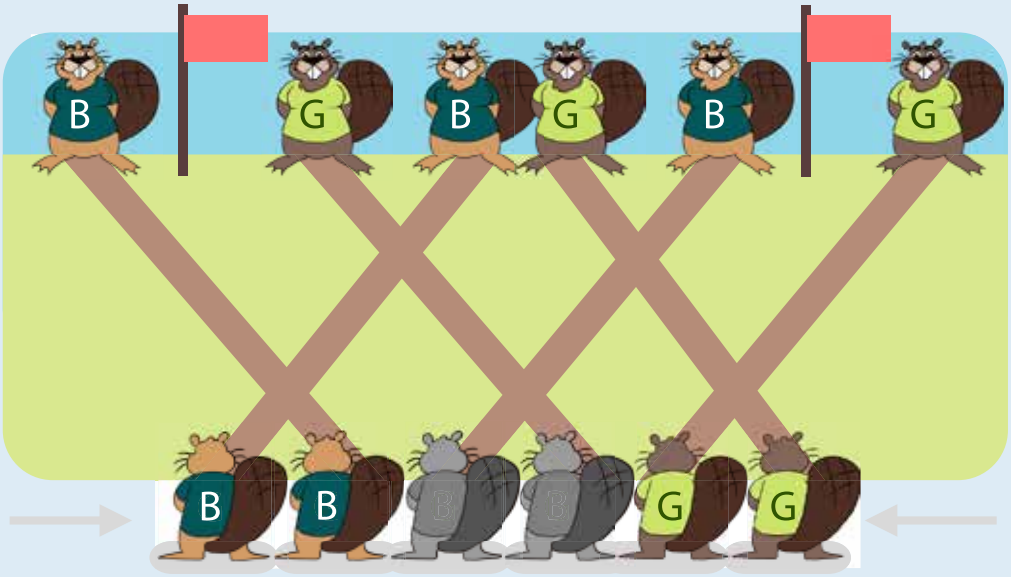
Drei blaue und drei grüne Biber schwimmen gleichzeitig an den Eingängen los.

An den Ausgängen sollen sie so ankommen (von links nach rechts, B = blau und G = grün):
B G B G B G

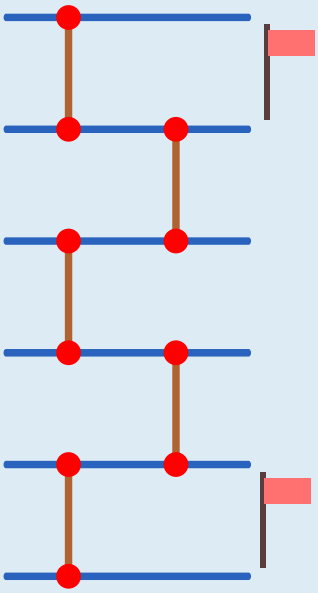
Wie müssen sie an den Eingängen los schwimmen?



So ist es richtig:
Damit am linken Ausgang ein blauer Biber ankommen kann, müssen an den beiden linken Eingängen zwei blaue Biber los schwimmen. Ansonsten würde nämlich ein grüner Biber ganz links ankommen.
Damit am rechten Ausgang ein grüner Biber ankommen kann, müssen an den beiden rechten Eingängen zwei grüne Biber los schwimmen. Ansonsten würde nämlich ein blauer Biber ganz rechts ankommen.
An den mittleren beiden Eingängen spielt die Farbe keine Rolle: Egal, an welchem der dritte blaue und der dritte grüne Biber los schwimmen, schwimmt nach der ersten Kreuzung der grüne Biber links und der blaue Biber rechts weiter. Und dann kommen die Biber an den Ausgängen so an wie vorgegeben.
Welcher der beiden Biber an den mittleren Eingängen blau und welcher grün ist, ist also gleich.



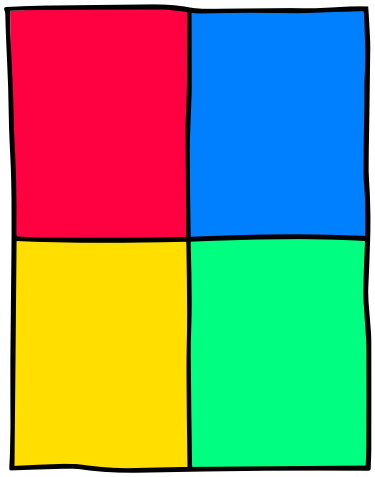
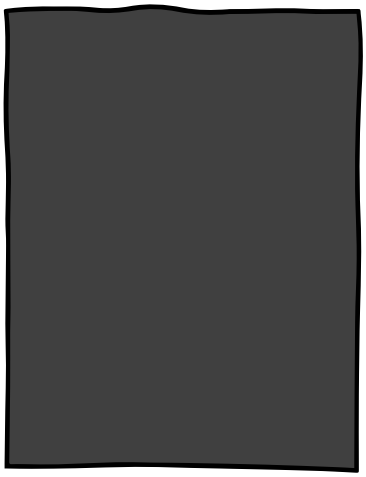
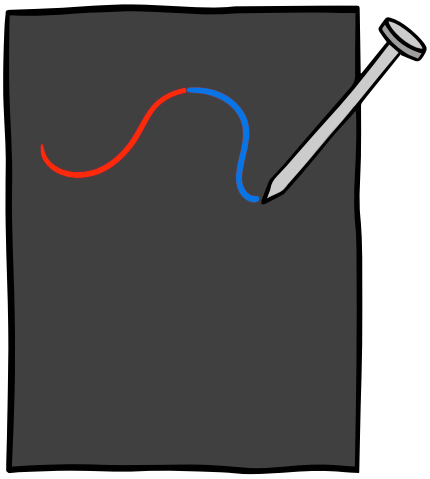
Das ist Informatik!
Computer verarbeiten Daten – häufig sehr viele Daten. Dabei ist es wichtig, dass in den Daten Ordnung herrscht. In der Informatik sagt man statt ordnen meist sortieren: Zum Beispiel können Zahlen nach aufsteigender Größe, Daten über Personen nach dem Geburtsdatum oder Daten über Bücher nach der ISBN-Nummer sortiert werden. Am Ende eines Sortiervorgangs stehen die Daten in der gewünschten Reihenfolge.
Die wichtigste Operation beim Sortieren ist der „Swap“: Daten an zwei Positionen in der Reihenfolge werden miteinander verglichen. Ist ihre Sortierung untereinander richtig, bleiben sie stehen, sonst werden sie vertauscht. Genau das passiert in dieser Biberaufgabe an den Kreuzungen des Kanalsystems.
Das Kanalsystem der Biber funktioniert nämlich wie ein Teil eines Sortiernetzes. In einem Sortiernetz wandern Daten entlang von Linien (hier: die Kanäle), die an manchen Stellen paarweise miteinander verbunden sind (hier: die Kreuzungen). Bei einer Verbindung wird geprüft, ob die Daten der beiden Linien getauscht werden sollen.
Das Bild stellt das Kanalsystem als Sortiernetz-Teil dar. Mehr Biberaufgaben mit Sortiernetzen findest du in den Biberheften von 2011 („Sortierende Brücken“), 2014 („Pfützenspringen“) und 2018 („Büchertausch“). Zum Thema Sortieren gibt es in diesem Biberheft noch die Aufgaben „Anproben“, „Teller-Ordnung“ und „Türme“.



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

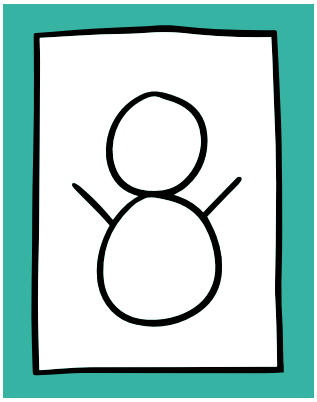
Kratzbilder

Die kleinen Biber machen gerne Kratzbilder.

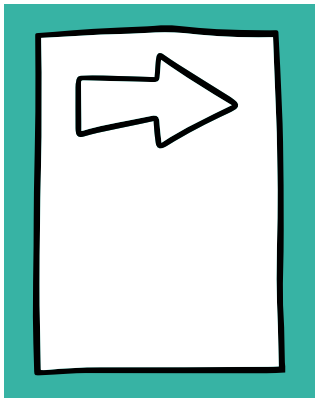
Zuerst malen sie 4 Farben genau so auf Papier.	Dann malen sie mit Schwarz darüber.	Zuletzt kratzen sie ein Bild in das Schwarze. Dann können sie die bunten Farben wieder sehen.
		

Die Biber kratzen diese Bilder.

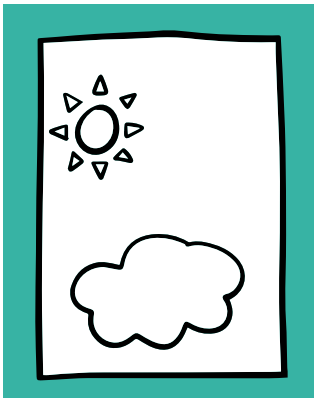
Bei welchem Bild können sie genau 3 bunte Farben sehen?



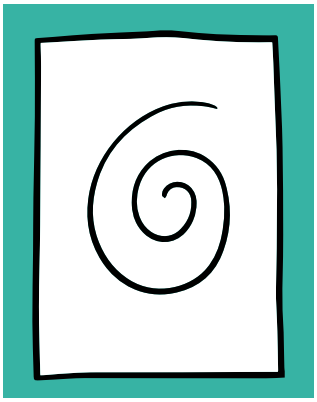
A)



B)



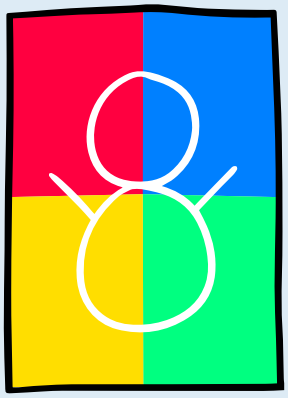
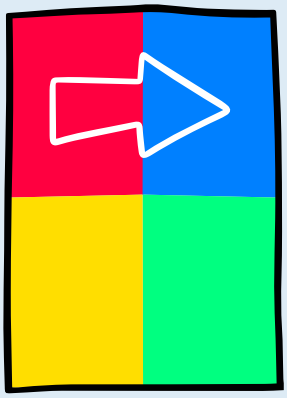
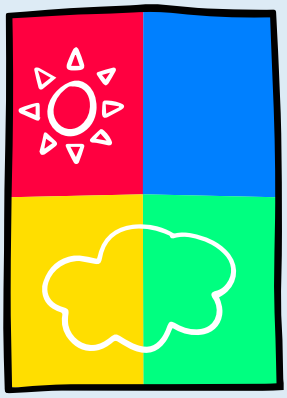
C)



D)

Antwort C ist richtig:

So liegen die Bilder über den bunten Farben:

A)	B)	C)	D)
			
4 Farben	3 Farben	3 Farben	4 Farben

Nur bei Bild C können die Biber genau 3 Farben sehen.

Das ist Informatik!

Die Kratzbilder bestehen aus zwei Schichten, nämlich aus der Schicht mit den bunten Farben und der schwarzen Schicht darüber. Um die Anzahl der Farben zu ermitteln, muss man genau wissen, wo die Farben auf der unteren Schicht unter der schwarzen Schicht versteckt sind.

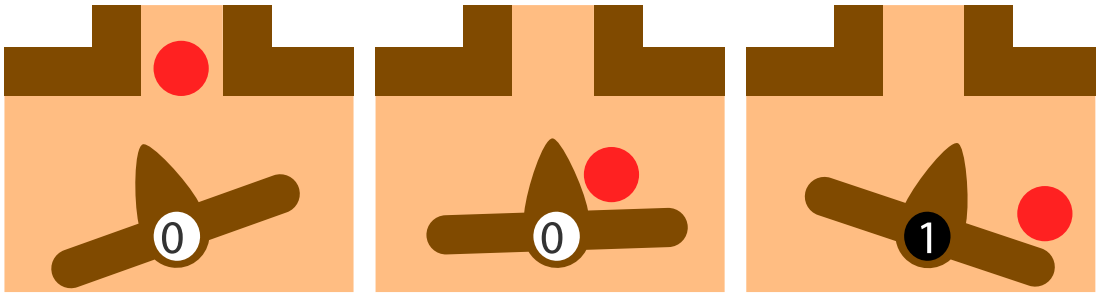
Wer am Computer Bilder macht, kann dabei auch Schichten (die oft auch Ebenen heißen) übereinander legen – wie beim Kratzpapier in dieser Biberaufgabe. So kann man den Hintergrund eines Bildes (zum Beispiel die Grashalme einer Wiese) und die Dinge, die vor dem Hintergrund erscheinen sollen (zum Beispiel verschiedene Blumen) auf unterschiedliche Schichten malen – und dann die Schicht mit dem Hintergrund in einem anderen Bild gleich noch einmal verwenden. Am Computer-Bildschirm können die Schichten sogar gemischt werden. Dazu muss man festlegen, dass die oberste Schicht halb durchsichtig (transparent) sein soll, wie etwa eine farbige Folie.

Kugelbahn

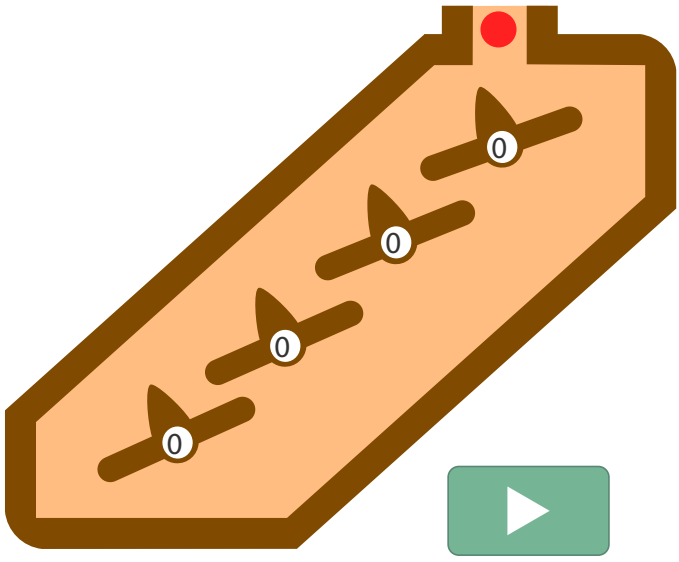
Eine Kugelbahn hat vier Wippen. Jede Wippe kann auf zwei Positionen stehen:

- Ist die Wippe nach links geneigt, steht sie auf Position 0.
- Ist die Wippe nach rechts geneigt, steht sie auf Position 1.

Wenn eine Kugel auf eine Wippe trifft, ändert die Wippe ihre Position und die Kugel rollt



Hier ist eine Animation der Kugelbahn.
Zu Beginn stehen alle Wippen auf Position 0; die Wippen zeigen insgesamt also: 0000
Dann rollen nacheinander zwei Kugeln. Danach zeigen die Wippen: 0010



Die Kugelbahn wird zurückgesetzt, und die Wippen zeigen wieder 0000.
Nun rollen nacheinander fünf Kugeln.

Was zeigen die Wippen jetzt?

0011 **0100** **0101** **0111** **1010** **1100**

„0101“ ist die richtige Antwort:
Die oberste Wippe ändert ihre Position bei jeder Kugel. Wenn also wie in diesem Fall eine ungerade Anzahl von Kugeln durch die Kugelbahn rollt, steht die oberste Wippe auf 1. Damit ist die letzte Ziffer eine 1.

Die zweitoberste Wippe ändert ihre Position nur, wenn die oberste Wippe vorher auf 1 stand. Das ist nur jedes zweite Mal der Fall. Sie ändert ihre Position also zweimal und steht am Ende auf 0. Damit ist die vorletzte Ziffer eine 0.

Die drittoberste Wippe ändert ihre Position nur, wenn die zweitoberste Wippe vorher auf 1 stand und überhaupt eine Kugel auf sie trifft, also auch die oberste Wippe vorher auf 1 stand. Damit ändert die drittoberste Wippe ihre Position nur jedes vierte Mal, also hier nur bei der vierten Kugel. Damit steht sie am Ende auf 1, Damit ist die drittletzte Ziffer eine 1.

Die unterste Wippe ändert ihre Position nur, wenn alle drei Wippen über ihr vorher in der Position 1 waren und sie damit jeweils von einer Kugel getroffen werden. Das ist nur jedes achte Mal der Fall, die unterste Wippe ändert ihre Position bei fünf Kugeln also nicht. Damit ist die erste Ziffer eine 0.
Insgesamt zeigen die Wippen: 0101

Das ist Informatik!
ie Wippen der Kugelbahn schalten zwischen zwei Zuständen hin und her: 0 und 1. Darin gleichen sie den elektronischen Bausteinen, die in digitalen Geräten die kleinste Speichereinheit realisieren: das Bit.

Zwei hintereinander liegende Wippen sind wie ein Halbaddierer verbunden. Dieser hat als Eingabe zum einen den aktuellen Zustand des Bits und zum anderen einen Impuls. Die Ausgabe ist ein neuer Zustand des Bits und ein Übertrag.

Gespeicherter Zustand	Impuls	Zu speichernder Zustand	Übertrag
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Die Kugel selbst liefert den Impuls 1 für die erste Wippe. Insgesamt verändern sich die Zustände der vier Wippen so: 0000 ⟶ 0001 ⟶ 0010 ⟶ 0011 ⟶ 0100 ⟶ 0101. Wenn man diese Bitfolgen als Binärzahlen versteht, bedeutet das: 0 ⟶ 1 ⟶ 2 ⟶ 3 ⟶ 4 ⟶ 5. Zusammen mit der Kugel funktionieren die „Bit-Wippen“ in dieser Biberaufgabe also wie ein (binäres) Zählwerk.

Leckeres Holz

Ben veranstaltet eine Party für sich und seine sechs Biber-Freunde, mit viel leckerem Holz!

Ahorn Birke Eiche Esche Linde Weide

Leider haben alle sieben Biber einen eigenen Geschmack:

Dieser Biber …	**… findet diese Holzsorten lecker:**
Anne	Ahorn, Eiche, Esche, Weide
Ben	Eiche, Linde, Weide
Carlo	Eiche
Doris	Birke, Esche
Emre	Ahorn, Birke, Weide
Freddy	Eiche, Esche
Georg	Ahorn, Linde

Ben möchte sich Arbeit sparen und möglichst wenige Holzsorten besorgen.
Aber natürlich möchte er für jeden Biber eine leckere Sorte haben.

Wie viele Holzsorten muss Ben mindestens besorgen?

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

Antwort C ist richtig:
Ben muss auf jeden Fall Eiche besorgen, nämlich für Carlo. Damit sind auch Anne, Ben selbst und Freddy zufrieden. Doris, Emre und Georg haben zu dritt keine leckere Holzsorte gemeinsam. Aber mit zwei weiteren Holzsorten, z.B. Birke und Ahorn, kann Ben auch diese drei Biber zufriedenstellen. Insgesamt muss Ben also drei Holzsorten besorgen, damit er für jeden Biber eine leckere Sorte hat.

