Biber der Informatik 2023
Aufgaben und Lösungen

OESTERREICHISCHE COMPUTER GESELLSCHAFT ®
AUSTRIAN COMPUTER SOCIETY

Ein Wettbewerb der OCG

Das Österreichische Biber Team 2023

Liam Baumann, Österreichische Computer Gesellschaft (OCG)

Wilfried Baumann, OCGr

Gerald Futschek, Technische Universität Wien

Josefine Hiebler, OCG

Der Biber der Informatik

ist ein Projekt der Österreichischen Computer Gesellschaft OCG in Zusammenarbeit mit dem Institut Information Systems Engineering der TU Wien. Der Biber der Informatik wird vom Bundesministerium für Bildung, Wissenschaft und Forschung empfohlen und wurde 2018 mit dem eAward in der Kategorie Bildung ausgezeichnet.

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

Masiar Babazadeh, Scuola universitaria professionale della Svizzera italiana / Schweiz. Verein für Informatik in der Ausbildung (SVIA)

Liam Baumann, Österreichische Computer Gesellschaft (OCG)

Wilfried Baumann, OCG

Tobias Berner, PH Zürich

Christian Datzko, Wirtschaftsgymnasium und Wirtschaftsmittelschule Basel

Susanne Datzko, freischaffende Graphikerin / ETH Zürich

Nora A. Escherle, SVIA

Gerald Futschek, Technische Universität Wien

Angélica Herrera Loyo, ETH Zürich / Ausbildungs- und Beratungszentrum f. Informatikunterricht (ABZ)

Josefine Hiebler, OCG

Juraj Hromkovic, ETH Zürich/ABZ

Dennis Komm, ETH Zürich / ABZ

Regula Lacher, ETH Zürich / ABZ

Gabriel Parriaux, Haute École Pédagogique Vaud / SVIA

Jean-Philippe Pellet, Haute École Pédagogique Vaud / SVIA

ZsuzsaPluhár,ELTE Informatikai Kar

Giovanni Serafini, ETH Zürich / ABZ

Bernadette Spieler, PH Zürich

Bebras: International Challenge on Informatics and Computational Thinking

Der Biber der Informatik ist Partner der internationalen Initiative Bebras. 2004 fand in Litauen der erste Bebras Challenge statt. 2006 traten Estland, die Niederlande und Polen der Initiative bei, und auch Deutschland veranstaltete im damaligen Informatikjahr als „EI:Spiel blitz!“ einen ersten Biber-Testlauf. Seitdem kamen viele Bebras-Länder hinzu.

Zum Drucktermin sind es weltweit 78, und weitere Länderteilnahmen sind in Planung. Insgesamt hatte der Bebras Challenge 2023 weltweit über 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 34 Aufgaben des Informatik-Biber 2023 stammen aus 22 Ländern: Australien, Belgien, Brasilien, Deutschland, Indien, Iran, Irland, Kanada, Litauen, Neuseeland, Österreich, Peru, Rumänien, Russland, Schweiz, Slowakei, Südkorea, Tschechien, Ukraine, Ungarn, Uruguay und aus den Vereinigten Staaten.

Der brasilianische Biber

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

Der uruguayaische Biber

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

Der peruanische Biber

Der Biber der Informatik

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

Der Informatik-Biber findet jährlich im November statt. An der 17. Austragung im Jahr 2023 beteiligten sich 380 Schulen und andere Bildungseinrichtungen mit 57.970 Schülerinnen und Schülern; das sind neue Rekordwerte.

Die Teilnahme am Biber der Informatik 2023 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 diese Dynamik der Aufgabenbearbeitung nicht vorführbar. Handlungstipps in den Aufgabenstellungen und Bilder von Lösungssituationen geben aber eine Vorstellung davon.
Der Umgang mit dem Wettbewerbssystem konnte in den Wochen vor der Austragung geübt werden.

Der Biber der Informatik wurde in fünf Altersgruppen durchgeführt. Die Aufgaben jeder Altersgruppe sind in die Schwierigkeitsstufen leicht, mittel und schwer eingeteilt. In den Klassenstufen 3 bis 4 waren innerhalb von 30 Minuten 9 Aufgaben zu lösen, drei in jeder Schwierigkeitsstufe. In den Klassenstufen 5 bis 6 waren innerhalb von 35 Minuten 12 Aufgaben zu lösen, vier pro Schwierigkeitsstufe. In den Klassenstufen 7 bis 8, 9 bis 10 und 11 bis 13 waren innerhalb von 40 Minuten 15 Aufgaben zu lösen, jeweils fünf in jeder S

Wir laden die Schülerinnen und Schüler ein, auch 2024 wieder beim Informatik-Biber mitzumachen, und zwar in der Zeit vom 7. bis 18. November. Weitere Informationen werden über die Website ocg.at/biber und per E-Mail an die Koordinatorinnen und Koordinatoren bekannt gegeben.

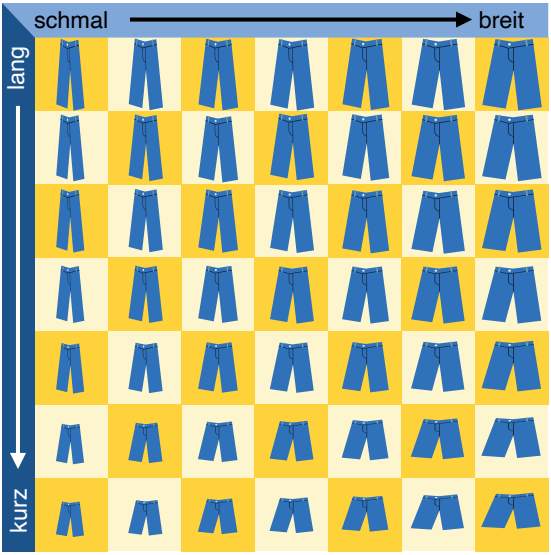
Aufgabenliste

Das sind die 34 Aufgaben des Informatik-Biber 2023, grob geordnet nach steigender Schwierigkeit und gelistet mit einer Klassifikation ihres informatischen Inhalts.

Anprobe

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

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



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

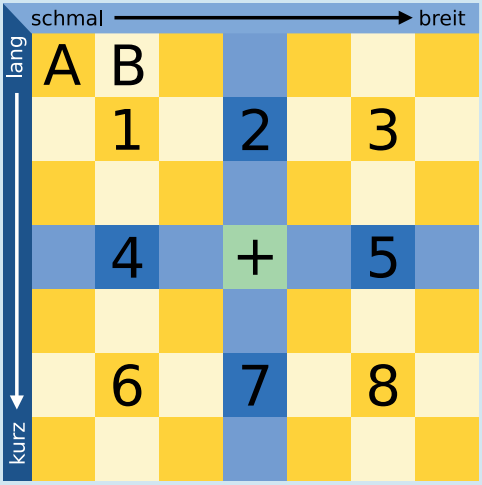
Doch Christian ist eine Methode eingefallen, die richtige Größe in jedem Fall nach möglichst wenigen Anproben zu wissen.

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

2 ist die richtige Antwort.

Natürlich kann Christian Glück haben und direkt bei der ersten Anprobe die Hose in der richtigen Größe erwischen. Aber auf Glück kann er sich nicht verlassen und geht nach dieser Methode vor: Zuerst probiert er die Hose in der Mitte an (Position + im Bild). Bei der Anprobe prüft er Länge und Breite der Hose.

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



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

Nun gibt es wieder mehrere Möglichkeiten:

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

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

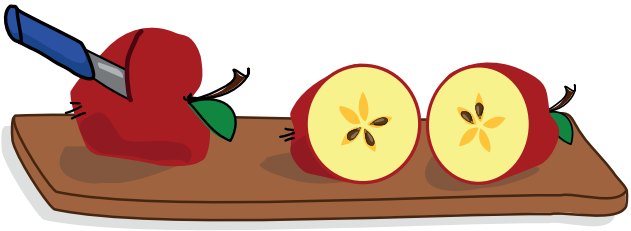
Das ist Informatik!

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

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

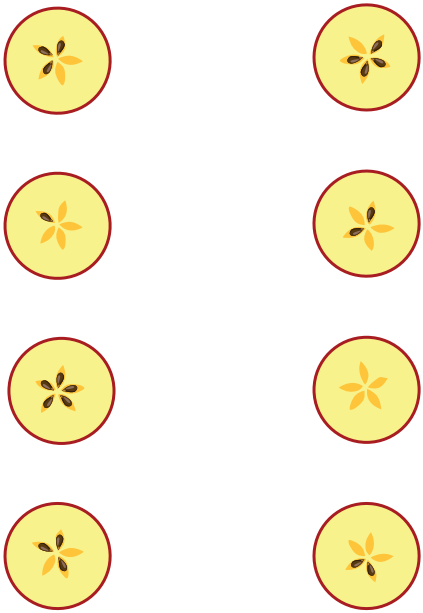
Äpfel halbieren

Äpfel kann man in eine obere und untere Hälfte teilen. Einige Apfelkerne bleiben in der oberen Hälfte, die anderen in der unteren Hälfte. An den Löchern und Kernen sieht man, dass die Hälften zusammen passen:

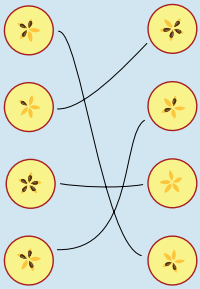


Gala halbiert vier Äpfel. Sie legt die oberen Hälften links und die unteren Hälften rechts untereinander.

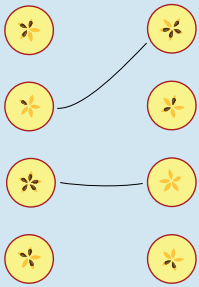
Welche Apfelhälften passen zusammen? Ordne die Apfelhälften einander zu.

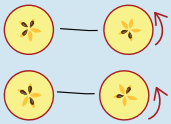


So ist es richtig:



Jeder Apfel hat 5 Apfelkerne. Zwei zueinander passende Apfelhälften müssen also insgesamt 5 Kerne haben. Diese Hälften können damit einfach zugeordnet werden:





Die übrigen vier Hälften sind nicht so einfach zuzuordnen. Die beiden oberen Hälften haben jeweils 3 Kerne, die beiden unteren Hälften haben jeweils 2 Kerne. Deshalb schauen wir uns die Muster der Apfelkerne genauer an. Denn wenn zwei Hälften zusammen passen, passen auch die Muster zusammen. Um das zu sehen, kann es aber nötig sein, die Hälften zu drehen.

Dann lassen sich die Hälften so zuordnen wie im Bild: Oben hat die obere Hälfte drei Kerne direkt hintereinander und danach zwei Löcher (K-K-K-L-L), die untere Hälfte hat drei Löcher direkt hintereinander und danach zwei Kerne (L-L-L-K-K): die Hälften passen zusammen. Auch unten passen die beiden Hälften zusammen: Das Muster der oberen Hälfte lautet (wenn man oben beginnt und im Uhrzeigersinn weitermacht) K-L-K-L-K, das Muster der unteren Hälfte (wenn man unten beginnt) L-K-L-K-L.

Das ist Informatik!
Bei der Erklärung der richtigen Antwort haben wir gesehen: Wenn zwei Hälften zusammen passen, passen nicht nur die Anzahlen der Kerne zusammen, sondern auch die Reihenfolgen der Kerne und Löcher (also der leeren Kernfächer). Für die richtige Zuordnung der Hälften muss man also auch diese Reihenfolgen betrachten. Es genügt nicht zu wissen, wie viele Kerne in jeder Hälfte sind.

Bei Problemen, die mit Hilfe von Computerprogrammen gelöst werden sollen, stellen sich ähnliche Fragen. Informatikerinnen und Informatiker müssen sich Gedanken machen, wie die Informationen, die das Programm berücksichtigen soll, als Daten beschrieben werden. Dabei wird oft versucht, es so einfach wie möglich zu machen. Einfache Programme sind nämlich weniger anfällig für Fehler. Beim Apfelhälften-Problem in dieser Biberaufgabe schien es zunächst ausreichend zu sein, die Hälften allein durch die Anzahl der Kerne zu beschreiben. Doch dann wurde klar, dass das nicht in allen Fällen genügt. Zur Beschreibung der Apfelhälften in einem Computerprogramm muss es also möglich sein, eine Reihenfolge zu beschreiben. Das geht zum Beispiel mit Hilfe der Datenstruktur *Liste,* die in den meisten Programmiersprachen zur Verfügung steht.

Aylas Regenschirm

Das ist Aylas Regenschirm:

Eines der vier Bilder zeigt Aylas Regenschirm. Welches?

A

B

C

D

Antwort C ist richtig:
Jedes Muster auf Aylas Regenschirm kommt genau einmal vor.

Um das korrekte Bild zu finden, vergleichen wir nacheinander jedes der Bilder mit Aylas Regenschirm:
- Wir wählen das Muster, welches am weitesten links ist, und suchen dessen Position auf Aylas Regenschirm.
- Wir prüfen, ob die angrenzenden Muster dieselben sind wie die auf Aylas Regenschirm.

Antwort	A	B	C	D
Antwortbild				
Aylas Regenschirm				

Jedes Antwortbild zeigt eine Folge von fünf Mustern. Ob eine dieser Folgen mit der vollständigen Reihenfolge aller zehn Muster von Aylas Regenschirm übereinstimmt, können wir nicht wissen. Bild C zeigt aber als einziges eine Folge, die vollständig mit einer Folge von fünf Mustern auf Aylas Regenschirm übereinstimmt. Aus diesem Grund kann nur Bild C Aylas Regenschirm zeigen. Alle anderen Bilder zeigen Musterfolgen, die nicht oder nur teilweise mit Aylas Regenschirm übereinstimmen. Diese Bilder können also nicht Aylas Regenschirm zeigen.

Das ist Informatik!
In den Antwortmöglichkeiten ist jeweils nur ein Teil der Musterfolge abgebildet. Obwohl sie nur eine Teilinformation enthalten, können wir feststellen, welches der vier Bilder Aylas Regenschirm zeigt: Ein Bild zeigt nur dann Aylas Regenschirm, wenn die Musterfolge vollständig in der Musterfolge von Aylas Regenschirm enthalten ist.

Das gleiche Prinzip wie bei der „Regenschirm-Mustersuche“ wird bei der Suche in einem Textdokument angewendet. Der Computer sucht mit gegebenen Teilinformationen (Suchwort) nach passenden *Zeichenketten* im Dokument. Eine Zeichenkette ist eine Folge von Zeichen (z.B. Buchstaben, Ziffern, Sonderzeichen). Dabei gilt bei der Suche:

- Je länger das Suchwort, desto weniger mögliche Übereinstimmungen gibt es und desto grösser ist die Chance, die gesuchte Stelle im Dokument zu finden.

- Je kürzer das Suchwort, desto mehr mögliche Übereinstimmungen ergibt die Suche und desto ungenauer ist die Suche.

Um das Durchsuchen zu verbessern, wurden verschiedene Suchverfahren (oder *Suchalgorithmen*) entwickelt. Sie sollen möglichst schnell eine genaue Suche durchführen und ein passendes Resultat liefern. Diese Suchalgorithmen werden ständig weiterentwickelt und können riesige Datenmengen in sehr kurzer Zeit durchsuchen (z.B. Internetsuchmaschinen verwenden solche Suchalgorithmen).

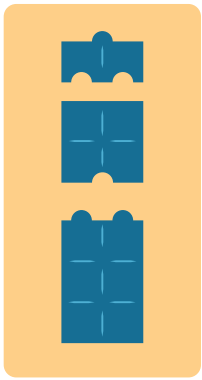
Biber Bausteine

Die Biber-Bausteine unterscheiden sich in vier Eigenschaften:

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

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

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

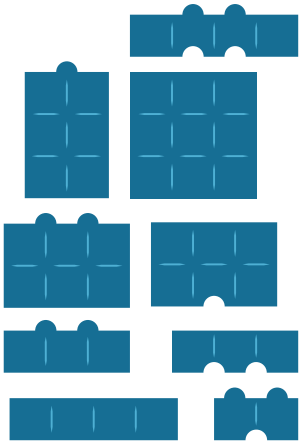


Rechts ist eine von Ottos Gruppen.

Denn diese drei Steine haben alle

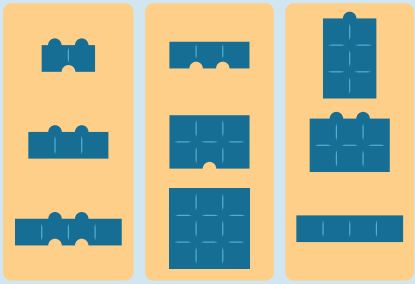
- die gleiche Breite,
- unterschiedliche Höhen,
- unterschiedlich viele Noppen und
- unterschiedlich viele Nuten.

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



So ist es richtig:

Die Steine sind so in Gruppen eingeteilt, wie Otto es machen würde.



Gruppe 1 Gruppe 2 Gruppe 3

Die Tabelle zeigt für die drei Gruppen, bei welchen Eigenschaften die Werte alle unterschiedlich bzw. alle gleich sind.

Eigenschaft	Gruppe 1	Gruppe 2	Gruppe 3
Breite	unterschiedlich	gleich	unterschiedlich
Höhe	gleich	unterschiedlich	unterschiedlich
Noppen	gleich	gleich	unterschiedlich
Nuten	unterschiedlich	unterschiedlich	gleich

Aber ist das die einzige Möglichkeit, die Steine so einzuteilen, wie Otto es machen würde? Man kann überlegen: Wenn für eine Eigenschaft die Werte in allen Gruppen unterschiedlich sein sollen, müssen die verschiedenen Werte in allen Steinen genau so oft vorkommen, wie es Gruppen gibt. Ist das nicht der Fall, muss es mindestens eine Gruppe geben, in der die Werte für diese Eigenschaft alle gleich sind. Ein genauer Blick auf alle Steine zeigt, dass bei der Breite die Werte schmal und breit jeweils nur zweimal vorkommen. Es muss also eine Gruppe geben, in der alle Steine die Breite mittel haben. Von den fünf Steinen mit mittlerer Breite gibt es keinen mit nur einer Noppe; deshalb kann keine Gruppe mit unterschiedlich vielen Noppen gebildet werden. Es gibt aber drei Steine mit null Noppen – und sie haben alle unterschiedliche Höhen und unterschiedlich viele Nuten. Damit ist Gruppe 2 die einzig mögliche Gruppe von Steinen mit mittlerer Breite. In den anderen beiden Gruppen müssen die Breiten alle unterschiedlich sein. In den verbleibenden sechs Steinen kommen bei der Höhe die Werte groß und mittel nur noch einmal vor. Es muss also eine Gruppe geben, in der alle Steine die Höhe klein haben. Gruppe 1 ist die einzig mögliche Dreier-Gruppe von Steinen mit kleiner Höhe, die Ottos Vorstellungen entspricht. Damit bleiben die drei Steine in Gruppe 3 übrig. Sie bilden ebenfalls eine Dreier-Gruppe, wie Otto sie auch bilden würde.

Das ist Informatik!

In dieser Biberaufgabe werden die Biber-Bausteine mit Hilfe von vier Eigenschaften *(oder Attributen)* beschrieben. Um die Bausteine wie Otto in Dreier-Gruppen aufteilen zu können, muss man für jeden Stein die Werte der Eigenschaften kennen.

Dazu genügt Menschen ein Blick auf jeden Baustein. Ein Computerprogramm, das die Dreier-Gruppen zusammenstellen soll, kann in der Regel nicht sehen und braucht eine Beschreibung in einer *Datenstruktur.* Zum Beispiel kann man die Steine in einer Datenbank als Zeilen einer Tabelle beschreiben. Die Spalten der Tabelle entsprechen den Eigenschaften, und in jeder Zeile (auch *Datensatz* genannt) stehen die Werte eines Bausteins in den passenden Spalten:

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

Der Entwurf von Datenbank-Tabellen gehört zu den üblichen Tätigkeiten für Informatikerinnen und Informatiker. Sie müssen dabei gründlich vorgehen und überlegen, welche Eigenschaften von Objekten für die Verarbeitung durch ein Computerprogramm wichtig sind. Nachträgliche Änderungen sind nicht so einfach, insbesondere wenn schon Daten vieler Objekte gespeichert sind.

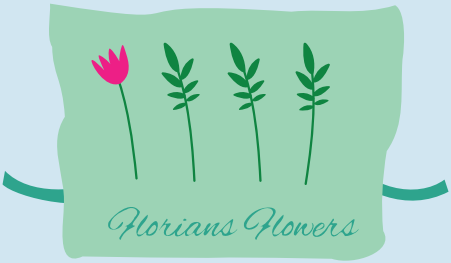
Blumenstrauß

Florian verkauft Blumensträuße. Jeden Blumenstrauß bindet Florian nach dieser Anleitung:

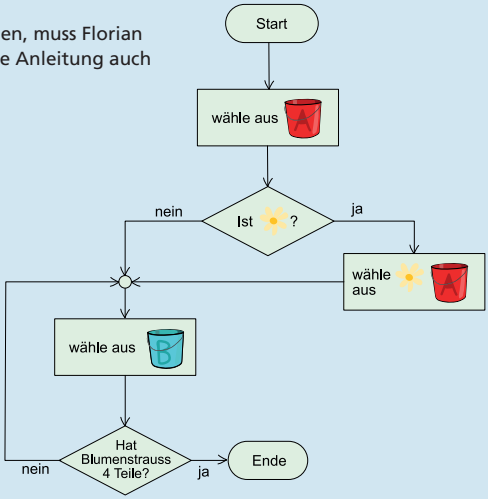
1. Nimm die erste Blume aus Eimer A.
2. Wenn die erste Blume eine Margarite ist, nimm noch eine Margarite .
3. Nun nimm solange einen Zweig aus Eimer B, bis der Blumenstrauß 4 Teile hat. Fertig!