Das ist Informatik!
In der Aufgabenstellung ist für jeden Biber in einer Tabelle angegeben, welche Holzsorten er lecker findet. Für die Beantwortung der Frage wäre auch eine andere Tabelle nützlich gewesen, in der nämlich für jede Holzsorte angegeben wird, welcher Biber sie lecker findet:

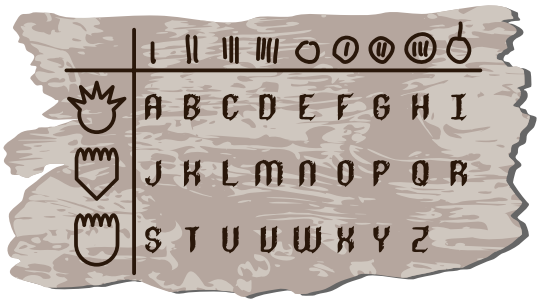
Diese Holzsorte …	**… finden diese Biber lecker:**
Ahorn	Anne, Emre, Georg
Birke	Doris, Emre
Eiche	Anne, Ben, Carlo, Freddy
Esche	Anne, Doris, Freddy
Linde	Ben, Georg
Weide	Anne, Ben, Emre

Mit jeder Holzsorte ist dann ein Teil der Gesamtmenge der sieben Biber verbunden. Gastgeber Ben sucht dann die kleinste Auswahl aus diesen Teilmengen, so dass insgesamt alle sieben Biber in der Auswahl enthalten sind. Zum Beispiel sind insgesamt alle Biber in den Mengen für Ahorn, Birke und Eiche enthalten.

Das Problem, aus vielen Teilmengen einer Gesamtmenge möglichst wenige so auszuwählen, dass die Vereinigung dieser Teilmengen die Gesamtmenge ergibt, ist in der Informatik und der Mathematik gut bekannt. Es wird als „Mengenüberdeckungsproblem“ oder auf Englisch auch als „set cover problem“ bezeichnet. Auch wenn diese Biberaufgabe nicht allzu schwer zu lösen war: „Set cover“ gehört im Allgemeinen zu den schwierigsten Problemen der Informatik, den so genannten „NP-vollständigen Problemen“. Die Informatik kennt keine Verfahren, mit denen diese Probleme in allen Fällen effizient, also in für Menschen sinnvoller Zeit perfekt gelöst werden können. Stattdessen sind häufig Verfahren bekannt, die zwar nicht immer die besten, aber recht gute Lösungen finden.

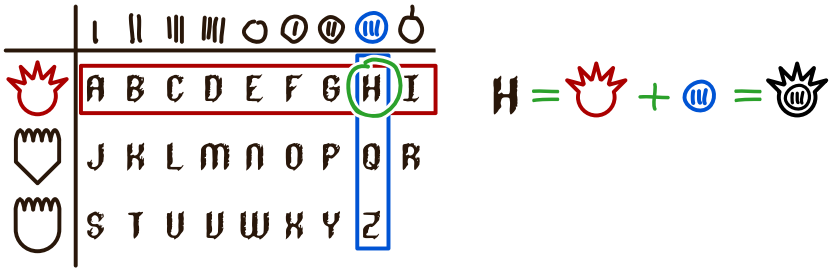
https://en.wikipedia.org/wiki/Set_cover_problem

Lenas Nachricht



Am Biberdamm findet Lena ein Stück Holz. Darin sind Zeichen und Buchstaben eingeritzt und in einer Tabelle angeordnet.
Lena überlegt sich, dass sie mit Hilfe der Tabelle Nachrichten kodieren kann.

Jeder Buchstabe wird durch eine Kombination aus zwei Zeichen kodiert. Ein Beispiel:



Lena kodiert nun eine Nachricht aus neun Buchstaben mit Hilfe der Tabelle:

Wie lautet Lenas Nachricht?

A) SAVEWATER **B)** CLEARDAYS **C)** SAVEMYDAM **D)** CAREFORME

Antwort A ist richtig:
Den ersten Buchstaben ihrer Nachricht hat Lena so als Kombination aus zwei Zeichen kodiert: + . Der Buchstabe steht also in der dritten Reihe und der ersten Spalte der Tabelle: das S. Damit können nur noch die Antworten A und C richtig sein. Die folgenden drei Zeichenkombinationen in Lenas Kodierung entsprechen nacheinander den Buchstaben A, V und E, so dass immer noch beide Antworten in Frage kommen. Den fünften Buchstaben hat Lena so kodiert: + . Der fünfte Buchstabe steht also in der dritten Reihe und der fünften Spalte der Tabelle: das W. Damit kann nur noch Antwort A richtig sein, und die letzten Zeichen in der Kodierung entsprechen dann auch den Buchstaben A, T, E und R.
Die richtige Antwort lässt sich aber auch schneller finden. Den letzten Buchstaben in ihrer Nachricht hat Lena so kodiert: + . Der letzte Buchstabe steht also in der zweiten Reihe und letzten Spalte der Tabelle: das R. Nur Antwort A endet mit einem R.

Das ist Informatik!
Wenn Daten wie etwa die Buchstaben des Alphabets durch andere Daten (wie Lenas Zeichen) so ersetzt werden, dass eine Rückersetzung eindeutig möglich ist, spricht die Informatik von einer Kodierung bzw. einem Code. Es gibt viele Gründe dafür, einen Code anzuwenden. Zum Beispiel kann eine passende Kodierung die Verarbeitung von Daten erleichtern oder sogar erst ermöglich: Daten, die mit Hilfe von Computern gespeichert und verarbeitet werden sollen, müssen binär kodiert werden, also indem Buchstaben, Ziffern, Bildpunkte, Töne usw. durch Folgen von 0- und 1-Werten ersetzt werden.
Besonders berühmt sind Codes zur Verschlüsselung und damit Geheimhaltung von Daten. Beim Verschlüsseln ist es aber keine gute Idee, jedes Zeichen immer wieder auf die gleiche Weise zu ersetzen, wie in dieser Biberaufgabe. Solche Verschlüsselungen bezeichnet man auch als „monoalphabetische Substitution“ (Substitution ist ein Fremdwort für Ersetzung), und sie sind besonders leicht zu knacken. Moderne Verfahren zur Verschlüsselung arbeiten deshalb mit ganz anderen, meist stark mathematisch geprägten Kodierungsmethoden; ein Beispiel ist die „Elliptic Curve Cryptography“.
https://de.wikipedia.org/wiki/Kryptographie

Lutscher

Es gibt Lutscher, umsonst! Vier Biber stellen sich an.
Wer an der Reihe ist, bekommt den Lutscher, der am nächsten ist.
Der erste Biber bekommt also den Lutscher mit dem grünen Quadrat .

Wer bekommt den Lutscher mit dem roten Dreieck ?

So ist es richtig:
Die Biber stehen rechts. Also ist immer der Lutscher, der am weitesten rechts ist, am nächsten.

Zuerst ist der Lutscher am weitesten rechts und wird an den ersten Biber verschenkt.

Danach ist der Lutscher am weitesten rechts und wird an den zweiten Biber verschenkt.

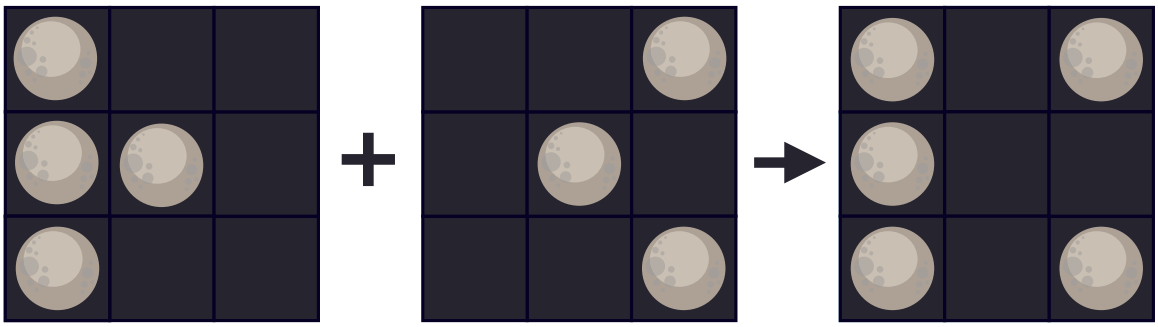
Dann ist der Lutscher am weitesten rechts und wird an den dritten Biber verschenkt.
Das Bild zeigt, welche Biber welche Lutscher bekommen.

Das ist Informatik!
Die Biber stehen Schlange und warten darauf, einen Lutscher geschenkt zu bekommen. Der Biber, der in der Biberschlange vorne ist, ist als nächster an der Reihe. Dann verlässt er die Schlange, und der nächste Biber ist dran. Würde noch ein Biber dazu kommen, müsste er sich hinten in der Schlange anstellen. Warteschlangen wie in dieser Biberaufgabe gibt es auch in der Informatik. Dort sind sie unter dem englischen Begriff „queue“ (sprich: kjuh) bekannt. In einer Queue werden Datenobjekte gespeichert und nach dem Prinzip verwaltet, dass das zuerst hinzugefügte Objekt auch zuerst wieder entfernt wird (englisch “First-In-First-Out”, kurz “FIFO”). In der Computerprogrammierung werden Queues zum Beispiel dann verwendet, wenn Daten zwischengespeichert und in der Reihenfolge der Speicherung bearbeitet werden müssen. Wenn etwa viele Druckaufträge auf einmal an einen bestimmten Drucker geschickt werden, werden diese Aufträge in der Reihenfolge des Eintreffens in einer Warteschlange zwischengespeichert. Es wird dann derjenige Auftrag als nächster ausgedruckt, der schon am längsten, also „vorne“ in der Warteschlange ist.

Mondzauber

Der weise Anaxagoras besitzt Mondzauberkarten,
die für jedes ihrer neun Felder nach einer festen Regel funktionieren.
Bei zwei Karten gibt er in jedes Feld ein Mondsymbol ein:
Neumond (schwarz) oder Vollmond.
Sofort erscheinen auch Mondsymbole auf einer dritten Karte.

Hier ein Beispiel, links die zwei Eingabekarten, rechts die Ausgabekarte:



Verstehst Du die Regel?
Dann klicke auf die Felder der Ausgabekarte rechts,
bis alle das richtige Mondsymbol zeigen.

So ist es richtig:

Die Regel ist: Wenn auf den beiden Eingabekarten im gleichen Feld genau einmal Vollmond vorkommt, dann zeigt das entsprechende Feld der Ausgabekarte das Mondsymbol Vollmond, andernfalls das Mondsymbol Neumond.

Das ist Informatik!

In allen Bereichen der Informatik wird gerne zweiwertige Logik benutzt, um über die Wahrheit und Falschheit von Aussagen innerhalb des Modells einer Anwendung zu entscheiden. Das geschieht immer unter der Voraussetzung, dass es im Modell dafür genau zwei Möglichkeiten (zwei Werte) gibt.
Eine dritte Möglichkeit ist ausgeschlossen: tertium non datur.

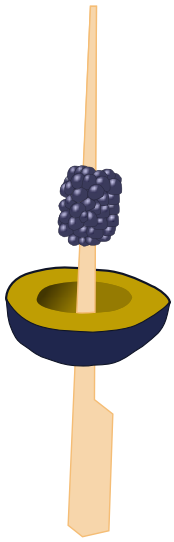
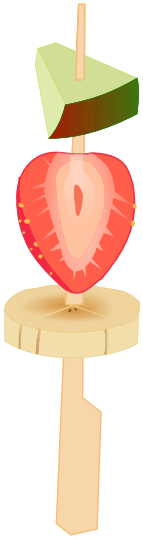
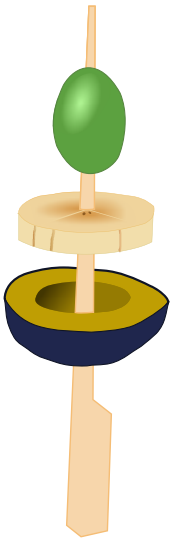
Einige Beispiele für solche Zwei-Werte-Mengen in Modellen sind: {zu auf}, {kalt heiss}, {ungerade gerade}, {leer voll}, {schwarz weiss}, {nein ja}, {0 1}, {Neumond Vollmond}. Auf einer solchen Grundlage kann man dann Aussagen innerhalb des Modells logisch miteinander verknüpfen. Eine logische Funktion liefert entweder den Wahrheitswert „falsch“ oder den Wahrheitswert „wahr“. Weil man auch logische Funktionen miteinander verknüpfen kann, gibt es davon im Prinzip beliebig viele.

In dieser Biber-Aufgabe benutzen wir die Funktion XOR („ausschließendes Oder“, englisch: „exclusive or“), weil sie besonders häufig vorkommt und leicht zu verstehen ist. XOR funktioniert so: Wenn von einer Anzahl Aussagen genau eine „wahr“ ist und alle anderen „falsch“, dann liefert XOR den Wahrheitswert „wahr“. In allen anderen Fällen liefert XOR den Wahrheitswert „falsch“. Im Beispiel zu dieser Aufgabe wird die logische Funktion XOR neunmal (Felder) mit zwei Aussagen (Eingabekarten) vorgeführt.

Übrigens: Anaxagoras hat als erster erkannt, dass der Mond nicht von sich aus leuchtet. Nun trägt ein Mondkrater den Namen mit dem versteckten XOR.

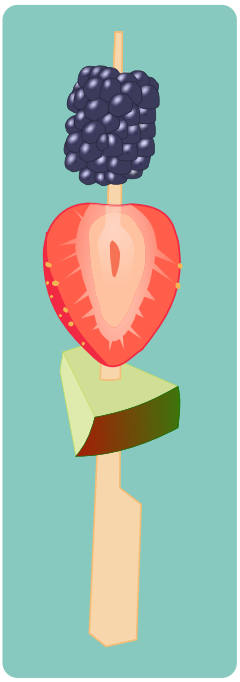
Obstspieße

Bei Leos Party gibt es Obstspieße.
Dafür stehen sechs Obstsorten zur Auswahl.
Die Sorten haben die Nummern 1 bis 6.
Die Tabelle zeigt, welche Obstsorten für die ersten vier Spieße gewählt wurden.

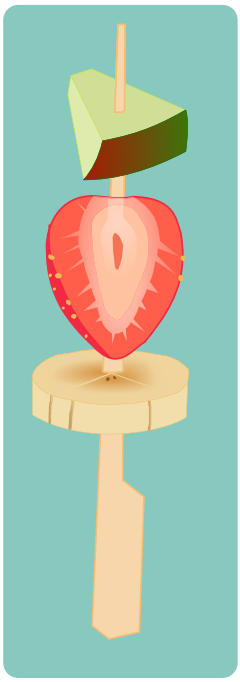
Spieß				
Obstsorten	3, 6	1, 2, 5	2, 5, 6	2, 3, 4

Leo wählt für seinen Spieß die Obstsorten 1, 5 und 6.

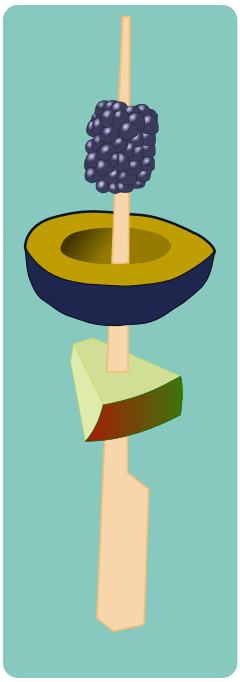
Welcher ist Leos Spieß?



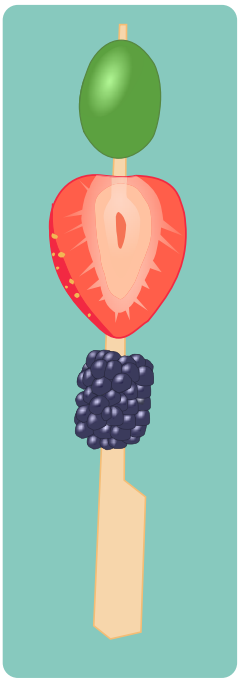
A)



B)

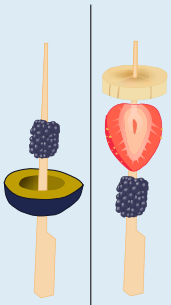
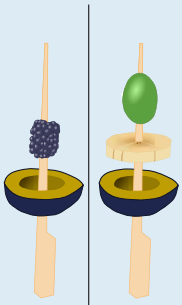
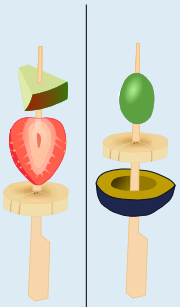
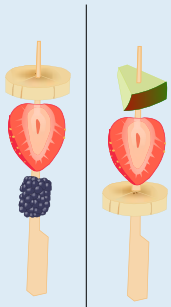


C)



D)

Antwort A ist richtig:
Um herauszufinden, welche Nummer zu welcher Obstsorte gehört, kann man die Spieße paarweise miteinander vergleichen. Zum Glück haben sie immer nur eine gemeinsame Sorte:

Verglichene Spieße				
Sorten	3, 6 2, 5, 6	3, 6 2, 3, 4	1, 2, 5 2, 3, 4	2, 5, 6 1, 2, 5
Gemeinsame Nummer	6	3	2	5
Gemeinsame Sorte	Brombeere	Zwetschge	Banane	Erdbeere

Spieß		
Sorten	1, 2, 5	2, 3, 4
Bekannte Sorten	2: Banane 5: Erdbeere	2: Banane 3: Zwetschge
Unbekannte Sorten	1: Apfel	4: Weintraube

Jetzt sind nur noch die Sorten 1 und 4 herauszufinden. Dazu kann man sich die zwei Spieße ansehen, die genau eine dieser Sorten enthalten – alle anderen Sorten sind also schon bekannt.
Nun ist für alle Nummern bekannt, zu welcher Obstsorte sie gehören. Leo hat für seinen Spieß also Apfel, Erdbeere und Brombeere gewählt. Das ist Spieß A.

Das ist Informatik!
Obstsorten und Nummern sollen eindeutig einander zugeordnet werden. Es sollen also Paare entstehen, bei denen jede Sorte bzw. jede Nummer wirklich nur genau einen „Partner“ hat. Als Ausgangspunkt sind für jede Obstsorte anhand der Obstspieße einige Nummern bekannt, die als Partner in Frage kommen. Das ähnelt sehr dem „Heiratsproblem“: Dabei gibt es einige Frauen und einige Männer (nicht unbedingt gleich viele). Für jede Frau ist bekannt, welche Männer sie sich als Ehemann vorstellen kann, und zwar mindestens einen. Dann wird nach einer eindeutigen Zuordnung (wie oben) gesucht, bei der jeder Frau einer ihrer Wunschkandidaten zugeordnet wird. In der Informatik spricht man von einem „Matching“ – und zwar auch dann, wenn es nicht um Frauen und Männer geht. Bei Obstsorten und Nummern wird in dieser Biberaufgabe sogar ein „perfektes Matching“ gesucht, bei dem jede Sorte eine Nummer abbekommt und umgekehrt. Dazu muss es natürlich genau so viele Nummern wie Sorten geben.
Übrigens:
In den USA erhalten Medizinstudenten per Matching-Algorithmus eine Stelle zur Facharztausbildung.

https://algo.rwth-aachen.de/~algorithmus/algo22.php
https://de.wikipedia.org/wiki/Stable_Marriage_Problem

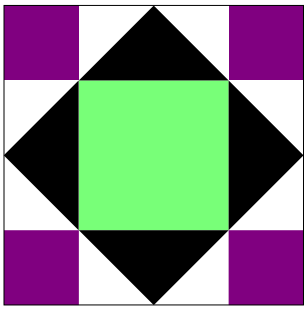
Rangoli

Rangoli sind kunstvolle Bodenmuster.
Man kann sie zum Beispiel mit bunten Fliesen legen.
Indu hat drei verschiedene Sorten Fliesen:
8 lila Dreiecke, 4 grüne Quadrate und 6 schwarze Dreiecke.