Hilf Florian: Folge der Anleitung und wähle Blumen und Zweige für einen Strauß aus.

So ist es richtig:
Es gibt zwei richtige Lösungen:



Um die Blumensträuße korrekt zu binden, muss Florian die Anleitung befolgen. Wir können die Anleitung auch mit einem Diagramm beschreiben:



Nachdem Florian die erste Blume aus Eimer A gewählt hat, folgt eine Entscheidung, die abhängig von der ersten Blume ist. Entweder nimmt er noch eine Margarite oder folgt dem „nein"-Pfeil und nimmt einen Zweig .

Dann überprüft er, ob er schon vier Teile hat. Wenn nicht, folgt er dem „nein"-Pfeil und muss einen weiteren Zweig nehmen und dann die Anzahl der Teile wieder überprüfen.

Wenn er also zuerst eine Margarite nimmt, wird er noch eine Margarite nehmen und dann zweimal einen Zweig nehmen. Wenn er aber zuerst eine Tulpe nimmt, wird er danach direkt zu „wähle aus Eimer B" gehen aus Eimer B solange einen Zweig nehmen, bis er 4 Teile hat – also insgesamt 3 Zweige nehmen.

Das ist Informatik!
Die Anleitung fürs Binden des Blumenstraußes ist klar und könnte von einer Maschine ausgeführt werden. In der Informatik nennt man dies einen *Algorithmus.* Die Anleitung benutzt einige Anweisungen, die auch in Computerprogrammen üblich sind:

- Die erste Anweisung ist eine zufällige Auswahl aus einer Menge von Objekten.
- Die zweite Anweisung nennt man eine *bedingte Anweisung* (engl. *if-statement*) oder eine *Verzweigung:* Denn du musst aus zwei oder mehr Möglichkeiten auswählen.
- Die dritte Anweisung sieht relativ einfach aus, muss aber in einem Computerprogramm gut strukturiert werden. Der innere Teil der Anweisung (selbst wieder eine Anweisung: „Nimm einen Zweig aus Eimer B") muss mehrmals ausgeführt werden, bis der Blumenstrauß aus 4 Teilen besteht. Die Ausführung der inneren Anweisung wird also solange wiederholt, bis die Bedingungen „Der Blumenstrauß besteht aus 4 Teilen." erfüllt ist. Eine solche *Wiederholungs-Anweisung* nennt man auch *Schleife.*

Ein Algorithmus kann unterschiedlich dargestellt werden. In dieser Biberaufgabe ist Florians „Blumenstrauß-Algorithmus" als Anleitung in natürlicher Sprache formuliert. In der Lösungserklärung ist er als *Programmablaufplan* dargestellt.

Floristik ist eine Handwerkskunst. Es existieren Traditionen und Regeln, wie ein Blumenstrauß oder ein Kranz gebunden wird. Dies ist ein Beispiel dafür, dass Anleitungen oder Algorithmen in vielen Lebensbereichen vorkommen, nicht nur in der Informatik.

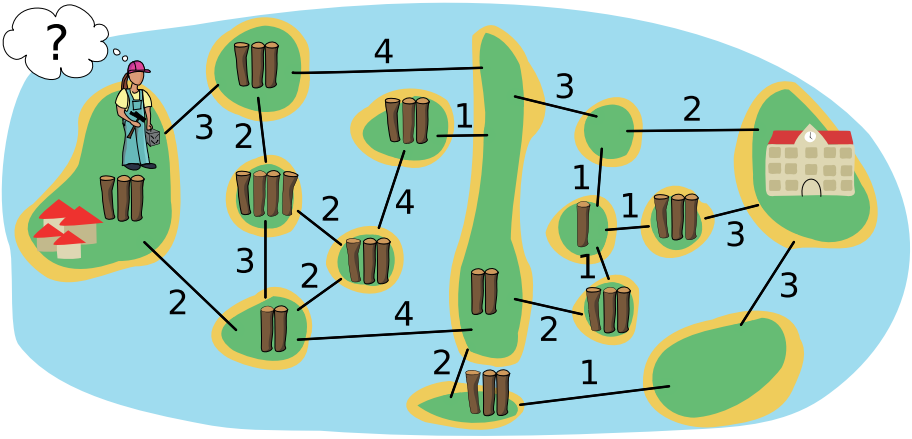
Brücken bauen!

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

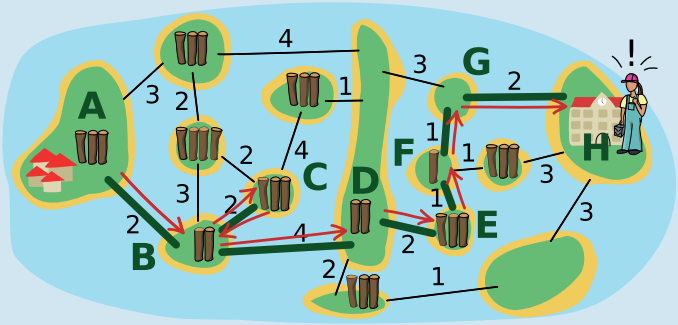
Die Insel-Karte zeigt, wie viele Baumstämme es auf jeder Insel gibt. Diese Baumstämme kann Bianca nehmen, um an den Linien Brücken zu bauen. Die Zahl an einer Linie sagt, wie viele Baumstämme dort für eine Brücke benutzt werden. Sobald es zwischen zwei Inseln eine Brücke gibt, kann Bianca darüber gehen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

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

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



So ist es richtig:



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

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

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

- Für die Brücken von Insel A bis Insel D hat Bianca 8 Stämme benutzt und kam ohne Stamm auf Insel D an. Wir notieren ihren Weg so: 2-[2,2]-4 (von der Insel A über die Linie mit der 2 zur Insel B, dann zwischen B und C hin und zurück über die 2, dann über die 4 zu Insel D. Ein Weg mit weniger Stämmen wäre 3-4, kann aber nur mit Umweg gebaut werden (3-[2,2]-4), verbraucht also 9 Stämme, wobei Bianca auf Insel D mit einem Stamm im Vorrat ankommt. Alle anderen Wege von Insel A bis D verbrauchen 9 Stämme oder mehr.
- Für die Brücken von Insel D bis H hat Bianca 6 Stämme benutzt. Den direkten Weg 3-2 kann sie nicht bauen, auch nicht mit einem Stamm im Vorrat. Alle anderen Wege von Insel D zu Insel H verbrauchen 6 Stämme oder mehr.

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

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

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

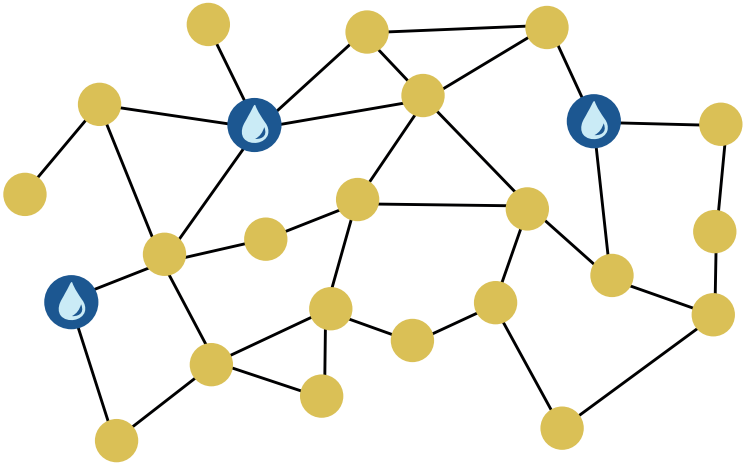
Brunnen

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

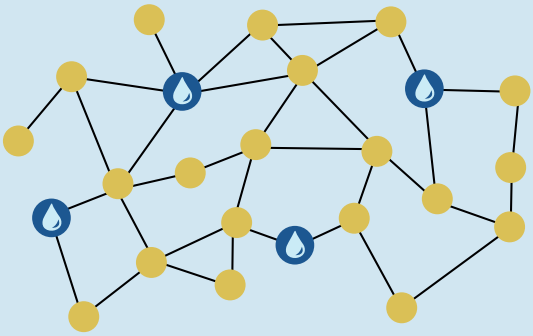
Die Brunnen sollen so stehen, dass man von jeder Straßenecke aus höchstens zwei Straßenabschnitte gehen muss, um einen Brunnen zu erreichen. Dann ist die Bürgermeisterin zufrieden.

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

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



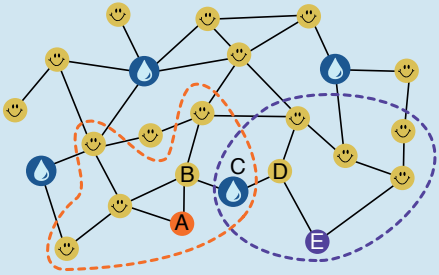
So ist es richtig:



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

Wie können wir herausfinden, an welcher Straßenecke ein weiterer Brunnen aufgestellt werden soll?

Im Stadtplan markieren wir alle Straßenecken mit einem ☺, die höchstens zwei Straßenabschnitte von einem der Brunnen entfernt sind, die bereits aufgestellt sind. In Bezug auf diese Ecken kann die Bürgermeisterin bereits zufrieden sein.



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

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

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

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

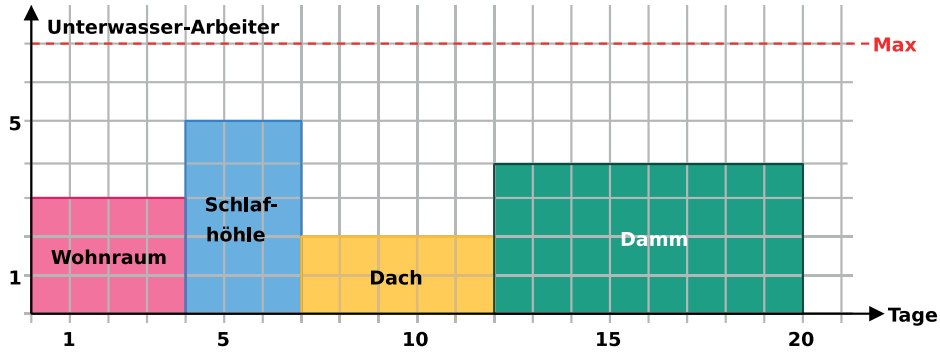
Solche „minimum distance *k*-dominating sets“ spielen in der letzten Zeit eine größere Rolle, insbesondere im Bereich des *Social Computing* (auf Deutsch auch *Sozioinformatik*): Zur automatischen Verarbeitung von Daten über soziale Netzwerke (etwa um die Verbreitung von Fake News zu erkennen) werden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

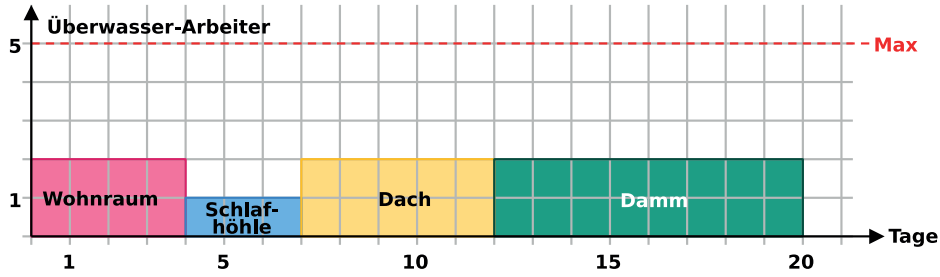
Burgenbau

Eine Biberburg besteht aus 4 Teilen. Bei jedem Teil wird gleichzeitig über und unter Wasser gearbeitet. Die beteiligten Biber sind aber spezialisiert: jeder arbeitet entweder nur unter Wasser oder nur über Wasser.

Für den Bau einer neuen Burg stehen höchstens 7 Unterwasser-Arbeiter und 5 Überwasser-Arbeiter zur Verfügung. Sie können auch gleichzeitig verschiedene Teile bauen. Wichtig: Das Dach kann erst gebaut werden, wenn die Schlafhöhle fertig ist!
Bei allen anderen Teilen ist die Reihenfolge egal.

Hier ist ein Arbeitsplan, mit dem die Biberburg nach 20 Tagen fertig wird. Der Plan zeigt für jedes Teil, wie lange dessen Bau dauert und wie viele Arbeiter unter und über Wasser dafür benötigt werden. Beim Wohnraum zum Beispiel arbeiten 3 Biber unter und 2 Biber über Wasser und sind nach 4 Tagen fertig.

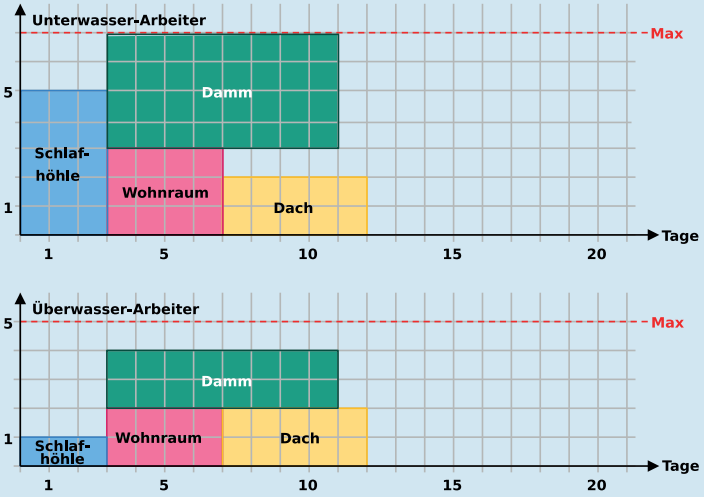




Überlege dir einen Plan, mit dem die Biberburg nach möglichst wenigen Tagen fertig wird. Wie viele Tage sind das?

12 ist die richtige Antwort:

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



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

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

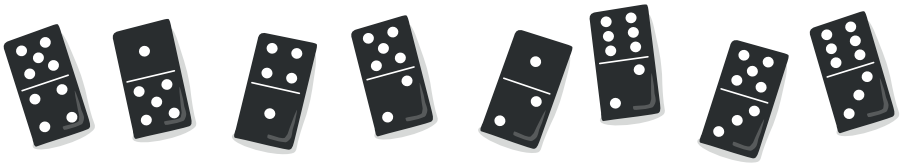
Das ist Informatik!
Einen optimalen, möglichst zügigen Ablauf eines Projekts zu planen, ist eine schwierige Aufgabe, bei der einige Bedingungen zu berücksichtigen sind. Zwischen Teilaufgaben eines Projekts bestehen oft zeitliche Abhängigkeiten; z.B. kann es Teilaufgaben geben die erst nach Beendigung einer anderen Teilaufgabe begonnen werden können – wie hier bei Dach und Schlafhöhle. Außerdem braucht jede Teilaufgabe bestimmte Ressourcen wie Arbeitskraft, Zeit und Geräte. Bei der Erstellung von Projektplänen hilft es, wenn man den Plan gut darstellen kann. Die in dieser Biberaufgabe gezeigten Diagramme sind eine Art von Gantt-Diagrammen, die von Henry Gantt (1861-1919) zwischen 1910 und 1915 entwickelt wurden; ähnliche Darstellungen wurden unabhängig von Gantt zur gleichen Zeit auch in Deutschland verwendet. Sie zeigen die Nutzung von Ressourcen (in diesem Fall die beiden Arten von Arbeitskräften) im Zeitverlauf.

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

Domino

Jeder Dominostein hat zwei Felder. Auf jedem Feld sind 1 bis 6 Punkte.

Du hast diese acht Steine:

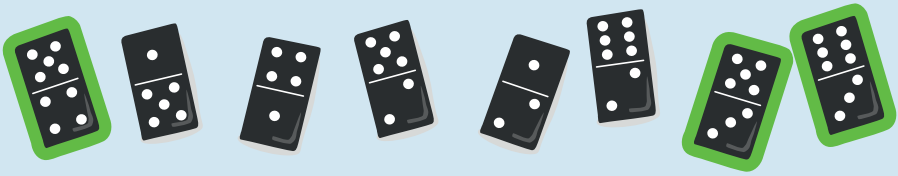


Alle acht Steine sollst du so in eine Reihe legen, dass auf den angrenzenden Feldern zweier benachbarter Steine immer gleich viele Punkte sind.

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

Welche Steine sind das?

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



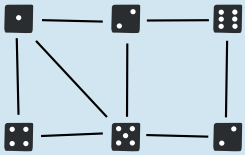
Um die Aufgabe zu lösen, untersuchen wir die Augenzahlen (die Punkte auf Dominosteinen nennt man auch Augen, wie bei einem Würfel) der 16 Felder der Dominosteine. Wir halten fest, wie häufig die einzelnen Augenzahlen vorkommen, und ob die Häufigkeit eine gerade oder eine ungerade Zahl ist:

	3	ungerade
	3	ungerade
	2	gerade
	2	gerade
	4	gerade
	2	gerade

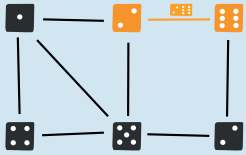
Felder mit Augenzahlen, die mit gerader Häufigkeit vorkommen, müssen paarweise mitten in der Reihe liegen oder gleichzeitig an Anfang und Ende. Felder mit Augenzahlen, die mit ungerader Häufigkeit vorkommen, können aber nicht alle mitten in der Reihe liegen: Es kann nämlich nicht für jedes Feld mit dieser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

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

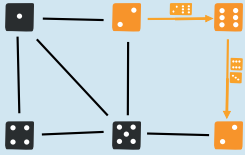
Das ist Informatik!
Es gibt mehrere Möglichkeiten, die acht Dominosteine dieser Biberaufgabe in eine korrekte Reihe zu legen. Um sich hier eine bessere Übersicht zu verschaffen, verwenden Informatiker sogenannte *Graphen:*



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



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



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

Ein besonderer Baum

Jona hat einen besonderen Apfelbaum im Garten:

- Landet ein Vogel  auf dem Baum, wachsen sofort zwei neue Äpfel.

- Klettert ein Eichhörnchen  auf den Baum, fällt ein Apfel runter.
  Wenn kein Apfel am Baum hängt, passiert nichts.

- Besucht eine Schlange  den Baum, verschwinden alle Äpfel sofort.

Heute Morgen hängen 25 Äpfel am Baum.
Dann besuchen einige Tiere nacheinander den Baum, zuletzt ein Eichhörnchen.
Jona hat ihre Reihenfolge genau aufgeschrieben:

Wie viele Äpfel hängen danach am Baum?

A) 3 Äpfel **B)** 7 Äpfel **C)** 17 Äpfel **D)** 31 Äpfel

Antwort B ist richtig.
Nachdem das letzte Eichhörnchen auf den Baum klettert, hängen noch 7 Äpfel am Baum.

Da alle Äpfel sofort verschwinden, wenn eine Schlange den Baum besucht, genügt es, nur die Tiere zu betrachten, die nach der zweiten (und letzten) Schlange den Baum besuchen. Nach dem Besuch dieser Schlange hängen 0 Äpfel am Baum. Dann landen nacheinander vier Vögel auf dem Baum; anschließend hängen am Baum $4 \times 2 = 8$ Äpfel. Zum Schluss klettert noch ein Eichhörnchen auf den Baum: Ein Apfel fällt herunter, und $8 - 1 = 7$ Äpfel bleiben übrig.

Das ist Informatik!
Der Besuch eines Tieres verändert den Zustand des besonderen Apfelbaums – aber nur auf ganz bestimmte Weise: Nur die Anzahl der Äpfel, die am Baum hängen, wird geändert. Auf andere Eigenschaften des Baumes, etwa die Anzahl der Blätter, die Länge einzelner Äste oder die Form der Baumkrone, hat der Besuch eines Tieres keinen Einfluss. Für diese Biberaufgabe ist es also ausreichend, die Anzahl der Äpfel zu betrachten.
Auch ein Computerprogramm hat einen Zustand, der von den einzelnen Anweisungen des Programms verändert wird. Als Zustand werden meist die von einem Programm gespeicherten Daten betrachtet; diese Daten speichert das Programm in den bei der Programmierung eingeführten *Variablen*.
Die Folge der Tierbesuche auf dem Baum in dieser Biberaufgabe ist wie ein Computerprogramm: Jeder Tierbesuch ist eine Anweisung, die den Zustand des Apfelbaums verändert. Dieser Zustand – also die Anzahl der Äpfel, siehe oben – kann in einer einzigen Variable gespeichert werden.
Beim Lösen der Aufgabe ist dir vielleicht aufgefallen, dass du nicht das ganze „Programm“ anschauen musstest, sondern nur den Teil nach dem letzten Vorkommen der Schlange. Durch genaue Betrachtung der Auswirkungen der einzelnen Anweisungen auf den Zustand des Programms konntest du eine besondere Eigenschaft des Programms herausfinden. Eine solche Analyse von (Computer-)Programmen gehört zu den häufigen Tätigkeiten von Informatikerinnen und Informatikern.