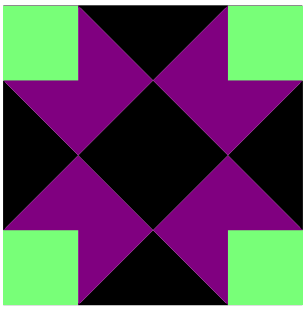
Sorte			
Anzahl	8	4	6

Indu möchte mit ihren Fliesen ein Rangoli legen.
Sie muss aber nicht alle Fliesen benutzen.

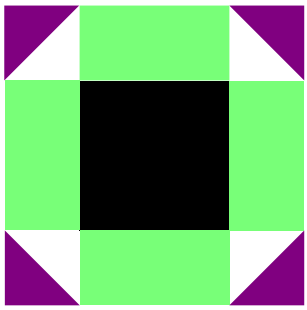
Welches Rangoli kann Indu mit ihren Fliesen legen?



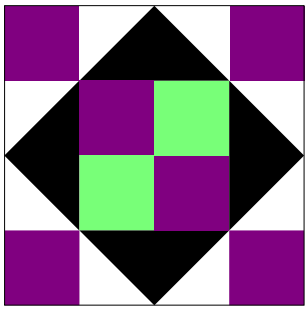
A)



B)

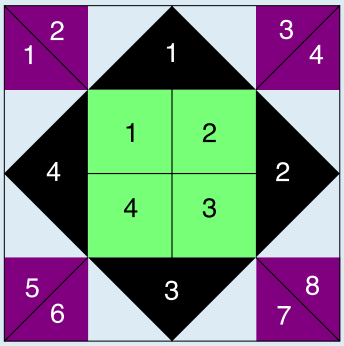


C)



D)

Antwort A ist richtig:
Zunächst muss man herausfinden, ob die Fliesen der drei Sorten so gelegt werden können, dass sie das jeweilige Rangoli ergeben. Anschließend ist zu prüfen, dass für jede Sorte die Anzahl der genutzten Fliesen nicht größer ist als die Zahl der verfügbaren Fliesen.
Das Bild zeigt eine Möglichkeit, wie Rangoli A gelegt werden kann. Die folgende Tabelle gibt an, wie viele Fliesen von jeder Sorte für die verschiedenen Rangoli benötigt werden und ob Indu genug Fliesen hat, um das Muster zu legen.



Rangoli	Benötigte Fliesen	Kann Indu das Rangoli legen?
A	8 lila Dreiecke 4 grüne Quadrate 4 schwarze Dreiecke	Ja, sie hat genug Fliesen von jeder Sorte.
B	12 lila Dreiecke 4 grüne Quadrate 6 schwarze Dreiecke	Nein, sie hat nur 8 lila Dreiecke.
C	4 lila Dreiecke 8 grüne Quadrate 4 schwarze Dreiecke	Nein, sie hat nur 4 grüne Quadrate.
D	12 lila Dreiecke 2 grüne Quadrate 4 schwarze Dreiecke	Nein, sie hat nur 8 lila Dreiecke.

Indu kann also nur Rangoli A mit ihren Fliesen legen.

Das ist Informatik!

Rangoli ist eine Kunstform, die in Indien traditionell aus gefärbtem Reis und Mehl, aber auch aus farbigem Sand oder Blüten erstellt wird. Rangoli haben vor allem dekorative Zwecke, sind aber auch mit regionalen, familiären oder auch religiösen Traditionen verbunden.

In dieser Biberaufgabe muss man komplexe Formen in kleinere Formen zerlegen, die man dann mit den vorhandenen Grundformen abgleichen kann. So kann man erkennen, wie viele von den Grundformen jeweils benötigt werden. Die Zerlegung von Strukturen in Unterstrukturen heißt auch Dekomposition – und wird in der Informatik zum Beispiel angewandt, wenn die gewünschte Funktionsweise eines Programms in Teilfunktionen zerlegt wird, die man als überschaubare und leicht zu testende Unterprogramme implementieren kann.

Dekomposition wird als ein wichtiger Bestandteil des „digitalen Denkens“ (engl.: computational thinking) angesehen, dass nicht nur für Informatikerinnen und Informatiker wichtig ist, sondern allgemein beim Lösen von Problemen hilfreich sein kann.

Rauchzeichen

Der Wetter-Biber ist oben auf dem Berg und sendet Rauchzeichen zu den Bibern im Tal.
Die Rauchzeichen bestehen aus fünf großen oder kleinen Rauchwolken.
Damit sendet der Wetter-Biber vier verschiedene Nachrichten:

			
stürmisch	regnerisch	bewölkt	sonnig

Eines Tages sehen die Biber im Tal dieses Rauchzeichen.

Das ging schief. Der Wetter-Biber auf dem Berg hat einen Fehler gemacht. Eine der fünf Rauchwolken hat die falsche Größe.

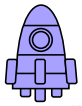
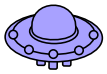
Welche Nachricht wollte der Wetter-Biber senden?

A) stürmisch **B) regnerisch** **C) bewölkt** **D) sonnig**

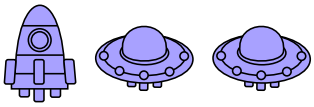
Antwort C ist richtig:
Wenn man die dritte Rauchwolke von einer großen zu einer kleinen ändert, erhält man das Rauchzeichen für „bewölkt“.
Antwort A ist falsch, da die erste und die letzte Rauchwolke geändert werden müssten, um eines der vier Rauchzeichen (für „stürmisch“) zu ergeben. Antwort B ist ebenfalls falsch, da die erste, zweite und vierte Rauchwolke geändert werden müssten. Antwort C ist ebenfalls falsch, da alle Rauchwolken außer der ersten geändert werden müssten.

Das ist Informatik!
Wenn eine Folge von Symbolen zur Kommunikation genutzt wird (von Menschen oder Computern), ist es besser, die Folge so zu wählen, dass man die gesendeten Informationen rekonstruieren kann, selbst wenn einzelne Teile der Folge fehlen oder beschädigt sind. Dies kann man erreichen, indem man mehr Informationen als nötig sendet. Die Information wird dadurch redundant. In der Informatik wird dies täglich genutzt, zum Beispiel beim Senden von Musik. Auf diesem Weg kann die Musik richtig abgespielt werden, selbst wenn die gesendeten Daten teilweise beschädigt sind.
Für den Wetter-Biber hätten Rauchzeichen mit zwei Rauchwolken ausgereicht, um die vier Wettervorhersagen zu unterscheiden:  .
In dieser Biberaufgabe benutzen wir für das Senden einer Nachricht insgesamt fünf Rauchwolken. Deshalb können wir die Nachricht verstehen, selbst wenn eine, zwei oder in manchen Fällen auch drei Wolken falsch gesendet wurden.

Raumfahrt

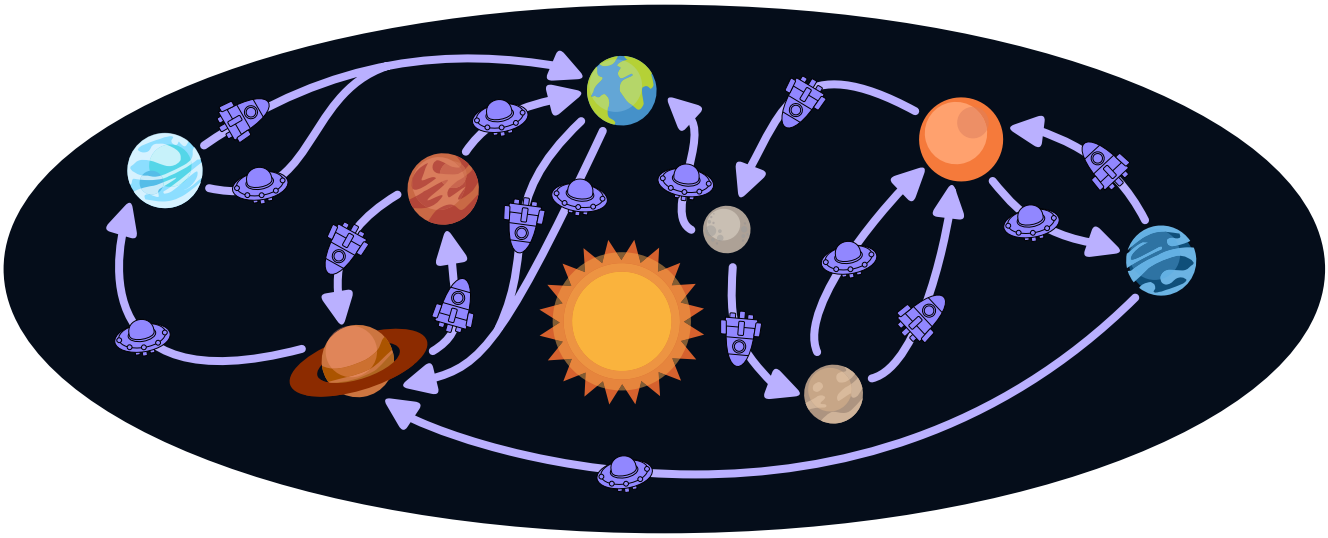
Im All fliegen Astronauten mit Rakete () oder Raumschiff () von Planet zu Planet. Das große Bild unten zeigt die Flugmöglichkeiten.
Für eine längere Reise benötigt man einen Reiseplan.

Ein Beispiel: Ein Astronaut möchte von der Venus () zum Saturn () reisen.

Er erhält diesen Reiseplan: 

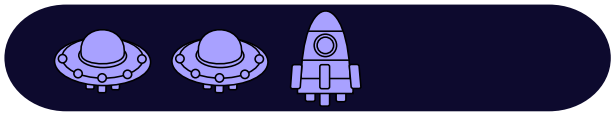
Das bedeutet: Zunächst fliegt der Austronaut mit der Rakete zum Jupiter ().

Weiter geht es mit dem Raumschiff zum Neptun () und von dort mit dem Raumschiff zum Saturn (). Angekommen!

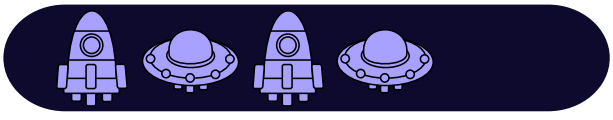


Astronautin Tine ist auf dem Neptun () und möchte zurück zur Erde () reisen.

Mit welchem Reiseplan kommt Tine NICHT auf der Erde an?

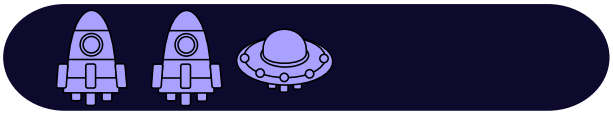


A)



B)

C)



D)

Antwort B ist richtig:

Nur mit Reiseplan B kommt Tine nicht auf der Erde an. Mit diesem Plan fliegt Tine mit der Rakete zum Jupiter, von dort mit dem Raumschiff zurück zum Neptun und dann noch einmal mit Rakete und Raumschiff über Jupiter zurück zum Neptun.

Mit allen anderen Reiseplänen kommt Tine auf der Erde an:
Mit Reiseplan A fliegt Tine mit dem Raumschiff zum Saturn, weiter mit dem Raumschiff zum Uranus und von dort mit der Rakete zur Erde.

Mit Reiseplan C fliegt Tine mit der Rakete zum Jupiter, dann mit dem Raumschiff zurück zum Neptun sowie weiter zu Saturn und Uranus und schließlich wieder mit der Rakete zur Erde.

Mit Reiseplan D fliegt Tine mit der Rakete zum Jupiter, wieder mit der Rakete zum Merkur und schließlich mit dem Raumschiff zur Erde.

Das ist Informatik!

Der große Verbindungsplan gibt für jeden Planeten eindeutige Flugziele für Rakete und Raumschiff an. Es ist also immer klar, wohin es vom aktuellen Planeten aus weitergeht, wenn Rakete bzw. Raumschiff als Nächstes auf dem Reiseplan stehen. Sind Start und Ziel bekannt (wie Neptun und Erde in dieser Biberaufgabe), kann mit Hilfe des Verbindungsplans immer entschieden werden, ob ein Reiseplan vom Start zum Ziel führt.

Ein solches Entscheidungsmodell mit Eingabezeichen (hier: Rakete und Raumschiff), Zuständen (die Planeten), eindeutigen Zustandsübergängen (der Verbindungsplan) sowie einem Startzustand (Neptun) und einem oder mehreren Endzuständen (hier nur einer: die Erde) ist in der Informatik als „deterministischer endlicher Automat“ (kurz: DEA) bekannt. Ein DEA kann zum Beispiel verwendet werden, um das Verhalten programmierter Maschinen wie etwa Geld- oder Fahrkartenautomaten oder die Verarbeitung von Eingaben durch ein Computerprogramm zu beschreiben. Außerdem kann man mit DEAs oder den damit gleichwertigen regulären Ausdrücken Vorgaben für Zeichenfolgen beschreiben, z.B. das Format einer E-Mail-Adresse.

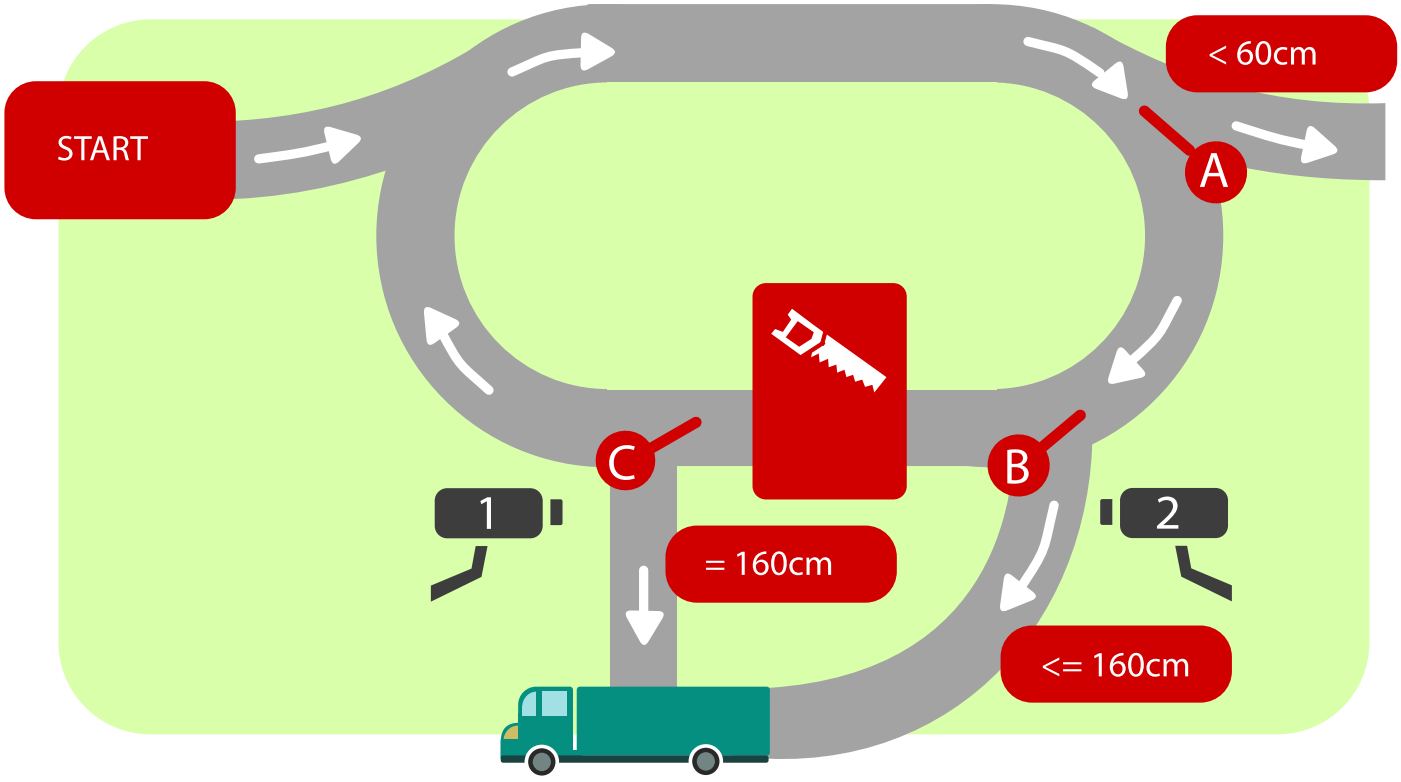
Entscheidungsprobleme, die mit DEAs beschrieben werden können – wie also die Entscheidung, ob eine Zeichenfolge eine gültige E-Mail-Adresse ist – sind einfach. Jede einzelne Entscheidung kann in sehr kurzer Zeit getroffen werden; die Dauer hängt nur von der Länge der Eingabe ab.

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

Sägewerk

Das Sägewerk verarbeitet Holzstücke.
Verladen und ausgeliefert werden nur Stücke zwischen 60 cm und 160 cm Länge.
Das funktioniert so wie im Bild gezeigt:

Bei **START** werden die Stücke angeliefert.
Bei **Weiche A** werden Stücke kürzer als 60 cm aussortiert.
Bei **Weiche B** gehen Stücke bis zu 160 cm Länge zur Verladung.
Die **Säge** schneidet ein 160 cm langes Stück ab.
Bei **Weiche C** gehen 160 cm lange Stücke zur Verladung. Andere Stücke bleiben im Sägewerk.
Kamera 1 und **Kamera 2** zählen die Stücke, die an ihnen vorbei kommen.



Früh morgens öffnet das Sägewerk.
Als erstes werden drei Holzstücke neu angeliefert. Sie sind 60 cm, 140 cm and 360 cm lang.
Das Sägewerk verarbeitet diese Stücke komplett, bevor weitere Stücke angeliefert werden.

Wie viele Stücke haben die beiden Kameras bis dahin gezählt?

A) Kamera 1: 3 Stücke, Kamera 2: 1 Stück
B) Kamera 1: 1 Stück, Kamera 2: 3 Stücke
C) Kamera 1: 2 Stück, Kamera 2: 2 Stücke
D) Kamera 1: 4 Stücke, Kamera 2: 0 Stücke

Antwort C ist richtig:
Das Sägewerk verarbeitet die drei Baumstämme so:

- Der 60 cm lange Baumstamm passiert Weiche A. Bei Weiche B geht er zur Verladung. Kamera 2 zählt ein Stück.
- Der 140 cm lange Baumstamm passiert Weiche A. Bei Weiche B geht er zur Verladung. Kamera 2 zählt ein zweites Stück.
- Der 360 cm lange Baumstamm passiert Weiche A und auch Weiche B.
- Die Säge schneidet ein 160 cm langes Stück ab, das restliche Stück ist 200 cm lang.
- Das 160 cm lange Stück geht bei Weiche C zur Verladung. Kamera 1 zählt ein Stück.
- Das 200 cm lange Stück passiert die Weichen A und B.
- Die Säge schneidet ein 160 cm langes Stück ab, das restliche Stück ist 40 cm lang.
- Das 160 cm lange Stück geht bei Weiche C zur Verladung. Kamera 1 zählt ein zweites Stück.
- Das 40 cm lange Stück wird bei Weiche A aussortiert.

Die Kameras 1 und 2 haben je 2 Stämme gezählt.

Das ist Informatik!
Während der Öffnungszeit des Sägewerks in dieser Biberaufgabe werden nach und nach immer mehr Holzstücke angeliefert. Sie mischen sich mit den noch nicht aussortierten und nicht verladenen Stücken zu einem Strom von Holzstücken. Die Maschinen des Sägewerks, also Säge, Weichen und Kameras, reagieren automatisch auf jedes Stück in diesem Strom und verarbeiten es nach ihren Vorschriften. So kann das Sägewerk flexibel mit seiner ständig nachströmenden „Eingabe“ zurechtkommen. Und auch als Ausgabe produziert es einen Strom von Holzstücken, die nach und nach weggefahren werden.

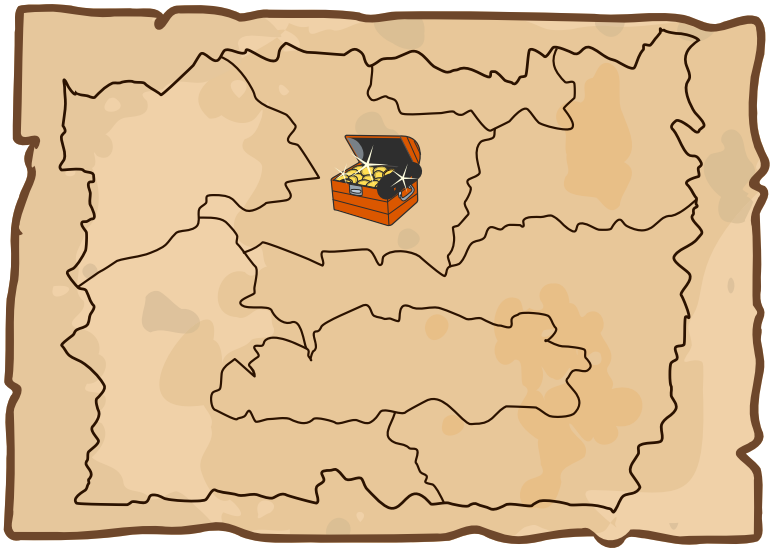
Auch ein Computerprogramm ist – gemeinsam mit dem Computer, auf dem es läuft – wie eine Maschine. Traditionell arbeitet diese nach dem EVA-Prinzip: Zuerst werden alle Eingaben erfasst, dann werden sie verarbeitet, und zuletzt werden Ergebnisse ausgegeben. Bei vielen modernen Computeranwendungen ist dieses Prinzip aber nicht mehr zeitgemäß. Eingaben kommen nach und nach als Datenstrom an, und die Programm/Computer-Maschine (zum Beispiel eine Smartphone-App) muss ständig darauf reagieren. Auch eine Tabellenkalkulation muss auf jede Veränderung einer Tabellenzelle reagieren und die in andere Zellen eingetragenen Formeln neu ausrechnen.

In der Informatik gibt es viele Ideen, wie man mit Datenströmen umgehen und das flexible Reagieren auf neue Daten organisieren kann. Einer der Ansätze heißt – wenig überraschend – „Reaktives Programmieren“.

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

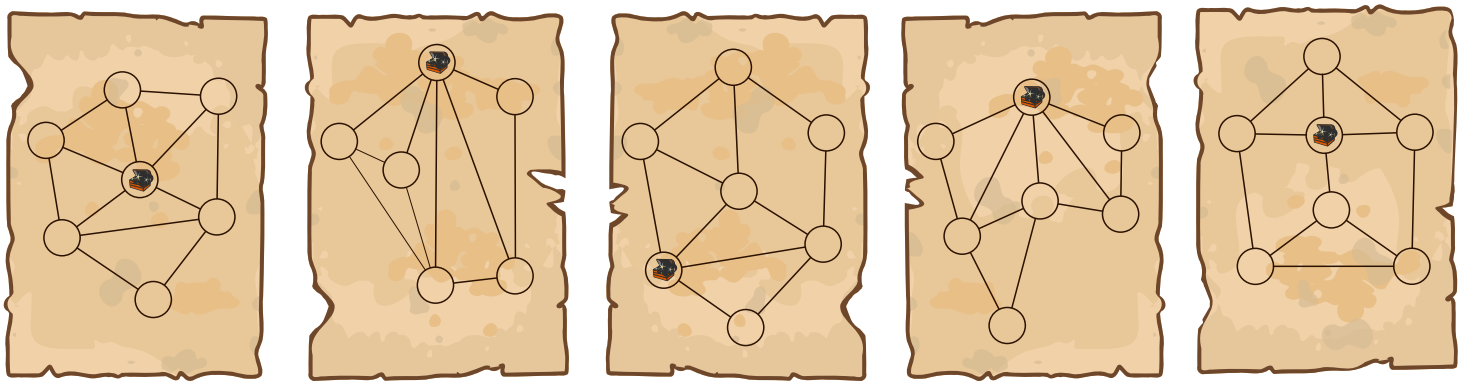
Schatzkarte

Die Landkarte zeigt das Reich des Biberkönigs mit seinen sieben Provinzen.
In einer Provinz hat der König seinen Schatz versteckt.



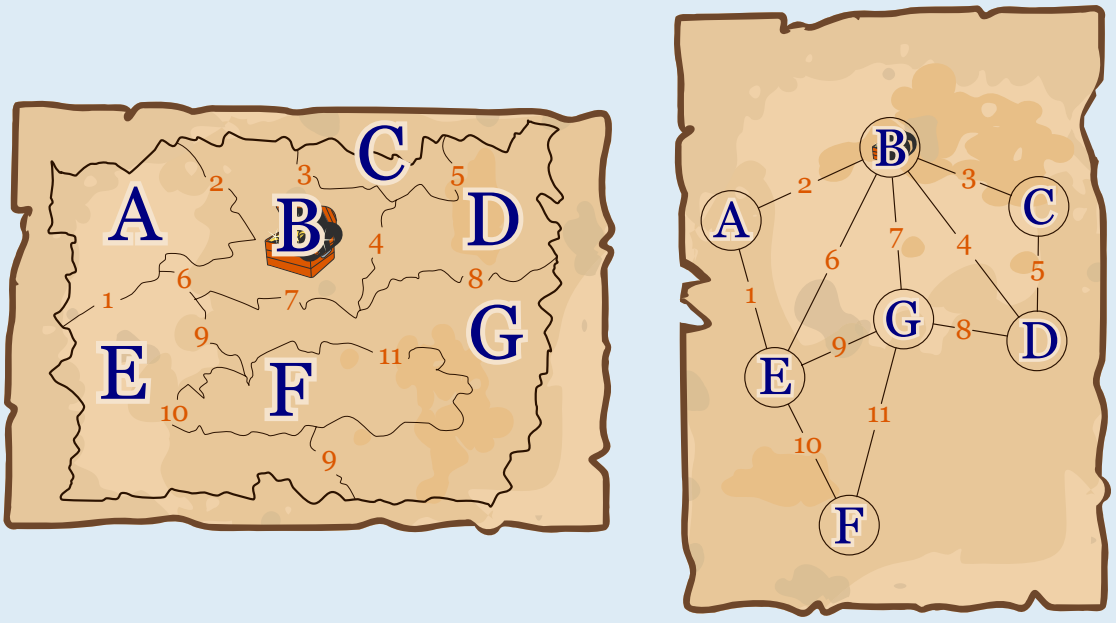
Der König will die Lage des Schatzes geheim halten.
Deshalb hat er eine besondere Schatzkarte gezeichnet.
Für jede Provinz ist darin ein Kreis eingezeichnet.
Eine Linie zwischen zwei Provinz-Kreisen zeigt, dass die beiden Provinzen aneinander angrenzen. Der Kreis für die Provinz mit dem Schatz ist markiert.
Um mögliche Räuber zu verwirren, hat der König zusätzlich vier falsche Schatzkarten gezeichnet.

Welches 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 bezeichnet. Wenn zwei Provinzen aneinander angrenzen, sind ihre gemeinsamen Grenzen (zwischen den Provinzen E und G gibt es zwei) mit einer Zahl bezeichnet; insgesamt mit den Zahlen 1 bis 11. Diese Buchstaben und Zahlen können wir entsprechend in die Schatzkarte D rechts übertragen und so erkennen, dass sie 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.

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

Schiebeparkplatz

Auf dem „Schiebeparkplatz“ in Tübingen darf man Wagen auch quer vor den normalen Plätzen abstellen. Ein quer stehender Wagen, der einen anderen Wagen blockiert, wird vorsichtig vorwärts oder rückwärts verschoben. Dann kann der blockierte Wagen ausfahren.

Das Bild zeigt ein Beispiel:
Wagen A wird nicht blockiert und kann ausfahren.
Wagen L wird blockiert. Wenn Wagen M verschoben wird, kann Wagen L ausfahren.
Im Beispiel wird ein Wagen zweifach blockiert:
Zwei andere Wagen müssen verschoben werden, damit er ausfahren kann.

Welcher Wagen wird zweifach blockiert?

So ist es richtig:
Wagen I wird zweifach blockiert. Wagen N muss verschoben werden, damit Wagen I ausfahren kann. Doch dafür ist nicht genug Platz. Nur wenn zuerst entweder Wagen O nach links oder Wagen M nach rechts verschoben wird, kann anschließend auch Wagen N verschoben werden. Dann kann Wagen I ausfahren.
Kein anderer Wagen wird zweifach blockiert: Die Wagen A, D, E, J und Q können direkt ausfahren.
Bei den Wagen B, C, F, G, H, K und L genügt es jeweils, einen Wagen zu verschieben, damit sie ausfahren können: Für die Wagen B und C muss Wagen P verschoben werden, für F und G muss O, für H muss N und für K und L muss M verschoben werden.

Das ist Informatik!
Autos verbrauchen Platz, und Platz ist kostbar. Da klingt es nach einer guten Idee, auch Parkplatz so gut wie möglich auszunutzen – wie auf dem Schiebeparkplatz in Tübingen (s. Foto).

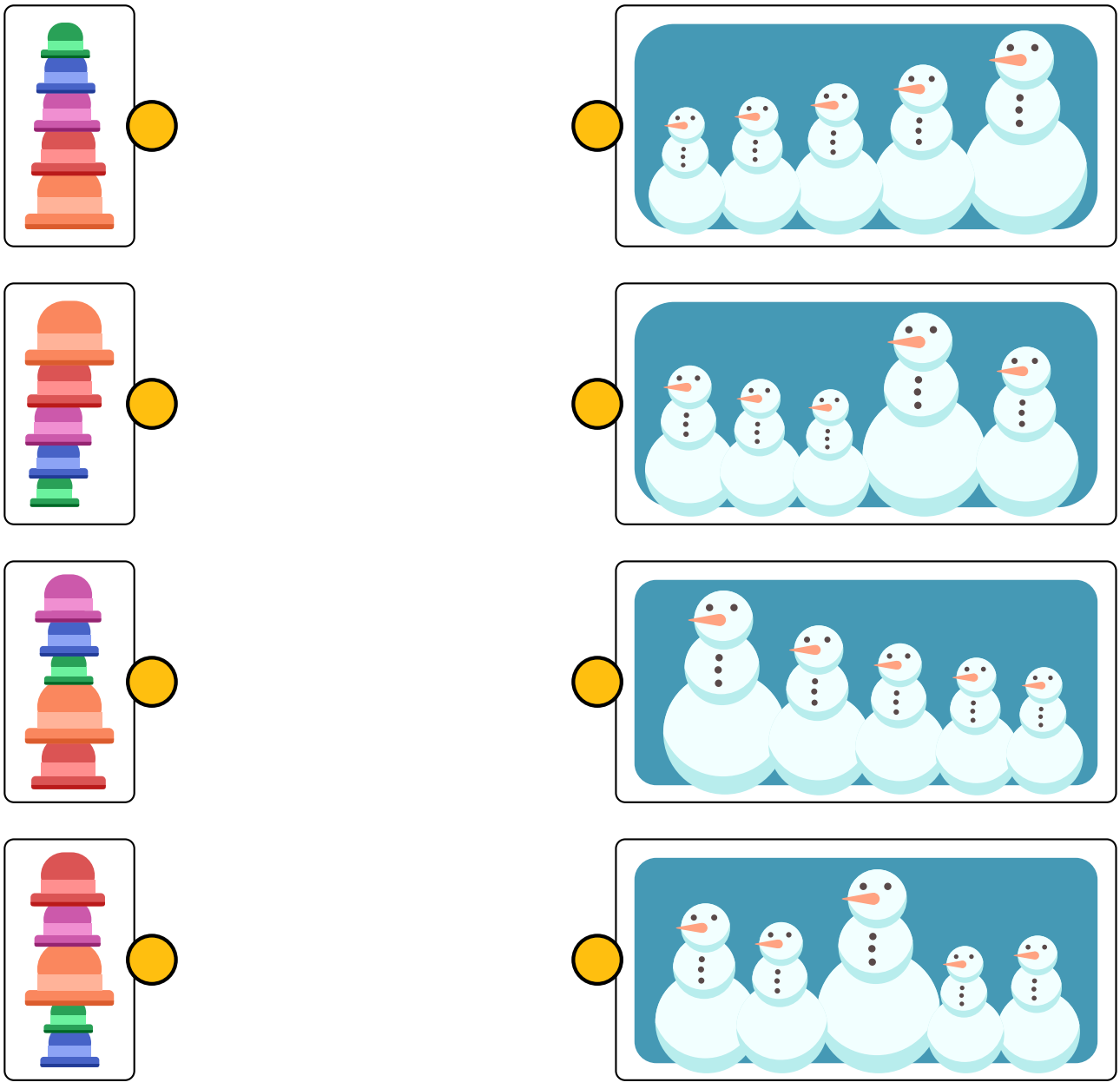
In der Informatik wird seit einiger Zeit daran gearbeitet, dass Autos autonom fahren können, also selbstständig, ohne Eingreifen einer Fahrerin. In einem Parkhaus in Stuttgart (nicht weit von Tübingen) funktioniert Parken ohne Fahrer schon jetzt. Darin fahren die Wagen aber nicht autonom, sondern werden ferngesteuert. Immerhin können sie so viel enger nebeneinander abgestellt werden, als das mit menschlichen Fahrern möglich wäre. Und auch quer parken ist denkbar, solange genügend Platz zum Verschieben bleibt – wie in dieser Biberaufgabe.

https://www.sueddeutsche.de/auto/daimler-bosch-autonomes-parken-1.4536231
https://de.wikipedia.org/wiki/Autonomes_Fahren

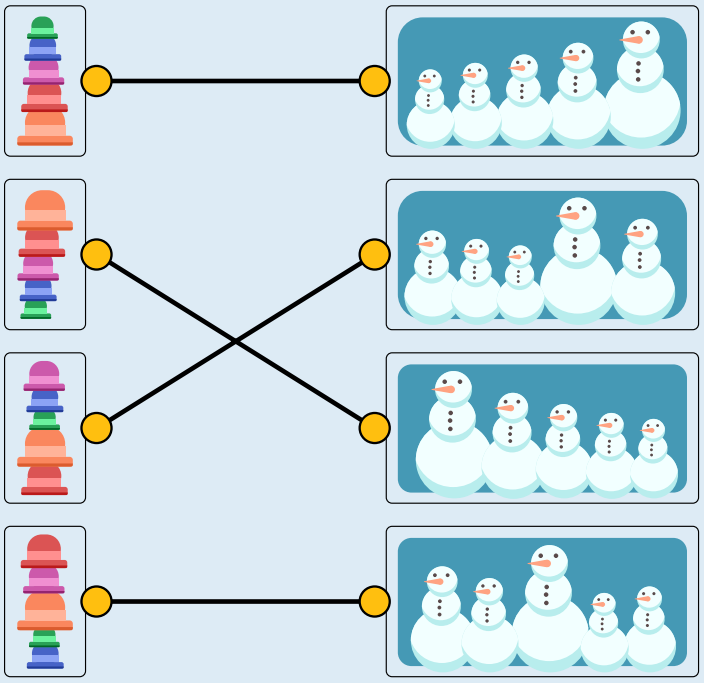
Schneemann-Hüte

Die Schneemänner bekommen Hüte. Sie stellen sich in vier Reihen an.
Jede Reihe sucht sich den passenden Hutstapel aus.
Dann nimmt der erste Schneemann (links) den obersten Hut vom Stapel, der zweite den nächsten Hut – und so weiter.
Am Ende hat jeder den passenden Hut:
Der kleinste Schneemann hat den kleinsten Hut, der zweitkleinste Schneemann den zweitkleinsten Hut – und so weiter.

Verbinde jede Schneemann-Reihe mit dem passenden Hutstapel.



So ist es richtig:
Der oberste Hutstapel passt zur obersten Schneemannreihe. In der Reihe stehen die Schneemänner nach Größe geordnet, der kleinste zuerst.



Auf dem Stapel liegen die Hüte nach Größe geordnet, der kleinste zuoberst.
Der zweite Hutstapel passt zur dritten Schneemannreihe. In der Reihe stehen die Schneemänner nach Größe geordnet, der größte zuerst. Auf dem Stapel liegen die Hüte nach Größe geordnet, der größte zuoberst.
Der dritte Hutstapel passt zur zweiten Schneemannreihe.

Um das zu erkennen, ordnen wir Schneemännern und Hüten Nummern zu, der Größe nach: Der größte Schneemann bzw. der größte Hut bekommt die Nummer 5, der zweitgrößte Schneemann bzw. Hut bekommt Nummer 4, und so weiter.

Die Schneemänner in der zweiten Reihe haben dann von vorne die Nummern 3-2-1-5-4. Die Hüte auf dem dritten Stapel haben von oben die Nummern 3-2-1-5-4 – das passt.
Die Schneemänner in der vierten Reihe haben von vorne die Nummern 4-3-5-1-2. Dazu passt der vierte Hutstapel: Darin haben die Hüte von oben auch die Nummern 4-3-5-1-2.

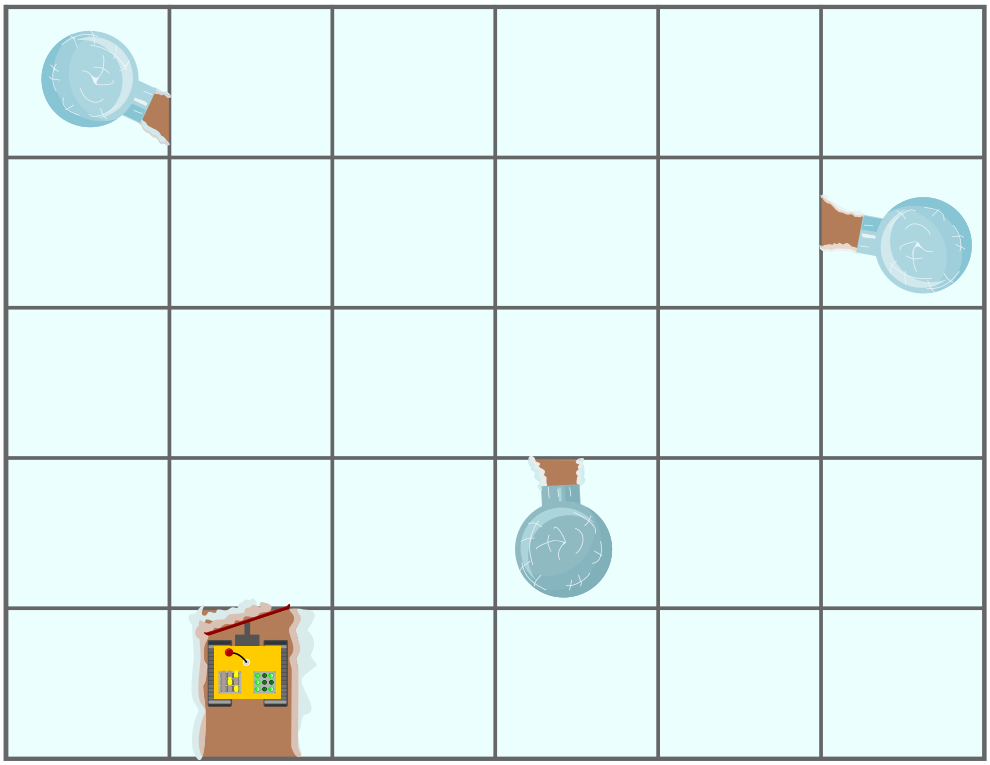
Das ist Informatik!