Emma erledigt

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

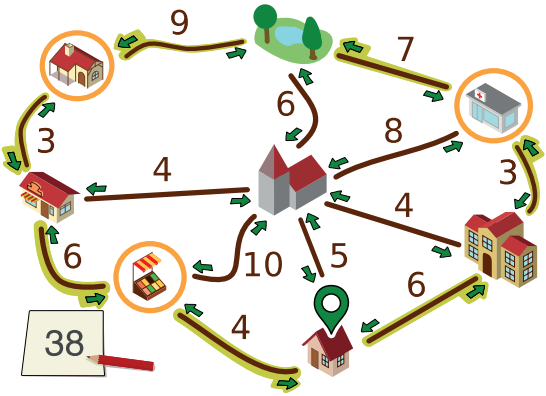
- beim Kiosk ein Päckchen abholen,
- auf dem Markt Obst kaufen und
- in der Apotheke ein Medikament besorgen.

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

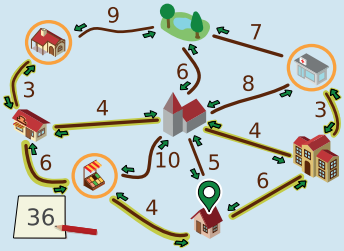
Auf einem Plan hat Emma eingetragen, wie viele Minuten sie für die Strecken zwischen einzelnen Orten ihrer Stadt benötigt. Außerdem hat sie im Plan markiert, welche Strecken sie auf ihrem Weg geht und in welcher Richtung. Für diesen Weg benötigt Emma insgesamt 4 + 6 + 3 + 9 + 7 + 3 + 6 = 38 Minuten.

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

Bestimme den kürzesten Weg, den Emma gehen kann, um ihre drei Aufgaben zu erledigen. Welche Strecken geht sie dazu in welcher Richtung?



So ist es richtig:



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

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

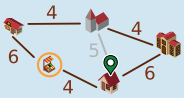
Nun wollen wir begründen, warum es keinen noch kürzeren Weg geben kann. Dazu benutzen wir eine vereinfachte Darstellung des Plans.

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

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

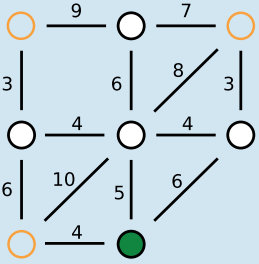
Emma muss zur Apotheke und zum Kiosk gehen. Dorthin kommt sie jeweils nur von der Bäckerei bzw. der Schule. Sie muss jeweils die Strecke zwischen diesen Orten hin- und her gehen. Das dauert jeweils 3 + 3 = 6, ingesamt also 12 Minuten. Das merken wir uns und fassen nun die beiden Orte oben mit denen darunter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

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



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

Diese Darstellung enthält alle für Emmas Weg wichtigen Informationen, nämlich
- Objekte: die Orte, wobei die für den Weg wichtigen Orte markiert sind;
- und Beziehungen zwischen den Objekten: die Strecken zwischen den Orten, für die jeweils eine Länge angegeben ist.



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

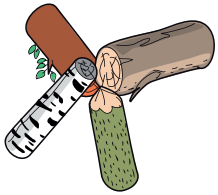
Die Informatik interessiert sich für Fragen, die in Bezug auf Graphen gestellt werden können, und für Algorithmen, mit denen man die Fragen beantworten kann. Eine für gewichtete Graphen bedeutsame Frage lautet: Was ist der kürzeste (oder schnellste) Weg zwischen zwei Knoten? Die „Graphen-Frage" in dieser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

Foto

Der Biber hat gerade ein Foto gemacht.

Welches der vier Fotos ist es?

A



B

C

D

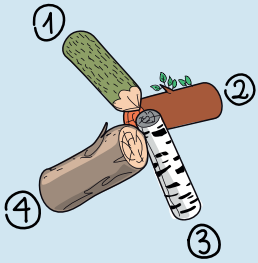
Antwort D ist richtig:

Die Baumstämme, die der Biber fotografiert hat, sind im Kreis angeordnet. Um herauszufinden, welches Foto das richtige ist, betrachten wir die Reihenfolge der Baumstämme in dieser Anordnung. Wir wählen einen Baumstamm aus (z.B. den angespitzten Baumstamm) und geben ihm die Nummer 1. Dann bestimmen wir, welcher Baumstamm links daneben ist und geben ihm die Nummer 2. Das machen wir solange, bis alle Baumstämme eine Nummer haben. In der Situation, die der Biber fotografiert hat, haben die Stämme also diese Reihenfolge: 1 (angespitzter Stamm) – 2 (brauner Stamm mit Blättern) – 3 (Birkenstamm) – 4 (dicker brauner Stamm).

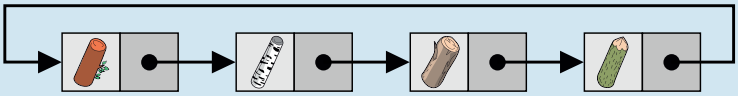
Nun betrachten wir die Reihenfolge der Stämme in den Fotos A bis D. Dabei beginnen wir wie oben mit dem angespitzten Baumstamm 1 und gehen immer nach links:

- Foto A: 1 – 3 – 2 – 4
- Foto B: 1 – 4 – 3 – 2
- Foto C: 1 – 3 – 4 – 2
- Foto D: 1 – 2 – 3 – 4

Nur Foto D zeigt die richtige Reihenfolge.



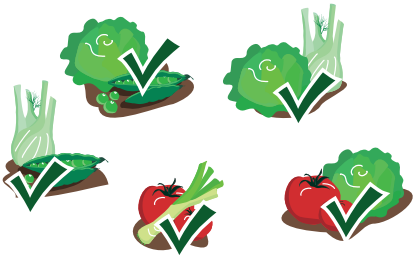
Das ist Informatik!
In dieser Biberaufgabe wird die Reihenfolge der Baumstämme betrachtet. Was bei wenigen *Elementen* (hier vier Baumstämmen) durch einfaches „Hinsehen" und Vergleichen der Nachbarpaare möglich ist, erfordert bei Problemen mit viel mehr Elementen ein automatisiertes Vorgehen. In einem Computerprogramm, das benachbarte Elemente verarbeiten soll, könnten die Elemente in einer geeigneten Datenstruktur wie einer verketteten Liste gespeichert werden:

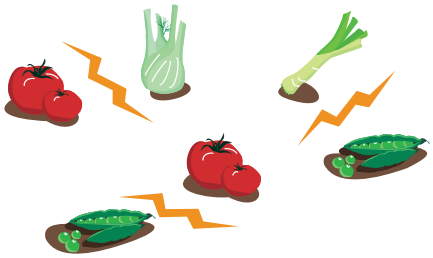


In einer *verketteten Liste* wird jedes Datenelement in einem einzelnen Knoten gespeichert. Zusätzlich ist in jedem Knoten ein *Verweis* auf den nächsten Knoten in der Liste gespeichert. Enthält der letzte Knoten einen Verweis auf den ersten Knoten, so handelt es sich um eine ringförmige Datenstruktur. Das ist im Beispiel wichtig, damit man bei jedem beliebigen Baumstamm starten und die Liste durchlaufen kann.

Gemüsebeet

Lisa legt ein Gemüsebeet an, mit sechseckigen Bereichen.
Darauf will sie Gemüse pflanzen, in jeden Bereich eines.
Es gibt fünf Sorten Gemüse.
Manche Gemüse vertragen sich gut miteinander (✓), andere nicht ( ):

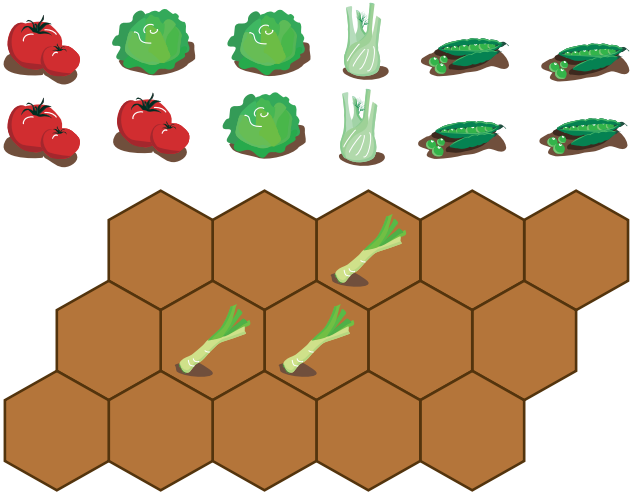




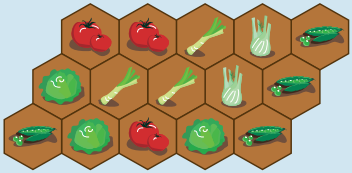
Beim Pflanzen beachtet Lisa folgende Regel: Gemüse, die sich nicht vertragen, dürfen nicht in Bereiche gepflanzt werden, die sich berühren.

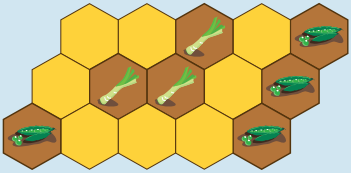
In drei Bereiche hat Lisa schon Lauch gepflanzt.

Bepflanze alle noch freien Bereiche und beachte Lisas Regel!



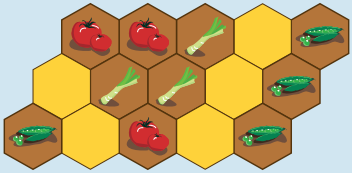
So ist es richtig:



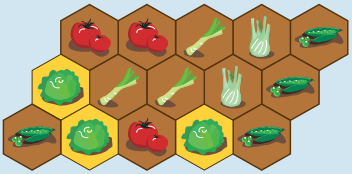


Weil Erbsen sich mit Lauch nicht vertragen, pflanzt Lisa keine Erbsen in die gelben Bereiche. Für die Erbsen bleiben nur die übrigen Bereiche.

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

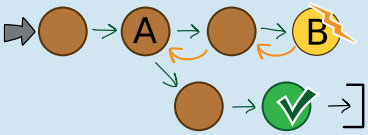


Weil Tomaten sich mit Fenchel nicht vertragen, pflanzt Lisa keinen Fenchel in die gelben Bereiche. Den Fenchel kann sie in die beiden Bereiche zwischen Lauch und Erbsen pflanzen. In die gelben Bereiche kann sie Salat pflanzen: Für Salat ist Lisa keine Unverträglichkeit bekannt.