Kenner früherer Biberaufgaben vermuten vielleicht, was jetzt kommt. Allzu offensichtlich sind die Bezüge zu den Datenstrukturen Stapel und Reihe bzw. Schlange bei Hüten und Schneemännern. Doch die Erklärung der Lösung legt ein anderes, viel grundsätzlicheres Informatikthema nahe: die Abstraktion – die bei der Aufgabe „Schatzkarte“ schon kurz angesprochen wurde. Um bei den etwas komplizierteren Fällen erläutern zu können, welcher Hutstapel zu welcher Schneemann-Reihe passt, haben wir aufgehört, über Hüte und Schneemänner zu reden. Stattdessen haben wir Nummern verwendet, welche die Größe von sowohl Hüten als auch Schneemännern beschreiben. So konnten die Abfolgen der Größen jeweils aufgelistet und die Größen-Listen miteinander verglichen werden. Alle anderen Eigenschaften von Hüten (wie Farben oder Form) und Schneemännern (etwa die Anzahl der Knöpfe auf der mittleren Kugel) können in dieser Biberaufgabe ignoriert werden.

Die Darstellung von Hutstapeln und Schneemannreihen als Größenlisten ist eine Abstraktion oder Reduktion auf die für das gegebene Problem relevanten Informationen. Aber sollen Hüte und Schneemänner immer nur bezüglich ihrer Größe betrachtet werden? Vielleicht erfordert das nächste Hut-Schneemann-Problem ganz andere Informationen? Informatiksysteme arbeiten immer mit Abstraktion: mit einer Abbildung der Realität auf die Möglichkeiten binärer Repräsentation und algorithmischer Prozesse. Bei Abstraktion geht Information verloren. Das ist nicht grundsätzlich schlecht, aber jede Informatikerin und jeder Informatiker sollte sich dessen bewusst sein.

Schneepflug-Roboter

Nach einem Schneesturm sind drei Iglus von der Hauptstraße abgeschnitten.
Ein Schneepflug-Roboter soll einen Weg zu allen drei Iglus freiräumen. Anschließend soll er zu seiner Startposition zurückkehren. Die interne Landkarte des Roboters ist in Quadrate eingeteilt. Von einem Quadrat kann der Roboter immer nur zum nächsten waagerecht oder senkrecht benachbarten Quadrat fahren.



Der Roboter benötigt:

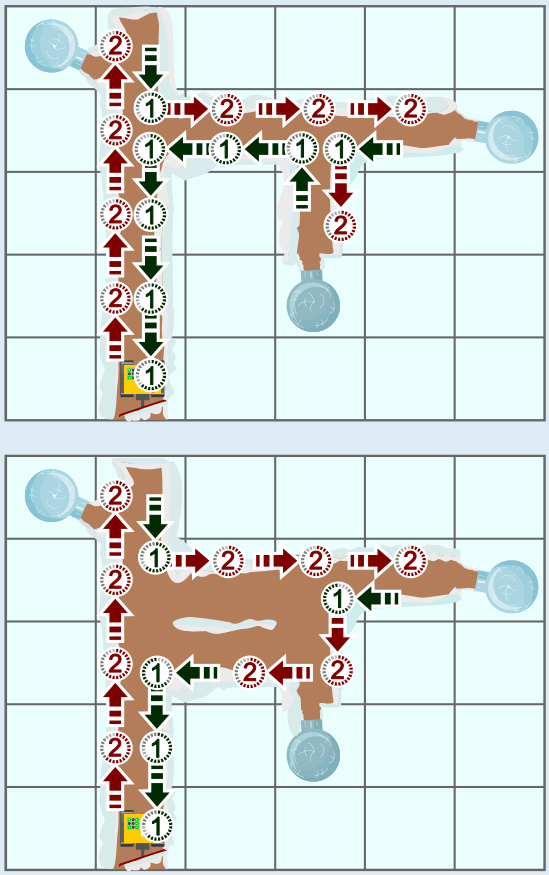
2 Minuten für die Fahrt zum nächsten Quadrat, wenn dort noch Schnee liegt.
1 Minute für die Fahrt zum nächsten Quadrat, wenn dieses bereits geräumt ist.
0 Minuten, um sich auf einem Quadrat zu drehen.

Der Roboter muss nicht auf die Quadrate mit den Iglus fahren. Es genügt, einen Weg bis zu den Eingängen zu räumen.

Sein Steuercomputer lässt den Roboter so fahren, dass er einen Weg zu allen drei Iglus räumt und dafür so wenig Zeit wie möglich benötigt.

Wie viele Minuten dauert die Fahrt des Roboters insgesamt?

23 ist die richtige Antwort:
Da es weniger Zeit benötigt, ein Quadrat erneut zu befahren, welches bereits geräumt wurde, ist es sinnvoll, bereits geräumte Wege so häufig wie möglich wiederzuverwenden. Nach diesem Grundsatz können zwei Routen berechnet werden, eine davon ist unten zu sehen. Das Bild zeigt, wie viele Minuten der Roboter für die Fahrten zu den nächsten Quadraten jeweils benötigt. Die Summe der Zahlen ist 24, der Roboter benötigt also 24 Minuten auf diesem Weg. Aber ist das schon der schnellste Weg?



Bislang haben wir alle geräumten Quadrate wiederverwendet, um Zeit zu sparen. Nun können wir noch nach Abkürzungen suchen, mit denen wir noch mehr Zeit sparen können. Und wirklich: Auf dem Weg vom letzten Iglu zur Startposition fährt der Roboter einen Umweg über bereits geräumte Quadrate. Wenn er den abkürzt, muss er zwar ein weiteres Quadrat räumen, benötigt dafür aber nur 2 Minuten statt 3 Minuten für den Umweg (unteres Bild). Auf diesem Weg dauert die Fahrt des Roboters also insgesamt nur 23 Minuten. Schneller geht es nicht, denn es gibt keine weitere Abkürzung.

Das ist Informatik!
In dieser Biberaufgabe wird nach einem Wegenetz gesucht, das alle Orte (die Iglus und die Startposition des Roboters) mit minimalen Kosten (die Zeit, die die Fahrt des Roboters dauert) verbindet. Solch ein Wegenetz enthält nicht unbedingt die kürzesten Wege zwischen allen Orten, aber die Kosten, um es zu erstellen, sind so klein wie möglich. Solche Wegenetze nennt man „Steinerbäume“. Sie werden zum Beispiel bei der Konstruktion von Computerplatinen oder beim Bau von wenig genutzten Eisenbahnnetzen für Güter erstellt.
Für eine Menge von Orten und ein Kostenmaß (häufig die geometrische Entfernung) einen Steinerbaum zu finden ist eines der schweren, zur Klasse der „NP-vollständigen Probleme“ gehörenden Optimierungsprobleme – so wie „Set Cover“ aus der Aufgabe „Leckeres Holz“ weiter vorne in diesem Biberheft. Dass hier die Kosten für einen Weg noch abhängig von der Wahl anderer Wege sein können, macht das Problem sogar noch schwerer.
https://de.wikipedia.org/wiki/Steinerbaumproblem

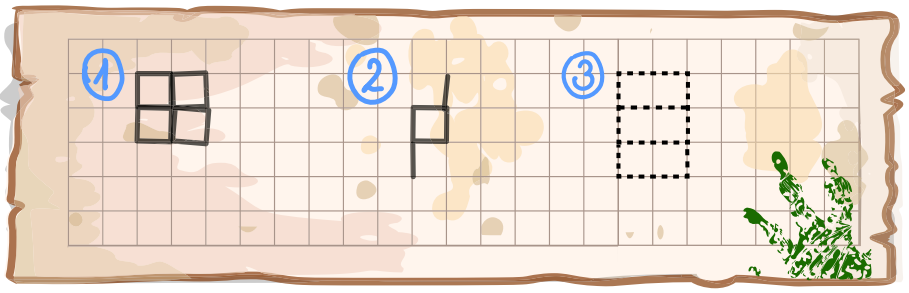
Stempel

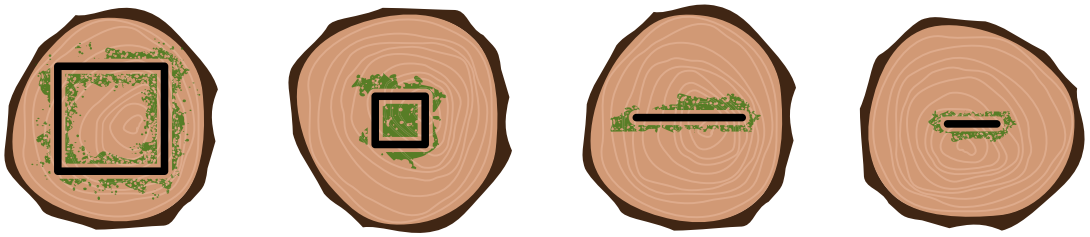
Biber Paul hat vier Stempel: A, B, C und D. Sie sind unten zu sehen.
Damit hat er bereits die Formen 1 und 2 gestempelt.

- Für Figur 1 hat er Stempel B viermal benutzt.
- Für Figur 2 hat er Stempel B einmal und Stempel D zweimal benutzt.

Jetzt will Paul Figur 3 stempeln. Seine Freundin Mia will ihm helfen.
Mia sagt: „Für Figur 3 benutze ich genau einen Stempel zweimal."

Welchen Stempel benutzt Mia?

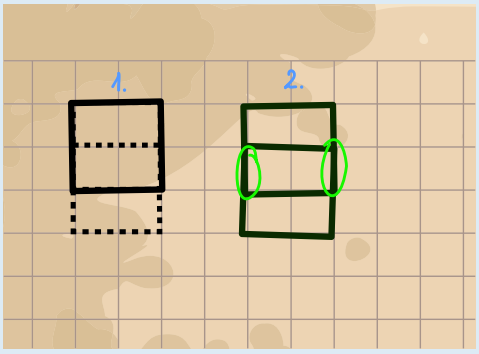




A) **B)** **C)** **D)**

Mia benutzt Stempel A:

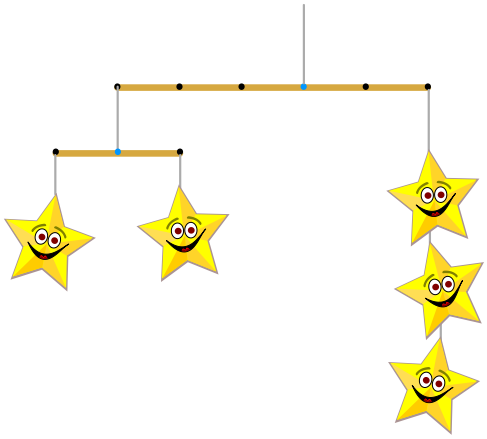
Das ist der Stempel  mit dem großen Quadrat. Im ersten Schritt stempelt sie den oberen Teil. Danach stempelt sie den unteren Teil. Hier überlappt sich die Farbe an zwei Stellen. Die grünen Markierungen zeigen diese Stellen.
Mia könnte Figur 3 auch mit den Stempeln C und D stempeln. Doch Stempel C müsste sie achtmal und Stempel D sogar vierzehnmal benutzen.
Mit Stempel B kann Figur 3 nicht gestempelt werden.



Das ist Informatik!
Ein wunderschönes Bild, doch wie hat der Maler es nur geschafft, dass die Farben so glänzen? Eine alte Geige, aber warum klingt sie so besonders? Die meisten Menschen möchten gerne verstehen, wie Dinge funktionieren. Das wird schwierig, wenn es niemanden (mehr) gibt, der das erklären kann oder will. In der Informatik stellen sich ähnliche Fragen: Ein schönes altes Computerspiel, aber der Programmcode ist nicht mehr aufzutreiben: Wie kann das Spiel auf moderne Computer übertragen werden? Eine Schadsoftware ist aufgetaucht, deren Entwickler unbekannt sind: Wie kann man Computer dagegen schützen? Da muss man sich selber helfen und auf geschickte Weise versuchen, aus dem Maschinencode von Spiel oder Schadsoftware auf deren Programmcode zu schließen. Wenn man Methoden anwendet, um die Funktionsweise eines (technischen) Konstruktes zu verstehen, wird von „Reverse Engineering" gesprochen. So einfach wie beim Stempelbild in dieser Biberaufgabe geht Reverse Engineering aber in den allermeisten Fällen nicht. Deshalb sind erfahrene Informatikerinnen und Informatiker gefragt, die auf verschiedene in der Informatik entwickelte Werkzeuge zurückgreifen.

Stern-Mobiles

Stern-Mobiles sind kunstvolle 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.

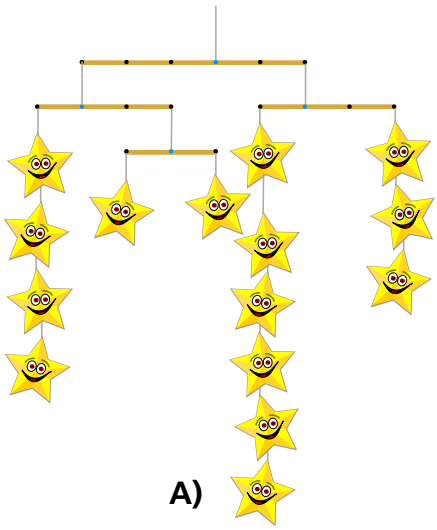


Das Bild zeigt ein einfaches Stern-Mobile.
Mit Zahlen und Klammern kann man es so beschreiben:
(-3 (-1 1) (1 1)) (2 3)

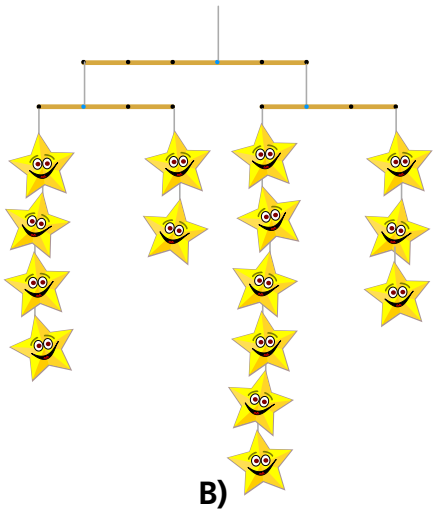
Die Zahlen geben jeweils an:
entweder den Abstand eines Stab-Endes zum Faden,
an dem der Stab hängt, oder eine Anzahl an Sternen.

Die Klammern geben die Struktur des Stern-Mobiles an.

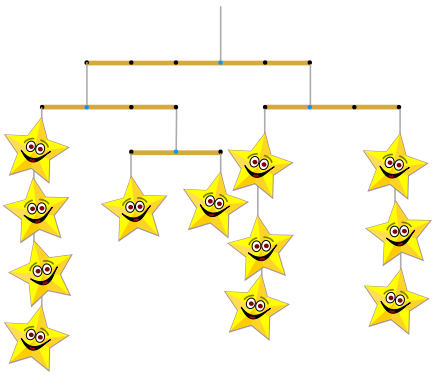
Welches der folgenden Stern-Mobiles kann man so beschreiben:
(-3 (-1 4) (2 (-1 1) (1 1))) (2 (-1 6) (2 3))



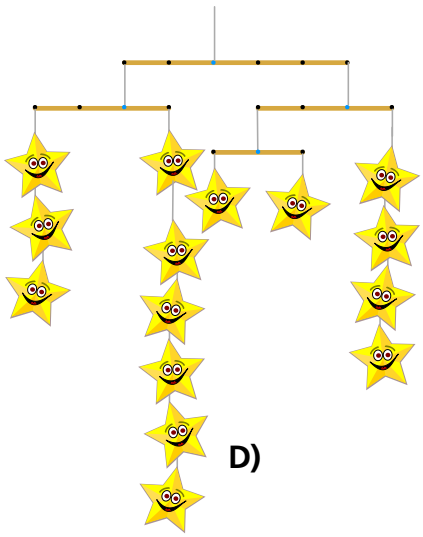
A)



B)



C)



D)

A ist die richtige Antwort.
Aus dem Beispiel und seiner Beschreibung kann man Folgendes erkennen:
- Ein Klammerpaar mit zwei Zahlen (A S) beschreibt einen Faden mit Sternen: A ist der Abstand zum Aufhängefaden des Stabs, an dem der Faden mit Sternen hängt. S ist die Anzahl der Sterne.
  Der rechte Teil der Beispielbeschreibung (2 3) bedeutet also, dass mit Abstand 2 nach rechts vom Aufhängefaden ein einfaches Stern-Mobile mit 3 Sternen hängt.
- Alle anderen Klammerpaare haben drei Bestandteile: (A M1 M2). A gibt wie oben den Abstand zum Aufhängefaden des Stabs an, an dem ein Stern-Mobile aufgehängt ist. M1 und M2 beschreiben die Teil-Mobiles, die am Stab des Stern-Mobiles aufgehängt sind.
  Der linke Teil der Beispielbeschreibung (-3 (-1 1) (1 1)) bedeutet also, dass mit Abstand 3 vom Aufhängefaden (und zwar nach links, deshalb -3) ein Stern-Mobile hängt, an dessen Stab wiederum zwei einfache Stern-Mobiles hängen.
- Die Klammerpaare in der Beschreibung eines Stern-Mobiles sind von links nach rechts so angeordnet wie die Teil-Mobiles, die am Stab des Stern-Mobiles hängen.

Die Beschreibung aus der Frage
(-3 (-1 4) (2 (-1 1) (1 1))) (2 (-1 6) (2 3))
bedeutet also:
- 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 Teil-Mobile 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 Mobile A.

Bei Mobile B hat das linke Teil-Mobile selbst kein Teil-Mobile.
Bei Mobile C gibt es keinen Faden mit 6 Sternen.
Bei Mobile 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:

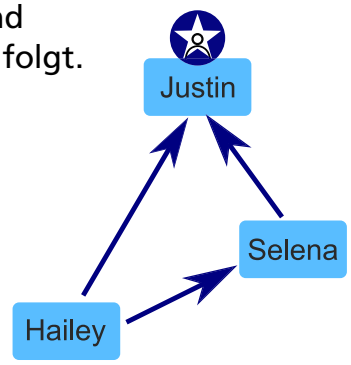
entweder (a) ein Faden mit einigen Sternen
oder (b) ein Faden mit einem Stab, an dessen Enden je ein Stern-Mobile hängt.

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.

Superstar

Im sozialen Netzwerk „TeeniGram“ können Mitglieder anderen Mitgliedern „folgen“. In TeeniGram gibt es außerdem Mitglieder-Gruppen. In einer Gruppe nennt man ein Mitglied „Superstar“, wenn • jedes andere Mitglied der Gruppe ihm folgt und
• es selbst keinem anderen Mitglied der Gruppe folgt.

Ein Beispiel: Eine TeeniGram-Gruppe hat drei Mitglieder, nämlich Hailey, Selena und Justin.
• Hailey folgt Justin und Selena.
• Selena folgt Justin.
• Justin folgt keinem Mitglied der Gruppe.
Justin ist ein Superstar in dieser Gruppe.

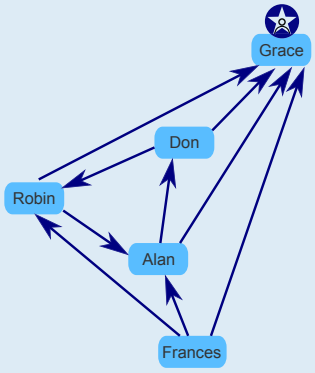


Eine andere TeeniGram-Gruppe hat diese Mitglieder: Alan, Don, Frances, Grace und Robin.
• Alan folgt Don und Grace.
• Don folgt Grace und Robin.
• Frances folgt Alan, Grace und Robin.
• Robin folgt Alan und Grace.

Gibt es einen Superstar in dieser Gruppe?

A) Ja, Alan ist ein Superstar in dieser Gruppe.
B) Ja, Frances und Robin sind Superstars in dieser Gruppe.
C) Ja, Grace ist ein Superstar in dieser Gruppe.
D) Nein, es gibt keinen Superstar in dieser Gruppe.

Antwort C ist richtig:
Grace ist ein Superstar in dieser Gruppe. Sie erfüllt beide Bedingungen: Jedes andere Mitglied der Gruppe (also Alan, Don, Frances und Robin) folgt ihr, und sie selbst folgt keinem Mitglied der Gruppe.
A: ist falsch, weil Alan zwei anderen Mitgliedern folgt (Don und Grace)
B: ist falsch, weil sowohl Frances als auch Robin anderen Mitgliedern folgen. Außerdem ergibt sich aus den Superstar-Bedingungen, dass es höchstens einen Superstar in einer Gruppe geben kann.
D: ist falsch, weil C richtig ist und Grace ein Superstar in dieser Gruppe ist.



Das ist Informatik!
Ein soziales Netzwerk wie TeeniGram entsteht durch die Beziehungen zwischen seinen Mitgliedern. Es gibt unterschiedliche Arten solcher Beziehungen: Wenn Mitglied A einem anderen Mitglied B „folgt“, muss B nicht auch A folgen; eine solche Beziehung nennt man auch gerichtet. Wenn die beiden Mitglieder A und B aber „Freunde“ sind, ist das gegenseitig oder ungerichtet. Beide Arten von Beziehungen können mit Hilfe von Graphen modelliert werden. Für die Auswertung und Untersuchung solcher Beziehungsmodelle kennt die Informatik viele gute Algorithmen.
Echte soziale Netzwerke bilden sehr große Graphen. Die Unternehmen, die diese Netzwerke realisieren, sind sehr daran interessiert, Strukturen in ihren Beziehungs-Graphen zu erkennen oder zu entdecken – wie etwa den Superstar in einer (möglicherweise sehr großen) Gruppe zu bestimmen. Auch wenn es in einem sozialen Netzwerk keine Gruppen gibt, kann es sich lohnen, Mitglieder zu finden, denen viele andere Mitglieder folgen, die aber selbst nur wenigen Mitgliedern folgen. Solche Mitglieder können als Meinungsführer oder „Influencer“ wirken.

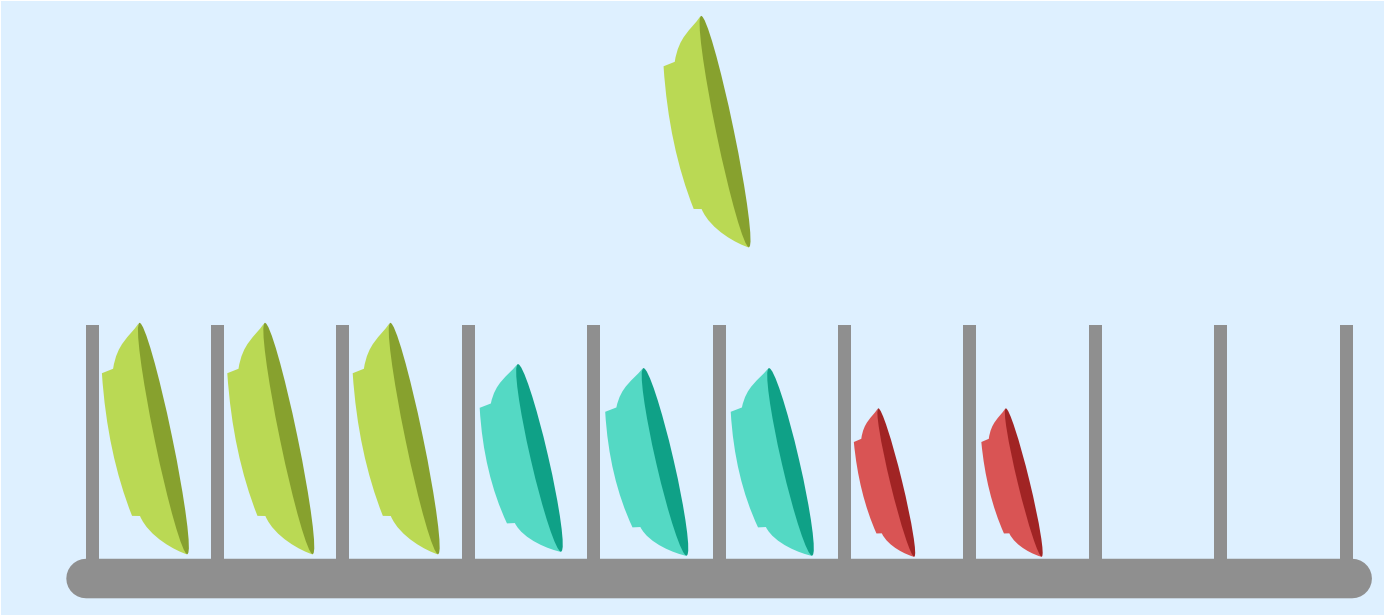
https://en.wikipedia.org/wiki/Social_network_analysis

Teller-Ordnung

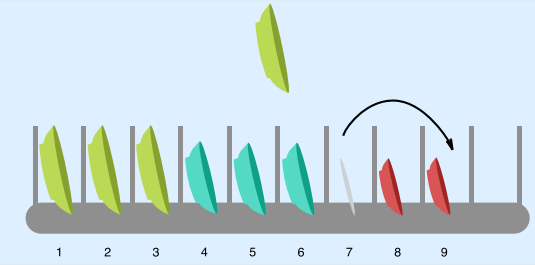
In der Spülmaschine herrscht Ordnung!
Die Teller werden immer so in die Maschine einsortiert (von links):
Zuerst die großen Teller, dann die mittleren und schließlich die kleinen Teller.
Es gibt keine Lücken zwischen den Tellern.

Nach dem Abendessen soll ein weiterer großer Teller in die Spülmaschine einsortiert werden.
Damit danach wieder Ordnung herrscht, müssen einige andere Teller umgestellt werden.

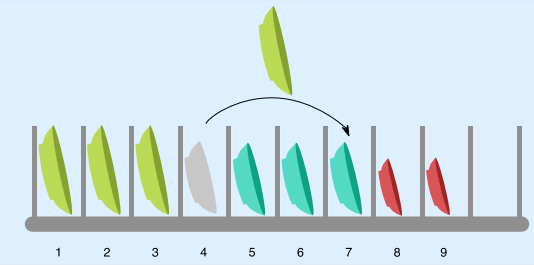
Sortiere den Teller richtig ein.
Stelle dazu so wenige Teller um wie möglich.



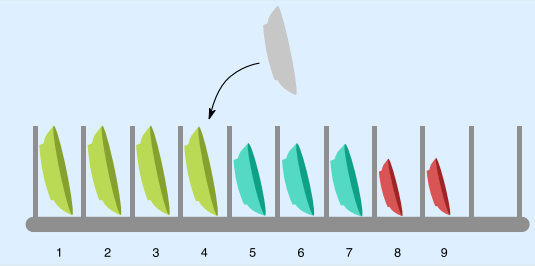
So ist es richtig:
Man kann den Teller einsortieren, indem man zwei Teller umstellt.



Zuerst stellt man den Teller von Position 7 auf Position 9.



Dann stellt man den Teller von Position 4 auf Position 7.



Dann kann man den neuen Teller an Position 4 einsortieren.

Geht es vielleicht mit noch weniger Umstellungen? Es gibt keine Lücke bei den großen Tellern, in die man den neuen großen Teller stellen könnte. Man muss also einen goßen Teller oder den mittleren Teller auf Position 4 umstellen, um die Lücke zu schaffen.
Wenn man diesen Teller auf eine freie Position stellen würde, wären die Teller in der Spülmaschine nicht mehr sortiert. Man muss also mindestens noch einen weiteren Teller umstellen, um für diesen Teller eine Lücke zu schaffen oder die Sortierung wieder herzustellen. Folglich muss man mindestens zwei Teller umstellen; besser geht es nicht.

Das ist Informatik!
Die Ordnung in der Spülmaschine soll, wie man hört, in vielen Haushalten ein beliebtes Thema sein. Darüber, wie wichtig diese Ordnung ist, kann man unterschiedlicher Meinung sein. Dass Ordnung im Computer aber sehr wichtig ist, darüber wurde bei der Biberaufgabe „Kanalsystem“ weiter vorne in diesem Biberheft bereits gesprochen. Und bei der Aufgabe „Anproben“ wird erklärt, wie Computerprogramme in sortierten Daten sehr schnell suchen können. Da Computer ständig in Daten suchen müssen, lohnt es sich, Daten zu sortieren und ihre Sortierung beim Einfügen neuer Daten zu erhalten.

Ein allgemeines Verfahren (also ein Algorithmus), der die Sortierung in der Spülmaschine erhalten kann, lautet so:

- Bewege alle Teller, die kleiner sind als der neue Teller, um eine Position nach rechts.
- Stelle den neuen Teller in die so entstandene Lücke.

Dieses Verfahren würde im Fall dieser Biberaufgabe fünf Teller umstellen, während die vorgestellte Lösung mit zwei Umstellungen auskommt.

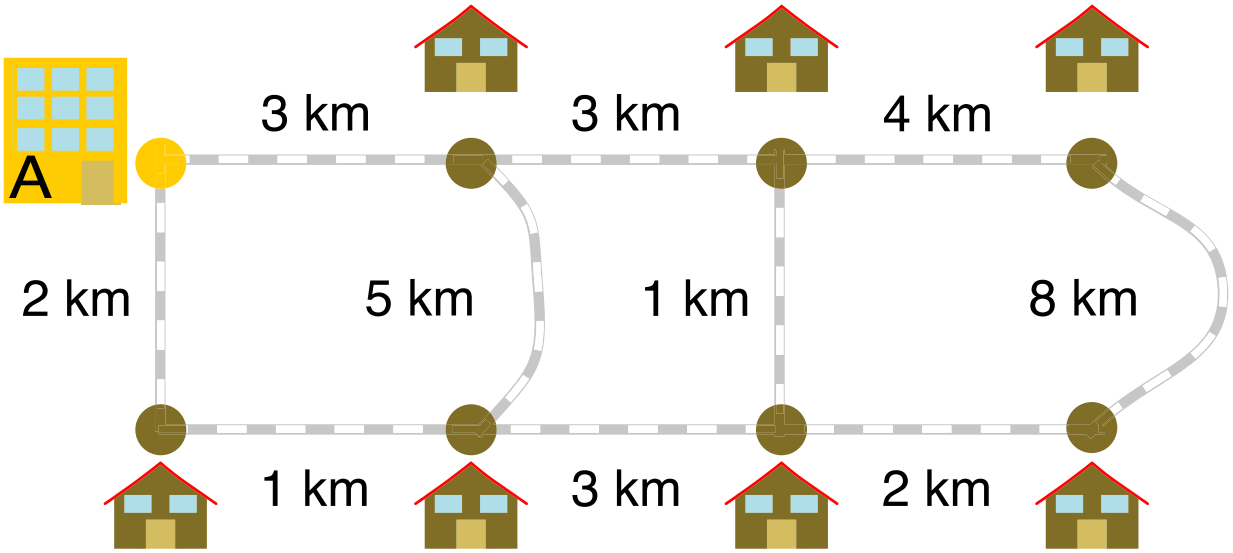
Dieser Einfüge-Algorithmus kann analog auch von Computerprogrammen zum Einfügen von Daten in eine Folge sortierter Daten verwendet werden. Aber er ist, wie sein Einsatz bei der Teller-Ordnung andeutet, nicht besonders schnell. Im schlechtesten Fall wird der Algorithmus alle bereits vorhandenen Teller bzw. Daten bewegen müssen. Bei großen Datenmengen bedeutet das viel Arbeit für den Computer. Informatikerinnen und Informatiker entwickeln deshalb clevere Algorithmen, mit denen es schneller geht. Zum Beispiel kann ausgenutzt werden, wenn es viele gleiche Daten in der sortierten Folge gibt – wie bei den Tellern in dieser Biberaufgabe.

Trainingstour

Ben arbeitet als Fahrradkurier. Jeden Tag fährt er eine Tour.
Er startet am Lager A und fährt zu allen Zielorten.
Jeden Zielort fährt er aber nur einmal an.
Beim letzten Zielort ist die Tour zu Ende.

Ben ist auch Radsportler. Aus jeder Tour macht er eine Trainingstour.
Das ist eine Tour mit maximaler Länge (in Kilometern).

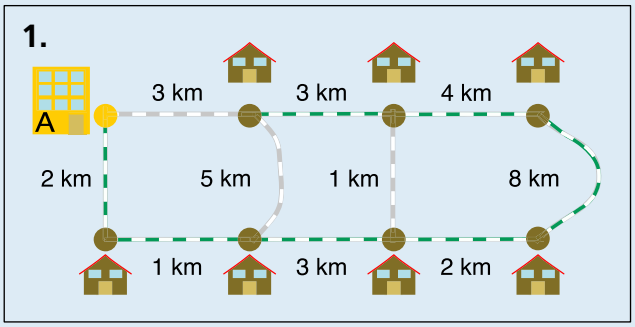
Heute hat Bens Tour sieben Zielorte.
Das Bild zeigt die Längen der Strecken, die Ben fahren kann.

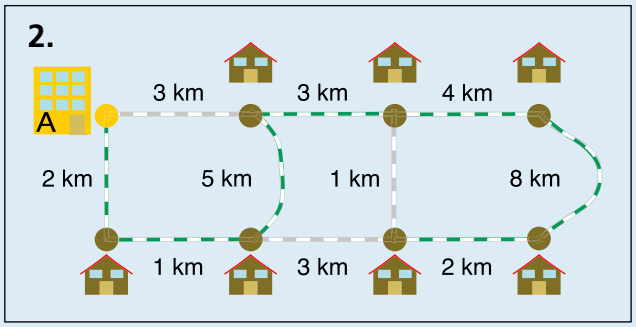


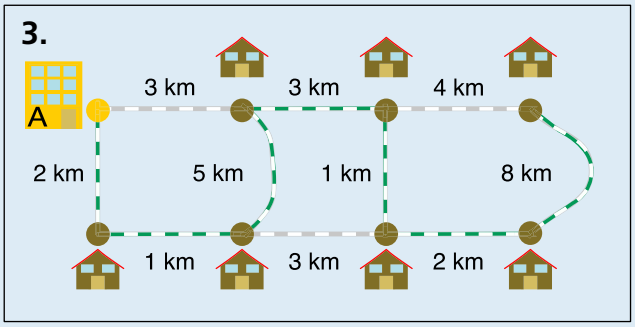
Wie lang ist Bens Trainingstour heute?

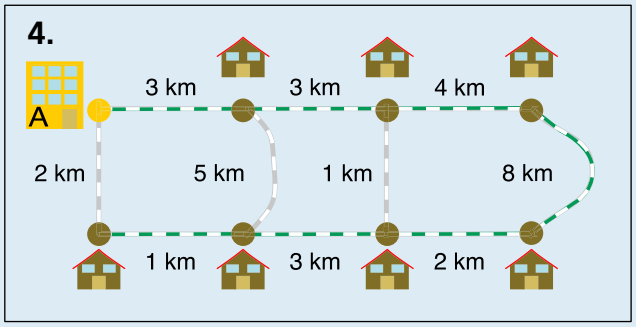
A) 22 km **B)** 23 km **C)** 24 km **D)** 25 km **E)** 26 km

B ist die richtige Antwort
Es gibt nur vier Möglichkeiten, beim Lager zu starten und alle Orte genau einmal anzufahren:









Die Längen dieser vier Touren sind:

2 + 1 + 3 + 2 + 8 + 4 + 3 = 23
2 + 1 + 5 + 3 + 4 + 8 + 2 = 25
2 + 1 + 5 + 3 + 1 + 2 + 8 = 22
3 + 3 + 4 + 8 + 2 + 3 + 1 = 24
Tour 2 ist mit 25 km die längste. Sie ist Bens Trainingstour.

Das ist Informatik!
Auf seiner Trainingstour möchte Ben alle Zielorte genau einmal anfahren. Für die Karte mit Zielorten und Strecken löst er damit das Hamiltonpfad-Problem. Die Karte ist nämlich wie ein Graph, eine mathematische Struktur aus Knoten und Kanten: Die Zielorte entsprechen den Knoten, und die Strecken den Kanten. Ein Hamiltonpfad ist ein Weg über Kanten des Graphen, bei dem alle Knoten genau einmal besucht werden. Da Ben außerdem die Längen der Strecken kennt und nicht nur irgendeine Tour fahren will, sondern eine Trainingstour, sucht er den längsten Hamiltonpfad im Karten-Graph.

Zu den wichtigsten Aufgaben der Informatik gehört es, die Schwierigkeit von Problemen einzuschätzen. Das Hamiltonpfad-Problem, also genau genommen die Entscheidung darüber, ob es in einem Graph einen Hamiltonpfad gibt, ist eines der berühmten NP-vollständigen Probleme und damit eines der schwierigsten Probleme, die die Informatik kennt. Den längsten Hamilton-Pfad zu bestimmen ist mindestens genau so schwierig. Das gilt allerdings nur bei allgemeiner Betrachtung. Wenn der Graph so übersichtlich ist wie die Streckenkarte in dieser Biberaufgabe, lässt sich der längste Hamiltonpfad gut bestimmen. Generell lässt sich die Informatik von einem schwierigen Problem nicht beeindrucken. Sie sucht dann unter anderem nach Spezialfällen, für die man das Problem doch gut lösen kann.
Für das Hamiltonpfad-Problem sind einige bekannt.