Das ist Informatik!

Wer Gemüse so pflanzen will, dass die Ernte möglichst groß wird, muss viele Bedingungen beachten: Die einzelnen Sorten haben zum Beispiel unterschiedlichen Bedarf an Platz, Nährstoff und Licht. In dieser Biberaufgabe betrachten wir nur eine Art von Bedingung: die Verträglichkeit zwischen den Gemüsesorten. Um eine Bepflanzung von Lisas Beet zu finden, die alle Verträglichkeitsbedingungen beachtet, könnte man so vorgehen: Man probiert systematisch alle Kombinationen aus, die Gemüse auf dem Beet zu platzieren. Erst wenn das Beet voll ist, wird geprüft, ob diese Kombination alle Bedingungen erfüllt und eine Lösung für Lisas Problem ist. In der Informatik ist solch ein Ausprobieren aller Kombinationen als *Brute-Force*-Methode bekannt. Bei Problemen mit vielen Kombinationen und nur wenigen Lösungen kann ein Vorgehen nach dieser Methode sehr lange dauern. Deshalb ist es meist besser, schrittweise vorzugehen und bei jedem Schritt alle Bedingungen zu berücksichtigen. Auf diese Weise haben wir die Lösung für Lisas Problem gefunden, und eine „falsche" Kombination bzw. Bepflanzung des Beets konnte gar nicht entstehen.



Zum Glück ließ sich die Lösung auf direktem Weg finden: Es gab immer Bereiche, in die wir einige der noch übrigen Gemüse pflanzen konnten. Das gelingt im Allgemeinen nicht immer. Wenn man versucht, die Lösung schrittweise zusammenzusetzen, kann es bei einem Schritt A mehrere Möglichkeiten geben, alle Bedingungen zu erfüllen. Je nach Wahl kann es bei einem späteren Schritt B keine Möglichkeit mehr geben. Dann nimmt man die letzten Schritte solange zurück, bis man beim Schritt A mit den mehreren Möglichkeiten wieder angekommen ist. Dort wählt man eine andere Möglichkeit und versucht damit eine Lösung zu finden. In der Informatik ist diese Rücknahme von Schritten als *Backtracking* bekannt.

Go-Bots

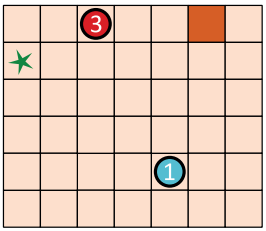
Die Go-Bots sind sehr einfache Roboter. Sie fahren über ein Spielbrett mit Feldern.

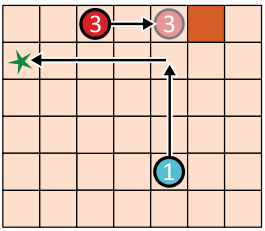
Um sie zu steuern, wählt man zunächst einen der Go-Bots aus. Den schickt man dann mit einem Pfeil-Befehl in eine Richtung: hoch, runter, links oder rechts.

Der Go-Bot fährt dann stur geradeaus, bis er direkt vor einem Hindernis oder einem anderen Roboter ankommt. Dort bleibt er stehen, bis er einen neuen Befehl bekommt.

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

Unten links ist ein Spielbrett mit zwei Go-Bots. Mit dieser Befehlsfolge erreicht Go-Bot 1 das Ziel – siehe unten rechts:

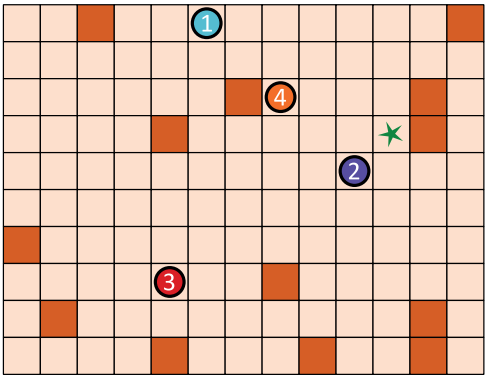




Unten ist ein anderes Spielbrett mit vier Go-Bots.

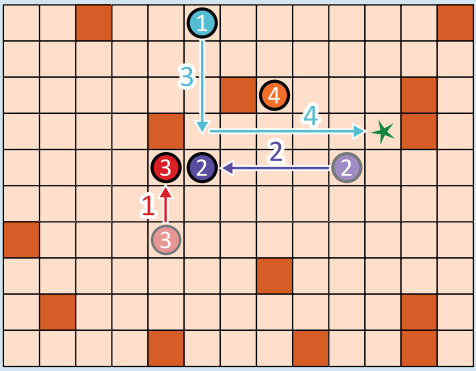
Erstelle eine Befehlsfolge mit vier Pfeilen, mit der Go-Bot 1 das Ziel erreicht!

Wähle immer abwechselnd einen Go-Bot und einen Pfeil aus, um die Befehlsfolge zu erstellen.



So ist es richtig: 3 ↑ 2 ← 1 ↓ 1 →

Damit Go-Bot 1 durch eine Befehlsfolge mit vier Pfeilen das Ziel erreichen kann, müssen drei Go-Bots kooperieren. Zuerst geht 3 nach oben, bis er vor einem Hindernis stehen bleibt. Damit wird er selbst zum Hindernis für 2 auf seinem Weg nach links.
Wenn man nun 1 nach unten schickt, geht er bis 2 und kann von dort aus nach rechts gehen, wo er vor dem Hindernis stehen bleibt – auf dem Ziel.



Wie findet man die richtige Befehlsfolge? Man kann hinten anfangen und sich überlegen, was die letzte Bewegung von Go-Bot 1 zum Ziel sein muss. Es gibt nur zwei Möglichkeiten:

(a) Er kommt von links, wie in unserer Lösung.

(b) Er kommt von oben. In diesem Fall müsste Go-Bot 4 mit drei Befehlen nach oben rechts bewegt werden, um für 1 als Hindernis zu dienen. Wir benötigten dann 3 + 2 = 5 Pfeil-Befehle. Gesucht ist aber eine Folge mit vier Befehlen. Also muss Möglichkeit a) korrekt sein und Go-Bot 1 kommt von links zum Ziel. Dann geht die vorletzte Bewegung des Go-Bots 1 von oben nach unten. Damit er an der richtigen Stelle stehen bleibt, müssen zuvor 2 und 3 wie im Bild bewegt werden.

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

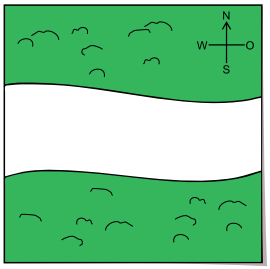
Aufgabenverteilung ist ein wichtiger Aspekt der Robotik. Zum Beispiel arbeiten in einem automatisierten Warenlager unterschiedliche Roboter zusammen, um Waren einzulagern, wiederzufinden und zu transportieren. Dabei werden alle Aktivitäten so koordiniert, dass möglichst wenig nutzlose Ruhezeiten entstehen,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

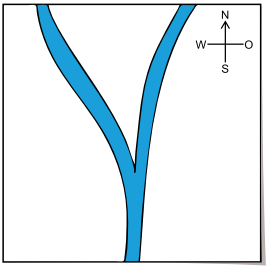
Ein besonderes Gebiet der Robotik sind Schwarmroboter. Das sind – wie die Go-Bots – einfache Maschinen, die in einer großen Gruppe gemeinsam eine Aufgabe lösen. In der Landwirtschaft können inzwischen Schwärme von Robotern die Aussaat von Mais erledigen, die Entwicklung der Pflanzen und die Bodenbeschaffenheit beobachten und schließlich sogar das Getreide ernten. Jeder Schwarmroboter ist klein und einfach konstruiert, aber der Schwarm als Ganzes kann Großes leisten. Dieses Prinzip gilt auch für Multiagentensysteme: Das sind einfache Softwareeinheiten, die gemeinsam komplexe Probleme lösen können. Die Aufgabe der Informatik ist, Algorithmen für eine optimale Koordination und Kooperation von Gesamtsystemen mit mehreren Akteuren – ob Hardware oder Software – zu entwickeln.

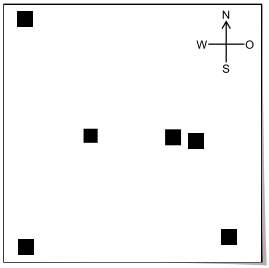
Karlas Traumhaus

Karla hat drei Karten, die alle genau das gleiche Gebiet zeigen. Eine Karte zeigt die Wälder, eine die Flüsse und eine die Häuser in diesem Gebiet. Karlas Traumhaus liegt im Wald und in der Nähe eines Flusses.

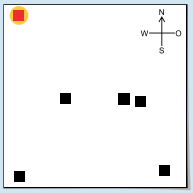
Welches ist Karlas Traumhaus?



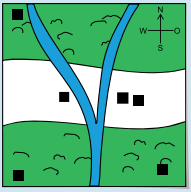




So ist es richtig:
Das Haus oben links auf der Hauskarte ist Karlas Traumhaus.



Um Karlas Traumhaus zu finden, müssen die Informationen aus allen drei Karten ausgewertet werden. Das Traumhaus muss sich in einem Waldgebiet und in der Nähe eines Flusses befinden. Das trifft nur auf das Haus oben links zu. Dies ist leicht zu erkennen, wenn die Karten übereinander gelegt werden:



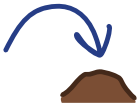
Das ist Informatik!
Wenn die Informationen über die Wälder, die Flüsse und die Häuser auf einer einzigen Karte dargestellt sind, ist es einfach, das gesuchte Haus zu finden.

Ein *Geoinformationssystem* (GIS) führt eine Vielzahl räumlicher Informationen (z.B. Wälder, Straßen, Landesgrenzen, Tankstellen, Rathäuser, Überschwemmungsgebiete usw.) zusammen und stellt diese auf einer Karte dar. Ein GIS dient also der Visualisierung und Analyse sogenannter *Geodaten.* Mit Hilfe eines GIS ist es z.B. für Katastrophenschutzbeauftragte möglich, Informationen für Evakuierungspläne zusammenzustellen.

Die Verwendung mehrerer Ebenen mit unterschiedlichen Bildinformationen ist auch aus Grafikprogrammen bekannt. Eine wichtige Frage ist immer, welche Ebene mit den darin enthaltenen Objekten die oberste ist und deshalb im Vordergrund dargestellt wird. Im Beispiel sollte die Hauskarte die oberste Ebene sein, damit die Häuser nicht von den Waldflächen verdeckt werden.