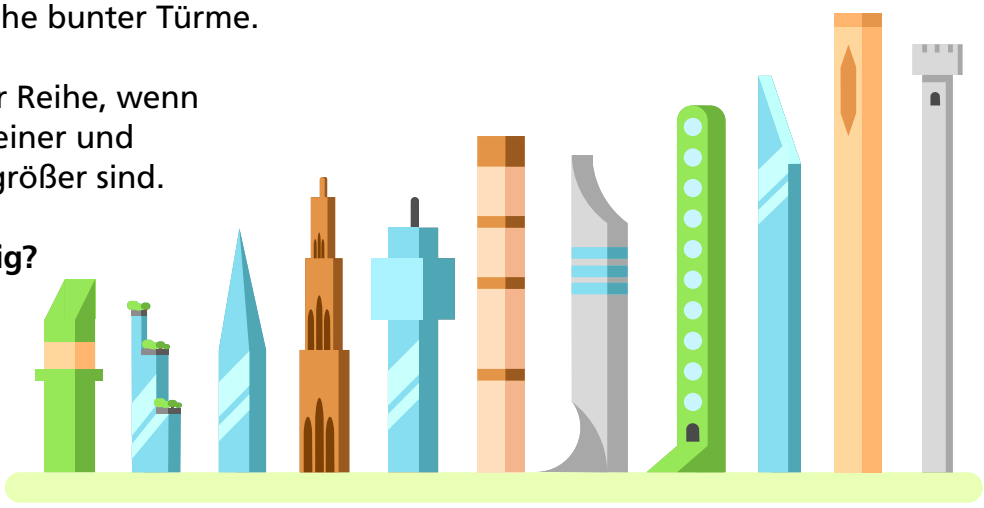
https://de.m.wikipedia.org/wiki/Hamiltonkreisproblem

Türme

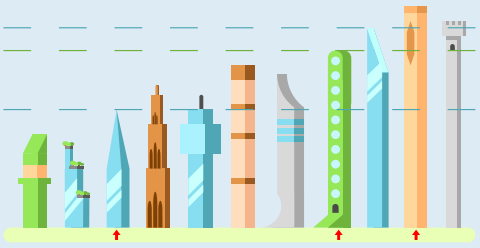
In Colortown steht eine Reihe bunter Türme.

Ein Turm steht richtig in der Reihe, wenn alle Türme links von ihm kleiner und alle Türme rechts von ihm größer sind.

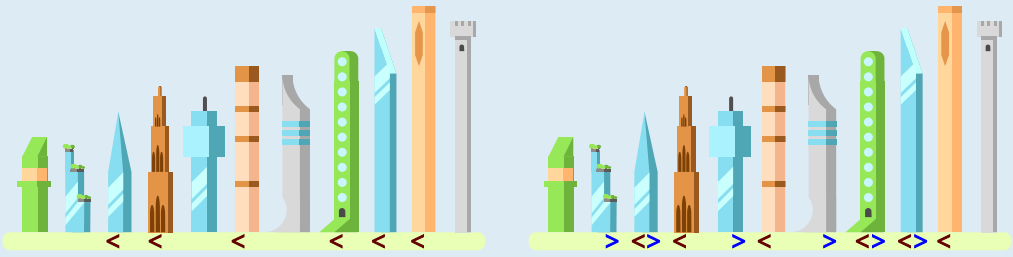
Welche Türme stehen richtig?



So ist es richtig:



Drei Türme stehen richtig, weil jeweils alle Türme links von ihnen kleiner und alle Türme rechts von ihnen größer sind. Das kann man an den gestrichelten waagerechten Linien erkennen, die genau auf der Höhe der Türme verlaufen.
Ein Verfahren, mit dem man die „richtigen“ Türme bestimmen kann, geht so: Zuerst geht man die Türme der Reihe nach durch und markiert einen Turm mit <, wenn alle Türme links von ihm kleiner sind. Dann geht man die Türme noch einmal durch und markiert einen Turm mit >, wenn alle Türme rechts von ihm größer sind. Alle Türme, die danach mit < > markiert sind, erfüllen beide Bedingungen und stehen richtig.

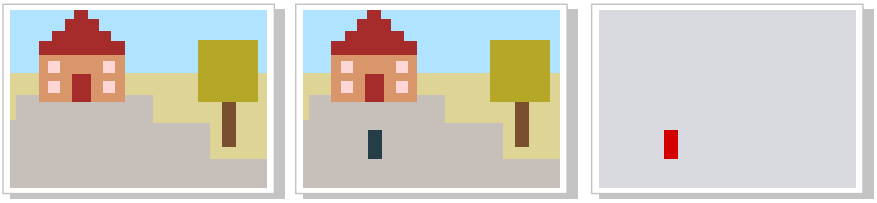


Das ist Informatik!
In dieser Biberaufgabe werden die Türme nach und nach anhand ihrer Höhe verglichen, um festzustellen, ob sie „richtig“ stehen. Ganz ähnlich kann man vorgehen, wenn man dafür sorgen will, dass alle Türme richtig stehen. Dazu geht man ebenfalls die Türme durch und vergleicht die Höhen benachbarter Türme miteinander. Findet man dabei einen Turm, der nicht richtig steht, weil der Turm direkt links von ihm größer ist, vertauscht man die beiden Türme. Das wiederholt man so lange, bis alle Türme richtig stehen. Dann ist die Turmreihe der Höhe nach sortiert.
Dieses Sortierverfahren kennt die Informatik unter dem Namen „Bubble-Sort“. Es funktioniert zuverlässig, aber nicht besonders schnell. Das macht sich besonders dann bemerkbar, wenn es um das Sortieren großer Datenmengen geht. Weil das Sortieren eine der wichtigsten Beschäftigungen von Computern ist, haben sich Informatikerinnen und Informatiker viele verschiedene und insbesondere schnelle Sortierverfahren überlegt. Sie werden unter anderem dadurch charakterisiert, wie viele Vergleiche beim Sortieren benötigt werden. Während Bubble Sort für 1.000 zu sortierende Elemente 1.000.000 Vergleichen benötigen kann, kommen schnellere Verfahren mit 10.000 Vergleichen aus. Bei 1.000.000. Elementen sind es schon 1.000.000.000.000 zu 20.000.000 Vergleichen. Auch wenn diese Zahlen nur grobe Annäherungen sind, ist der Unterschied jedenfalls groß.

Überwacht

Am Markplatz ist eine Überwachungskamera installiert. Sie macht alle 10 Sekunden ein Foto. Eine Software vergleicht jedes neue Foto mit dem vorherigen und erzeugt ein "Unterschiedsbild".
Dieses Bild enthält rote Pixelquadrate genau dort, wo sich die beiden verglichenen Fotos unterscheiden.

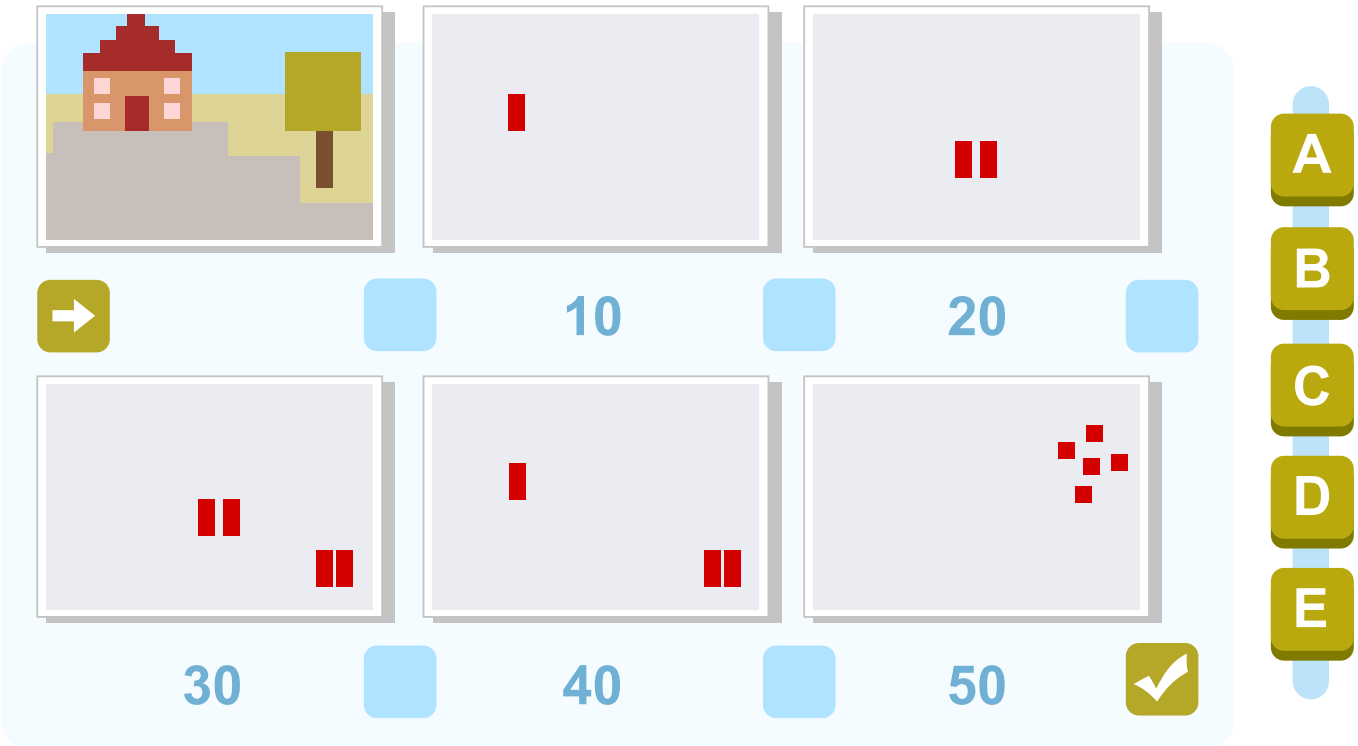
Ein Beispiel: Links sind zwei Fotos vom Markplatz, rechts ist das Unterschiedsbild des zweiten zum ersten Foto.



Nun sollst du über sechs hintereinander aufgenommene Fotos nachdenken. Unten siehst du das erste Foto, aber für die folgenden fünf Fotos jeweils nur das Unterschiedsbild zum vorherigen Foto. Zwischen den Fotos wurde immer ein neues Ereignis beobachtet, und zwar jeweils eines aus dieser Liste:

A	Tom trifft Tina.
B	Jemand öffnet die Rathaustür
C	Tom und Tina gehen Arm in Arm
D	Wind kommt auf.
E	Jemand schließt die Rathaustür.

Wann wurde welches Ereignis beobachtet?
Ziehe jeden Ereignis-Buchstaben auf die passende Position.



So ist es richtig:
Diese beiden Abfolgen von Ereignis-Buchstaben sind richtig: BACED und EACBD.
Zwischen dem ersten Foto und dem Foto nach 10 Sekunden: Jemand öffnet die Rathaustür (B).
Das Unterschiedsbild enthält rote Pixelquadrate an der Position der Tür.
Zwischen den Fotos nach 10 und nach 20 Sekunden: Tom trifft Tina (A).

Das Unterschiedsbild enthält rote Pixelquadrate am Treffpunkt von Tom und Tina. Auf dem Foto bei 10 Sekunden waren die beiden noch nicht zu sehen.

Zwischen den Fotos nach 20 und nach 30 Sekunden: Tom und Tina gehen Arm in Arm (C).
Das Unterschiedsbild enthält rote Pixelquadrate dort, wo die beiden vorher standen, und dort, wo sie jetzt sind.

Zwischen den Fotos nach 30 und nach 40 Sekunden: Jemand schließt die Rathaustür(E).
Das Unterschiedsbild enthält rote Pixelquadrate an der Position der Tür – und dort, wo Tom und Tina vorher standen; die beiden sind auf dem neuen Foto nicht mehr zu sehen.
Zwischen den Fotos nach 40 und nach 50 Sekunden: Wind kommt auf (D).
Das Unterschiedsbild enthält rote Pixelquadrate an der Position der Baumkrone. Der Wind hat die Blätter bewegt und ihre Position verändert.

Die Ereignisse B und E sind austauschbar, weil sie sich bezüglich der Unterschiedsbilder nicht unterscheiden.

Das ist Informatik!

Auf öffentlichen Plätzen, in Gebäuden wie Bahnhöfen und Flughäfen, in Parkhäusern: überall stehen Überwachungskameras. Wer wissen will, was an den beobachteten Orten passiert ist, kann sich die Aufzeichnungen der Kamerabilder ansehen – solange diese aufgehoben werden. Aber nicht nur Menschen durchsuchen die Aufzeichnungen, sondern auch Computerprogramme. Mit Methoden der Bildverarbeitung oder des maschinellen Sehens können ein unerwünschtes Eindringen in Sicherheitsbereiche erkannt oder im Rahmen einer Fahndung Verdächtige identifiziert werden. Wenn man aber Verdächtige identifizieren will, muss man erst einmal die Bilder aller Personen aufzeichnen und verarbeiten – wie bei Tom und Tina. Es geht also Privatsphäre verloren, wenn überall Überwachungs-kameras installiert werden. Auch hierzu leistet die Informatik einen Beitrag.

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

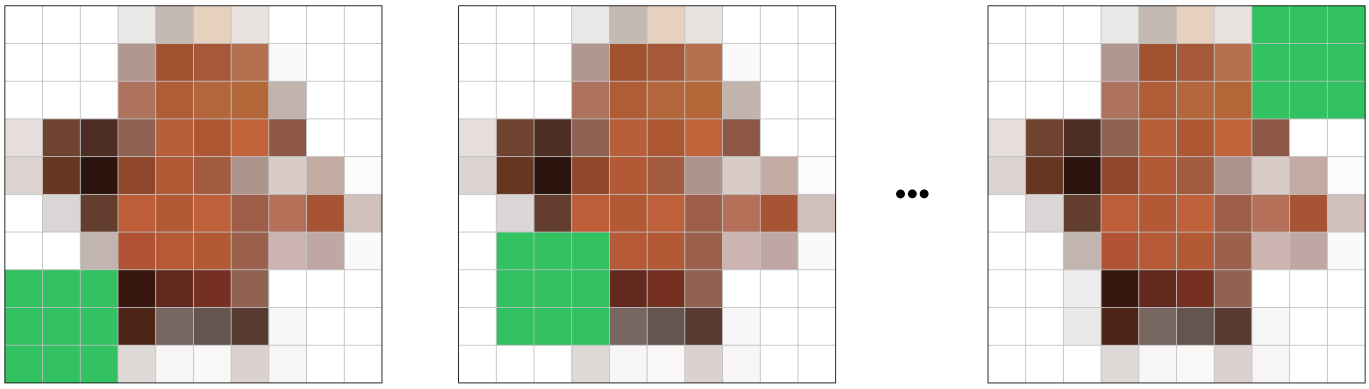
Video speichern

Videos benötigen viel Speicherplatz: Sie bestehen aus vielen Einzelbildern,
und jedes Einzelbild besteht aus vielen Bildpunkten.
Mit dem folgenden Verfahren kann man aber Speicherplatz sparen.

Dabei speichert man
- beim ersten Einzelbild alle Bildpunkte und
- bei allen weiteren Einzelbildern nur die Bildpunkte,
  die sich im Vergleich zum vorherigen Einzelbild geändert haben.

Hier ist ein (sehr kleines) Video. Seine Einzelbilder haben 10×10 Bildpunkte.
Das grüne Quadrat in der unteren linken Ecke des ersten Einzelbildes ist 3×3 Bildpunkte groß.

Das grüne Quadrat bewegt sich von einem Einzelbild zum nächsten diagonal um einen Bildpunkt nach rechts und nach oben, bis es im letzten Einzelbild in der oberen rechten Ecke landet.



Nun wird dieses Video mit dem oben beschriebenen Verfahren gespeichert.

Wie viele Bildpunkte müssen dabei für das gesamte Video gespeichert werden?

A) 100 **B)** 135 **C)** 140 **D)** 170 **E)** 180 **F)** 700 **G)** 800 **H)** 1000

D ist die richtige Antwort:
So sieht es aus, wenn man in den Einzelbildern des Videos die geänderten Bildpunkte markiert:

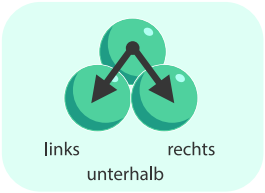
Jedes Einzelbild besteht aus 10 x 10 = 100 Bildpunkten. Für das erste Einzelbild müssen also 100 Bildpunkte gespeichert werden. Für jedes weitere Einzelbild müssen die geänderten Bildpunkte gespeichert werden. Das sind die fünf Bildpunkte unten links vom Quadrat, die durch die Bildpunkte des Hintergrunds ersetzt werden, sowie die fünf Bildpunkte oben rechts im Quadrat, die den bisherigen Hintergrund überdecken. Pro Einzelbild werden also 10 Bildpunkte geändert und müssen gespeichert werden. Das Quadrat braucht nach dem ersten Einzelbild 7 weitere Einzelbilder, um sich von unten links nach oben rechts zu bewegen. Für diese weiteren Einzelbilder müssen also 10 x 7 = 70 Bildpunkte gespeichert werden. Für das gesamte Video müssen 170 Bildpunkte gespeichert werden.

Das ist Informatik!
Computer – und damit auch Smartphones – ohne Videos? In diesem Biberjahr 2019 ist das nicht vorstellbar. Videoplattformen im Internet oder das Streaming von Filmen und Serien gehören zum Alltag zumindest jüngerer Menschen. Dabei ist das nicht selbstverständlich. Die Aufgabe deutet schon an, dass bei Videos große Datenmengen zusammenkommen. In der Realität sind die Videos nicht so klein wie in dieser Biberaufgabe. Ein Video im HD-Format hat 1920 x 1080 und damit über 2 Millionen Bildpunkte, und zwar für jedes Einzelbild. Bei den lange Zeit üblichen 3 Byte pro Bildpunkt ist ein Einzelbild also gut 6 Millionen Byte groß. Wenn man von den bei Kinofilmen gebräuchlichen 24 Einzelbildern pro Sekunde ausgeht, kommt ein 30-minütiges Video (zum Beispiel eine Serienfolge) auf 24 x 60 x 30 = 43.200 Einzelbilder und damit auf einen Speicherbedarf von grob 260 Gigabyte – und das wäre auch die Datenmenge, die beim Streaming über die Internetverbindung zu laden wäre. Das wäre selbst bei heutigen Festplatten und Downloadraten sehr viel.
In der Informatik wird deshalb schon lange daran gearbeitet, Videodaten möglichst gut zu komprimieren, um deren Speicherbedarf zu verringern. „Möglichst gut“ bedeutet dabei natürlich auch, Verluste bei der Bildqualität nur soweit hinzunehmen, wie das menschliche Auge sie nicht wahrnimmt. Ein wichtiger Bestandteil jeder guten Videokompression ist die Differenzkodierung, bei der nur die Unterschiede zu vorherigen Einzelbildern oder Bildpunkten gespeichert werden – wie in dieser Biberaufgabe.

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

Wackelig

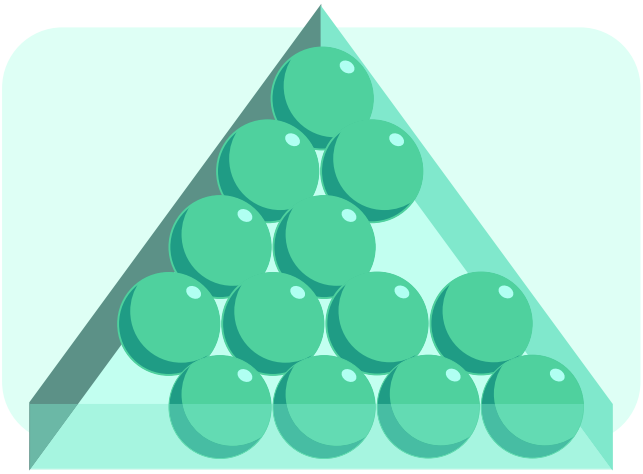
13 Kugeln liegen in einem dreieckigen Kasten (siehe unten).
Wenn wir den Kasten an der Spitze anheben und nach vorne kippen, drohen einige Kugeln nach unten zu rollen. Diese Kugeln nennen wir wackelig.