Karotten pflanzen

Der Kaninchenroboter kann folgende Anweisungen ausführen:

	Springe nach **links** auf den nächsten Hügel.
	Springe nach **rechts** auf den nächsten Hügel.
	**Pflanze** einen Karottensamen auf dem Hügel, auf dem du stehst.

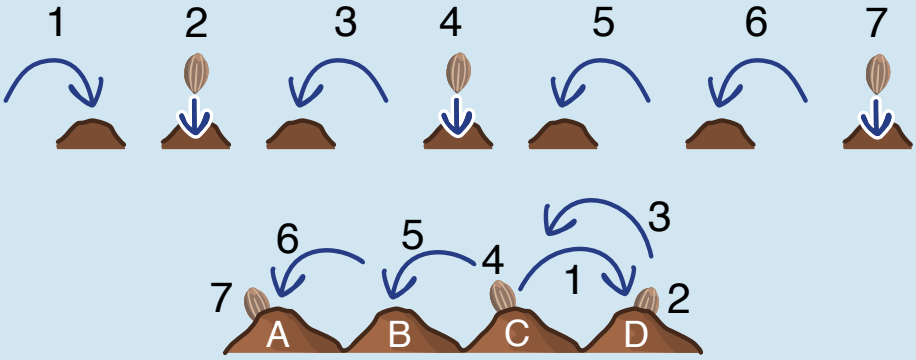
Der Kaninchenroboter hat diese Folge von Anweisungen ausgeführt:

Dabei ist der Roboter auf vier Hügeln gewesen. Wir wissen aber nicht, auf welchem Hügel er angefangen hat.

Auf welche Hügel hat der Roboter die Karottensamen gepflanzt?

So ist es richtig:

Um die richtige Antwort besser erklären zu können, geben wir den Hügeln Buchstaben (siehe oben) und den Anweisungen Nummern:



Zuerst bestimmen wir den Startpunkt des Roboters: Da der Roboter dreimal hintereinander nach links springt (Anweisungen 3, 5, 6), muss er vorher auf Hügel D stehen. Bevor er dreimal nach links springt, springt er einmal nach rechts (Anweisung 1). Der Roboter hat also auf Hügel C angefangen. Folglich werden die Karottensamen – den Anweisungen 2, 4 und 7 entsprechend – zuerst auf Hügel D, dann auf Hügel C und zuletzt auf Hügel A gepflanzt.

Das ist Informatik!
Echte Roboter haben eingebaute Computer, und die werden so ähnlich *programmiert* wie der Kaninchenroboter. Ein Computerprogramm besteht aus vielen einzelnen *Anweisungen.*

In unserem Fall wird die Abfolge der Anweisungen für den Roboter-Computer mit Hilfe von Bildblöcken angegeben. Das Ergebnis *(Output)* des Programms hängt nicht nur von der Startposition *(Input),* sondern auch von der Folge und Reihenfolge der Anweisungen ab.

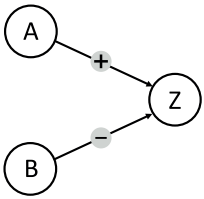
Diese Biberaufgabe zeigt ein Beispiel für den Einsatz von Robotern in der Landwirtschaft. Roboter können nicht nur pflanzen, sondern auch bewässern, bestäuben oder Pflanzenschutzmittel gezielt verteilen.

Konflikt-Detektor

Anna und Ben wollen einen „Konflikt-Detektor“ bauen, der anzeigt, ob sie unterschiedlicher Meinung sind.

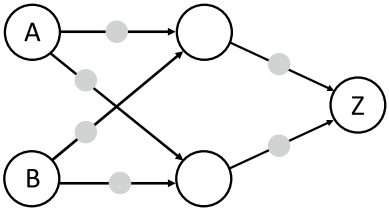
Sie verwenden Einheiten, die in zwei Zuständen sein können: Ja und Nein. Zwei Einheiten können mit einem Kabel verbunden werden. Wenn eine Einheit im Zustand Ja ist, sendet sie über alle ausgehenden Kabel ein Signal; ist sie im Zustand Nein, sendet sie kein Signal. Die Kabel werden so eingestellt, dass sie ein Signal als positives (+) oder negatives (–) Signal an die rechts angeschlossene Einheit übermitteln. Eine angeschlossene Einheit geht in den Zustand Ja, wenn sie mehr positive als negative Signale empfängt, und sonst in den Zustand Nein. Als Eingabe setzt Anna den Zustand der Einheit A und Ben den Zustand der Einheit B.

Zuerst bauen Anna und Ben diese Maschine:



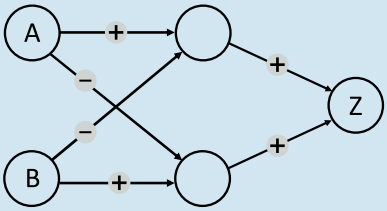
Sie bemerken, dass die Einheit Z nur dann Ja ist, wenn A Ja und B Nein ist. Das ist nicht das, was sie wollen.

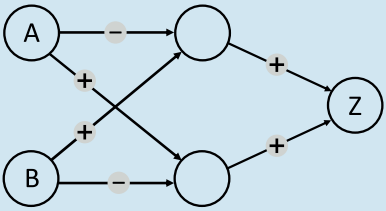
Dann bauen Anna und Ben eine größere Maschine (unten im Bild) und sind sicher, dass sie der Konflikt-Detektor sein kann: Z soll nur dann Ja sein, wenn A und B in unterschiedlichen Zuständen sind (Ja und Nein bzw. Nein und Ja). Ansonsten soll Z im Zustand Nein sein. Jetzt müssen nur noch die Kabel richtig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



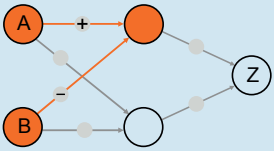
Stelle für jedes Kabel ein, ob es ein Signal positiv (+) oder negativ (–) übermittelt, damit der Konflikt-Detektor korrekt arbeitet.

So ist es richtig:
Mit diesen beiden Einstellungen arbeitet der Konflikt-Detektor korrekt.



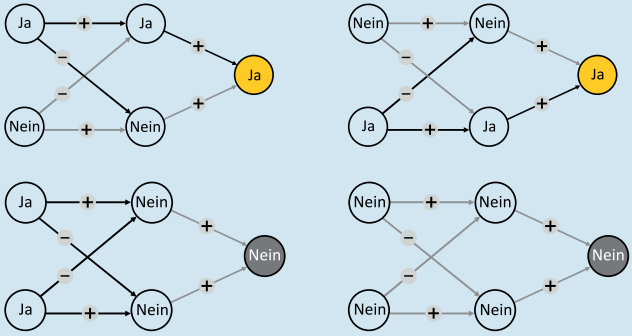


Im Konflikt-Detektor muss die Ausgabe-Einheit genau bei zwei unterschiedlichen Eingaben (A = Ja und B = Nein sowie A = Nein und B = Ja) auf Ja sein. Z kann nur Ja sein, wenn über die zwei eingehenden Kabel mehr positive als negative Signale ankommen. Mindestens eines der Kabel muss also ein positives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



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

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



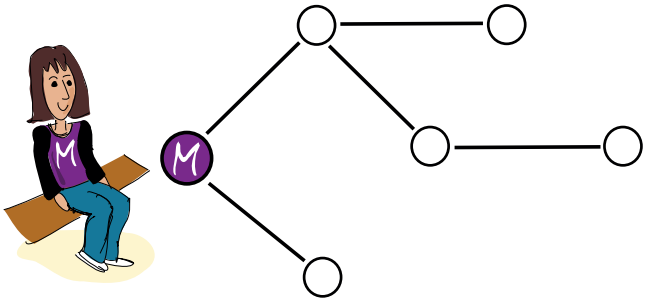
Das ist Informatik!

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

Martinas Dorf

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

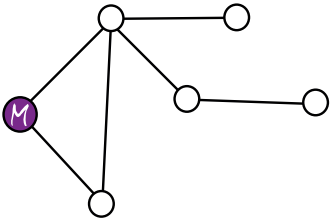
Martina hat eine besondere Karte des Dorfs gezeichnet. Sie hat darin Wege eingezeichnet, über die sie am schnellsten zu den anderen Häusern gehen kann.



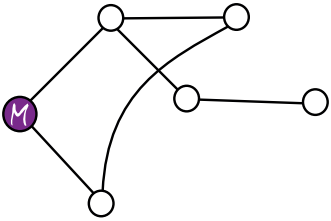
Natürlich gibt es auch eine richtige Karte des Dorfs, mit allen Wegen.

Welche dieser Zeichnungen kann nicht die richtige Karte sein?

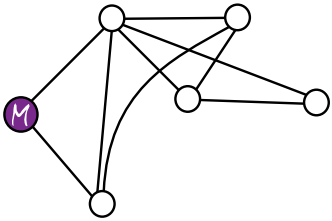
A



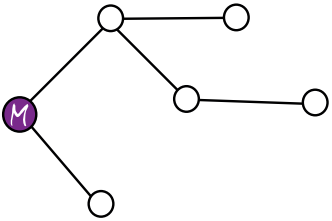
B



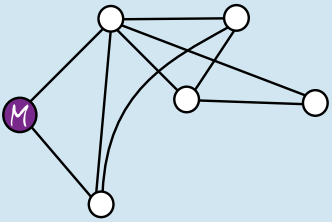
C



D



Antwort C ist richtig:



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

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

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

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

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

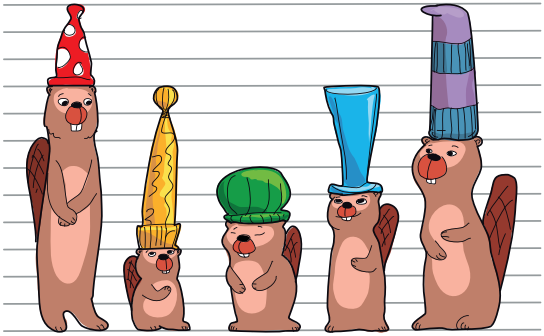
- Alle Knoten sind direkt (über eine Kante) oder indirekt (über mehreren Kanten) miteinander verbunden.
- Egal welche zwei Knoten man zufällig auswählt, es gibt immer nur genau einen Weg zwischen den beiden.

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

In der Informatik gibt es viele Anwendungen für Graph-Algorithmen, vor allem im Zusammenhang mit Netzwerken (Verkehrsnetze, Telekommunikationsnetze …), etwa bei der Berechnung von Routen in Navigationssystemen. Spannbäume können beim Aufbau kostengünstiger Netze eingesetzt werden und hilfreich bei der Lösung besonders schwieriger Probleme sein.