Eine Kugel ist wackelig, wenn (mindestens) eine der beiden folgenden Bedingungen erfüllt ist:

- Es gibt eine Lücke links oder rechts unterhalb der Kugel.
- Es gibt zumindest eine wackelige Kugel links oder rechts unterhalb der Kugel.

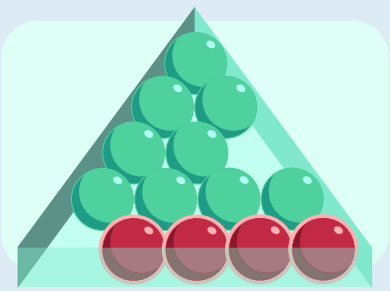
Welche Kugeln sind NICHT wackelig?

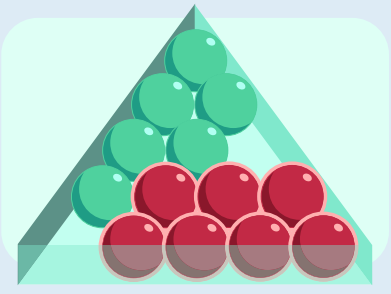


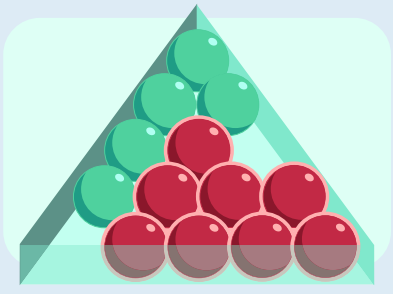
So ist es richtig:
Man kann die wackeligen Kugeln auf folgende Weise bestimmen:

1. Beginne mit der untersten Reihe in dem dreieckigen Kasten und markiere alle Kugeln in dieser Reihe.
2. Mache mit der nächsthöheren Reihe weiter.
3. Markiere alle Kugeln, die auf zwei markierten Kugeln liegen.
4. Wiederholle die Schritte 2 und 3, bis du alle Reihen bearbeitet hast..
5. Alle Kugel, die jetzt markiert sind, sind nicht wackelig.

Die Bilder illustrieren dieses Verfahren. Ab der vierten Reihe werden keine Kugeln mehr markiert.







Das ist Informatik!
In dieser Biberaufgabe heißt eine Kugel wackelig, wenn sie eine von zwei Bedingungen erfüllt. Wenn man die zweite Bedingung prüfen will, muss man feststellen, ob die darunter liegenden Kugeln wackelig sind. Dazu muss man eventuell wiederum feststellen, ob die darunter liegenden Kugeln wackelig sind. Das hört sich so an, als ob es endlos weitergehen würde. Tut es aber nicht. Denn der Rückgriff auf das Wackelig-sein anderer Kugeln hat eine Richtung: unten. Deshalb ist klar, dass in der unteren Reihe alle Kugeln nicht wackelig sind – es gibt weder Lücken noch Kugeln darunter.

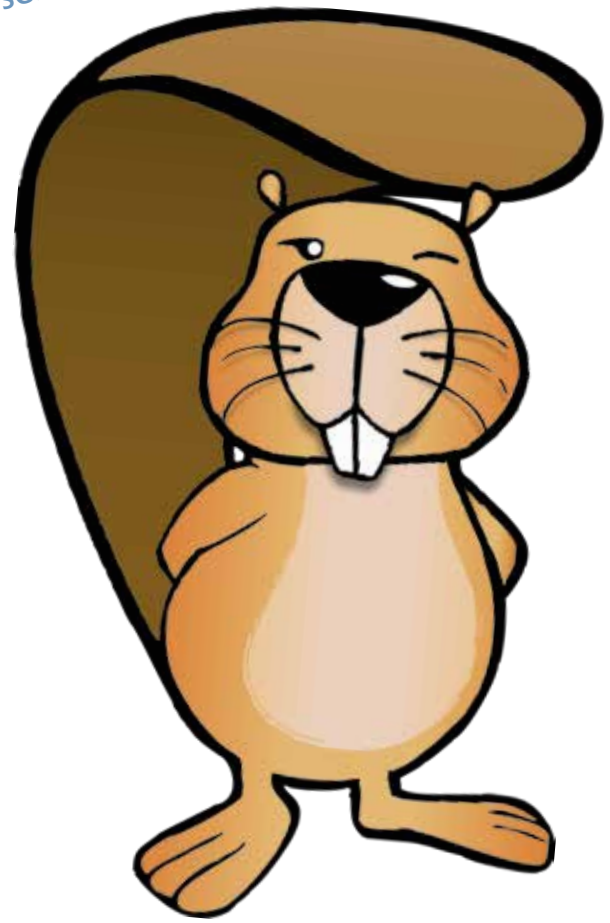
Die Beschreibung bzw. Definition der Wackeligkeit einer Kugel in dieser Biberaufgabe verwendet Rekursion – so wie bei den „Stern-Mobiles“ weiter vorne im Heft: Zur Definition einer Eigenschaft (oder eines Begriffs) wird sie selbst verwendet. Damit das funktioniert und nicht endlos wird, muss eine rekursive Definition einen Basisfall enthalten, den man ohne Rückgriff auf die definierte Eigenschaft entscheiden kann. Für die wackeligen Kugeln ist der Basisfall die Lage in der unteren Reihe.

Die Informatik kennt viele rekursiv beschriebene Strukturen. Deshalb ist es naheliegend, die Abläufe zur Verarbeitung dieser Strukturen auch rekursiv zu programmieren. Das ist in allen verbreiteten Programmiersprachen möglich. Damit die rekursiven Programme nicht endlos laufen, müssen die Programmiererinnen aber unbedingt auf eines achten: den Basisfall!

Du hast fast das ganze Biberheft gelesen.
Zielgruppe: Informatik-interessiert.
Du solltest Programmieren lernen:

jwinf.de

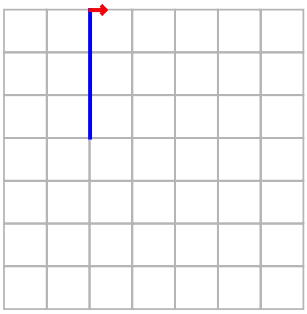
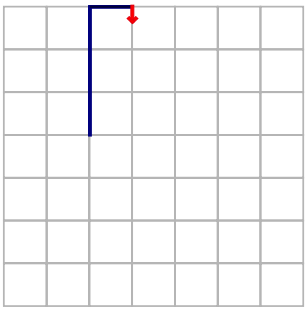
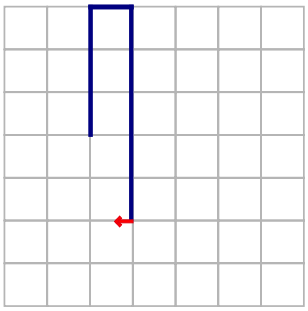
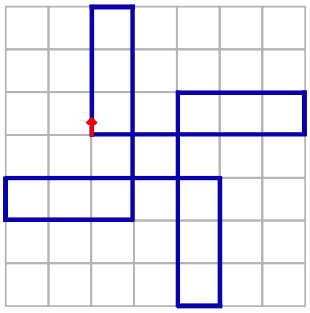
cscircles.cemc.uwaterloo.ca/de



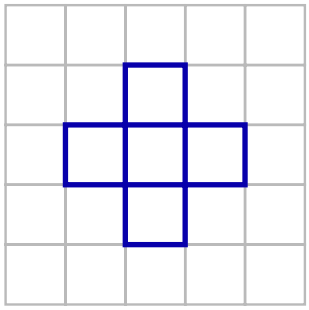
Zeichenroboter

Ein Roboter bewegt sich über ein Raster und zeichnet dabei Linien.
Mit drei Zahlen kann man den Roboter steuern.
Die Zahlen geben dem Roboter drei Schritte an, die er beliebig oft wiederholt.
So entsteht aus den Zahlen ein Bild.

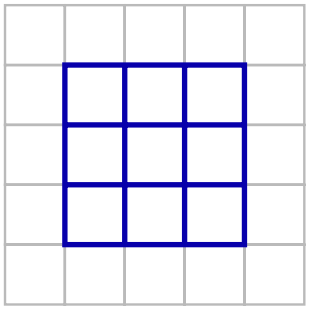
Ein Beispiel: Aus den Zahlen 3, 1, 5 entsteht das Bild rechts, und zwar so:
- gehe 3 Felder vor und drehe nach rechts (Schritt 1),
- gehe 1 Felder vor und drehe nach rechts (Schritt 2) und
- gehe 5 Felder vor und drehe nach rechts (Schritt 3).

Schritt 1	Schritt 2	Schritt 3	Bild
			

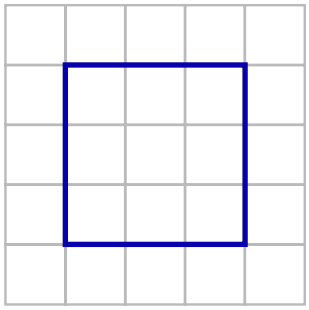
Welches Bild entsteht aus den Zahlen 2, 2, 3 ?



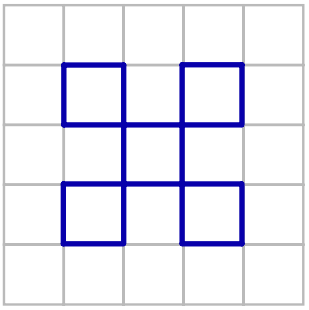
A)



B)

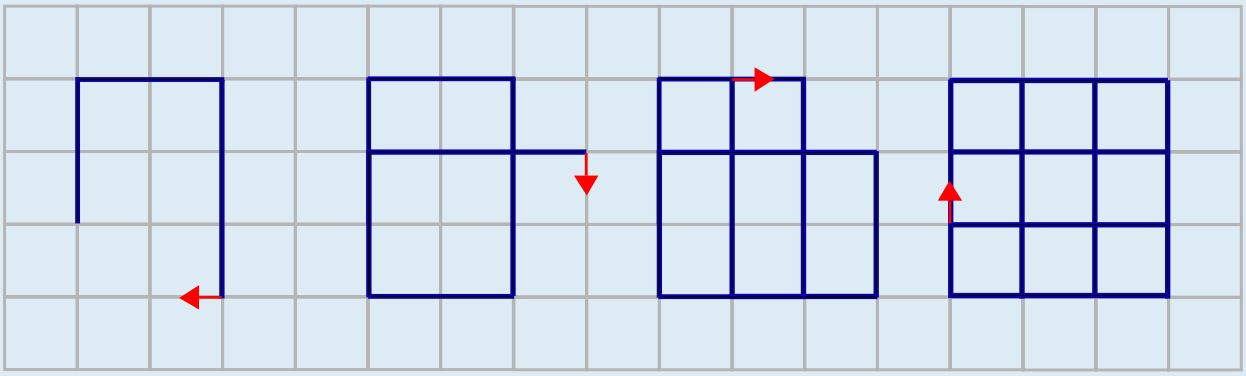


C)



D)

Antwort B ist richtig:
Die Zahlen 2, 2, 3 bedeuten, dass der Roboter zunächst 2 Felder vorgeht und sich nach rechts dreht, wieder 2 Felder vorgeht und sich nach rechts dreht und dann 3 Felder vorgeht und sich nach rechts dreht. Danach hat er die Linien im Bild links gezeichnet. Die nächsten Bilder zeigen, welche Linien der Roboter nach weiteren Wiederholungen der drei Schritte gezeichnet hat. Das Bild aus Antwort B entsteht also aus den Zahlen 2, 2, 3, indem der Roboter die drei Schritte mindestens viermal ausführt.



Das ist Informatik!

Der Zeichenroboter in dieser Biberaufgabe kann nur sehr einfache Programme ausführen. Die Programmiersprache, die der Roboter versteht, kennt nur Zahlen als Anweisungen, und jedes Programm muss aus genau drei Anweisungen, also drei Zahlen bestehen. Außerdem ist fest eingebaut, dass der Roboter die in einem Programm enthaltenen Anweisungen endlos wiederholt, ob die Programmiererin will oder nicht.

Die meisten Roboter und Computer verstehen sehr viel komplexere (Programmier-)Sprachen. Alle diese Sprachen haben die gleichen grundlegenden Eigenschaften: (1) Die Programme dürfen aus beliebig (aber nicht endlos) vielen Anweisungen bestehen. (2) Wiederholungsanweisungen verschiedener Art können, müssen aber nicht verwendet werden. (3) Außerdem gibt es bedingte Anweisungen, die abhängig vom Programmzustand unterschiedliche Programmausführungen erlauben.

Enthält eine Programmiersprache Wiederholungsanweisungen und bedingte Anweisungen, kann man mit ihrer Hilfe alles berechnen, was überhaupt berechenbar ist. In der Informatik werden solche Programmiersprachen als universell bezeichnet – oder auch als “Turing-vollständig”.

https://de.wikipedia.org/wiki/Turing-Vollständigkeit

Zum Strand

Es ist Sommer!
Angela (12 Jahre alt) geht zum Strand.
Sie nimmt ihren Bruder Fred mit. Fred ist 6 Jahre alt.
Am Eingang zum Strand ist ein Schild:

Wer darf auf den Strand?

A) Angela und Fred.

B) Angela, aber Fred nicht.

C) Angela nicht, aber Fred.

D) Keiner von beiden.

Antwort A ist richtig:
Auf dem Schild stehen zwei Regeln:

1. Personen, die 8 Jahre oder älter sind, dürfen auf den Strand.
2. Personen, die jünger als 8 Jahre sind, dürfen nur dann auf den Strand, wenn sie von einer Person begleitet werden, die älter als 10 Jahre ist.

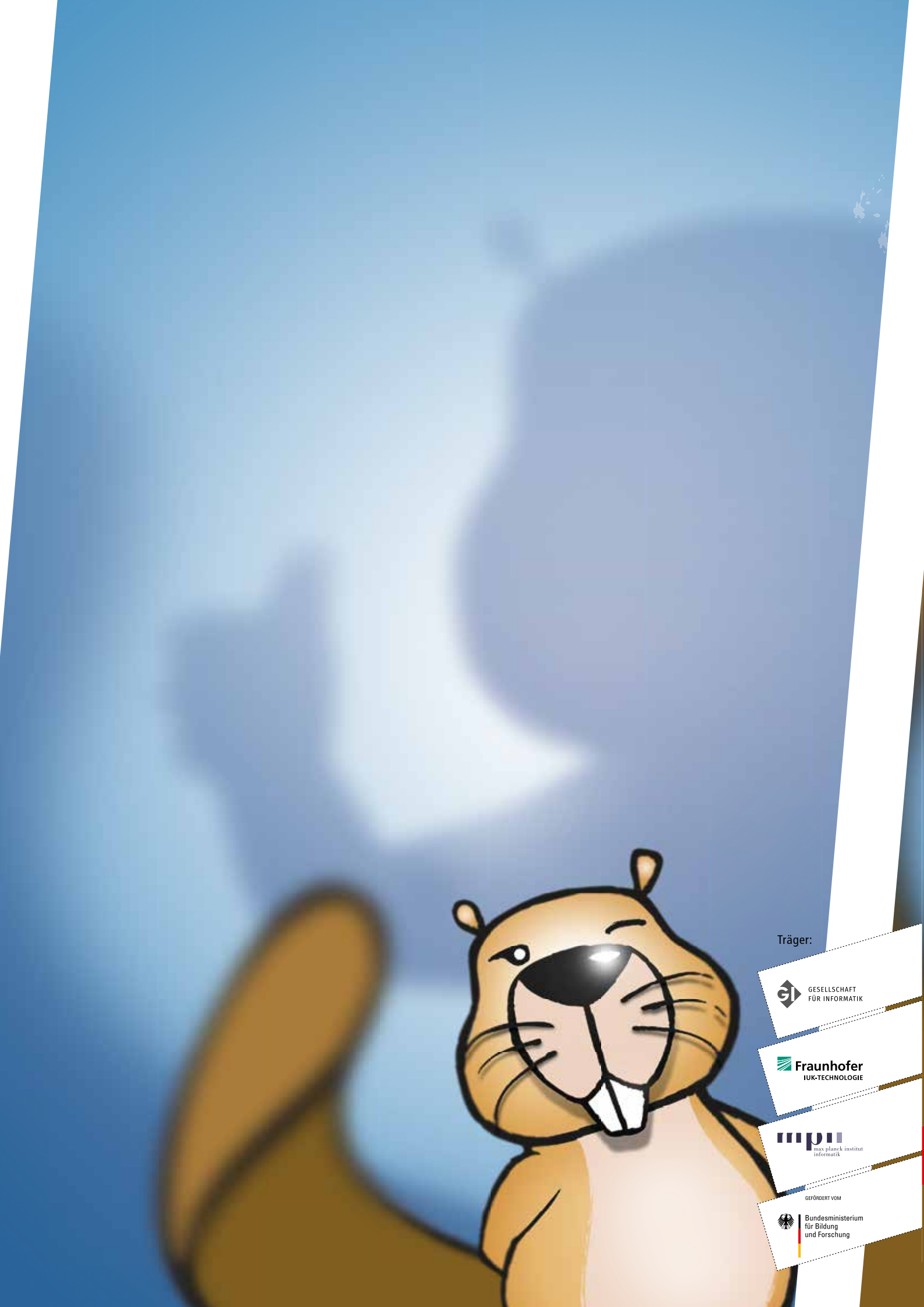
Da Angela älter als 8 Jahre ist, darf sie wegen Regel 1 auf den Strand. Fred ist jünger als 8 Jahre, aber da Fred von Angela begleitet wird und Angela älter als 10 Jahre ist, darf wegen Regel 2 auch Fred auf den Strand.

Das ist Informatik!

Darf eine Person auf den Stand? Das Schild in dieser Biberaufgabe nennt dazu zwei Bedingungen. Die zweite Bedingung muss aber nur geprüft werden, wenn die erste Bedingung nicht erfüllt ist. Für einen Kontrolleur, der die Bedingungen auf dem Schild umsetzen soll, kann man eine passende Dienstanweisung deshalb so aufschreiben:

Wenn die Person 8 Jahre oder älter ist:
    Dann lasse sie auf den Strand
    Sonst: Wenn die Person von einer älter als 10 Jahre alten Person begleitet wird:
        Dann lasse sie auf den Strand
        Sonst lasse sie nicht auf den Strand

Solche „Wenn – dann – sonst“ Anweisungen (englisch: if – then – else) sind ein wichtiger Bestandteil jeder guten Programmiersprache. In der Informatik heißen sie auch „bedingte Anweisungen“ oder „Verzweigungen“. Wer programmieren können will, muss den Umgang mit bedingten Anweisungen beherrschen, und zwar unbedingt.



Träger:

GESELLSCHAFT FÜR INFORMATIK

Fraunhofer IUK-TECHNOLOGIE

mpii max planck institut informatik

GEFÖRDERT VOM

Bundesministerium für Bildung und Forschung