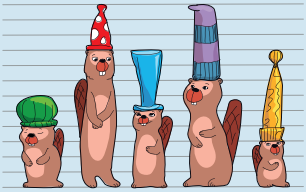
Neue Hüte

Die Biber haben neue Hüte.
Die Hüte sind unterschiedlich hoch.

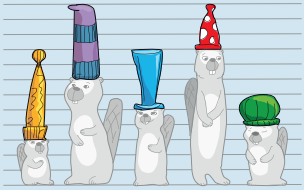


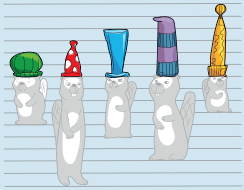
Sortiere die Hüte nach der Höhe. Der Biber mit dem kürzesten Hut soll links stehen.

So ist es richtig:
So sind die Hüte richtig sortiert; sie werden von links nach rechts immer höher:



Beim Umordnen der Biber achten wir nur auf die Hüte. Dann ist es viel einfacher, sie der Höhe nach zu sortieren.





Das ist Informatik!
Aus einer großen Menge von Dingen können wir bestimmte Dinge besser heraussuchen, wenn die Menge sortiert ist: Wenn Werkzeuge nach Größe sortiert sind, lässt sich das passende Werkzeug leichter finden. Weil die Einträge in einem Wörterbuch alphabetisch sortiert sind, kann man die Seite mit dem gesuchten Wort schneller finden. In dieser Biberaufgabe sollst du die Biber sortieren, und zwar nach der Höhe ihrer Hüte. Die Schwierigkeit dabei ist, dass die Eigenschaft „Höhe der Hüte“ nicht leicht erkennbar ist. Deshalb muss man sehr genau hinsehen.

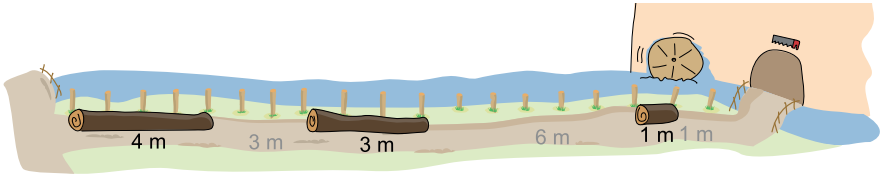
Beim Sortieren ist es also erstens wichtig, die Eigenschaft klar festzulegen, nach der sortiert werden soll. Zweitens müssen die Werte dieser Eigenschaft sortierbar sein. Nach Eigenschaften, die in Zahlen ausgedrückt werden (wie z.B. Höhe, Länge, Gewicht, ...) kann man sortieren: Für zwei verschiedene Zahlen können wir sagen, welche Zahl die kleinere ist. Wörter kann man sortieren, weil die Reihenfolge der Buchstaben im Alphabet festgelegt ist und deshalb für zwei verschiedene Wörter klar ist, welches weiter vorne im Wörterbuch stehen muss. Allgemein kann man sagen: Eine Eigenschaft ist sortierbar, wenn man für ihre einzelnen Werte eine eindeutige „kleiner als“-Beziehung (eine *Ordnung*) angeben kann.

Mit Computern werden große Datenmengen verwaltet. Um darin einzelne Daten schnell finden zu können, müssen die Datenmengen sortiert sein. Die Informatik kennt viele schnelle Sortierverfahren, und es ist gut untersucht, in welchen Fällen welche Verfahren angewendet werden sollen.

Noahs Sägerei

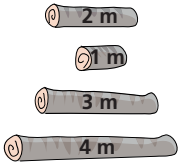
Biber Noah schneidet Holzstämme in verschiedenen Längen zu und verkauft sie dann. Sobald er einen Stamm zugeschnitten hat, legt er ihn auf dem 18 Meter langen Weg ab. Dabei beachtet Noah folgende Regel: Er legt den Stamm in die erste Lücke von links, in die der Stamm passt.

Noah verkauft einige Stämme. Danach gibt es drei Lücken auf dem Weg:



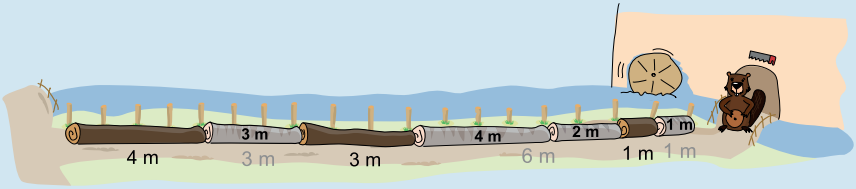
Nun will Noah vier Stämme zuschneiden, mit Längen von 1, 2, 3 und 4 Metern.

In welcher Reihenfolge muss Noah die Stämme zuschneiden, damit er alle vier in die Lücken legen kann?



1	2	3	4

So ist es richtig:



Schneidet Noah die Stämme in der Reihenfolge (3 m, 4 m, 2 m, 1 m) zu, passen sie alle auf den Weg: Für den 3 m-Stamm ist die 3 m-Lücke ganz links die erste freie Lücke von links, in die der Stamm passt; dort legt Noah den Stamm ab. Der 4 m-Stamm kommt dann in die 6 m-Lücke links. Dann ist die verbleibende 2 m-Lücke die erste freie Lücke von links; da rein passt der nächste Stamm, und den letzten Stamm legt Noah in die 1 m-Lücke ab.

Weitere richtige Reihenfolgen sind (3 m, 2 m, 4 m, 1 m) und (4 m, 3 m, 2 m, 1 m).

Alle anderen Reihenfolgen führen dazu, dass Noah nicht in der Lage ist, alle Baumstämme abzulegen: Der 1 m lange Stamm muss immer als letztes an der Reihe sein, weil nur dieser Stamm den letzten freien Platz ausfüllen kann. Der 2 m-Stamm darf nicht vor dem 3 m-Stamm kommen, weil er sonst in die 3 m-Lücke gelegt würde und dadurch eine zweite 1 m-Lücke entsteht. Außer den drei genannten Reihenfolgen gibt es keine Reihenfolgen, die diese Bedingungen erfüllen.

Das ist Informatik!
Diese Biberaufgabe ist ein Spezialfall des *Behälterproblems* (Englisch auch *bin packing problem*). Beim Behälterproblem müssen Objekte unterschiedlicher Größen in einer bestimmten Anzahl von Behältern untergebracht werden, die selbst auch wieder unterschiedliche Größen haben können. Die Objekte sind hier die Baumstämme, die Behälter die Lücken auf dem Weg.

Das Behälterproblem kommt in ganz unterschiedlichen Lebensbereichen vor. Einige Beispiele:
(a) In einem Möbellager müssen kleine und große Möbel platzsparend untergebracht werden.
(b) Eine Spedition kann Geld sparen, wenn sie zum Transport von Gütern durch geschicktes Packen weniger Lastwagen braucht.
(c) Das Betriebssystem eines Computers muss Dateien unterschiedlicher Größe auf der Festplatte speichern. Wenn Dateien gelöscht werden, entstehen Lücken auf der Festplatte. Diese Lücken müssen gefüllt werden, damit kein Speicherplatz verschwendet wird, ganz ähnlich wie auf der Straße bei dieser Biberaufgabe.

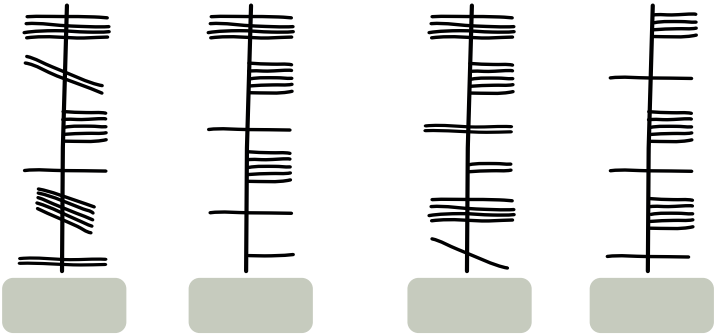
In der Informatik gilt das Behälterproblem als eines der schwersten Probleme; garantiert optimale Lösungen können auch von Computerprogrammen nur für kleine Fälle mit wenigen Objekten und wenigen Behältern gelöst werden. Es gibt aber verschiedene Verfahren und Strategien, mit denen gute Lösungen des Behälterproblems bestimmt werden können. In dieser Biberaufgabe ist die Strategie durch Noahs Regel vorgegeben. Sie legt jeden Baumstamm immer in die erste Lücke von links, in die er passt. Diese Strategie nennt man *First Fit.* Man sieht am Beispiel dieser Aufgabe, dass diese Strategie zu schlechten Ergebnissen führen kann: Nur wenn die Stämme in einer bestimmten Reihenfolge abgelegt werden, können alle Lücken gefüllt werden.

Ogham

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

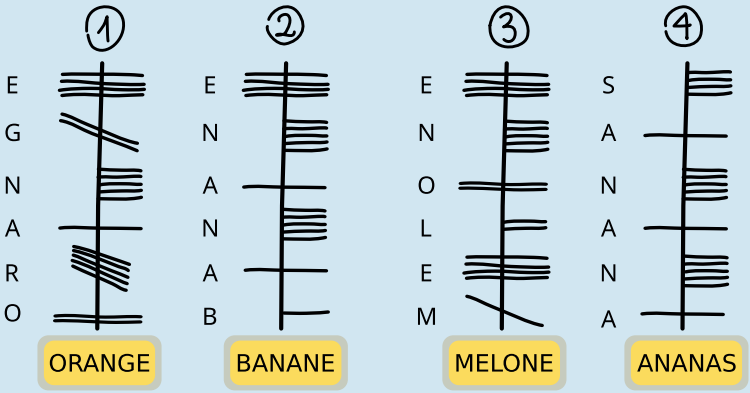
Sue benutzt Ogham als Code. Sie kodiert vier Wörter – ihre liebsten Obstsorten: ANANAS, BANANE, MELONE und ORANGE.

Welches Wort passt zu welchem Ogham-Code?



ANANAS BANANE MELONE ORANGE

So ist es richtig:



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

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

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

Das ist Informatik!
In dieser Biberaufgabe muss ein unbekannter Text entschlüsselt bzw. dechiffriert werden. Das ist hier nicht sehr schwierig, weil der entschlüsselte *Klartext* bekannt ist. Außerdem ist der unbekannte Text auf gleiche Weise in Buchstaben und Wörter eingeteilt wie der bekannte Text.

Beim Dechiffrieren eines geheimen Textes bzw. eines Textes in unbekannter Schrift, dessen Klartext nicht bekannt ist, hilft es oft, sich Gedanken über die Häufigkeit von Buchstaben und Wörtern zu machen und auf dieser Grundlage zu versuchen, sie im Text zu finden. Auf diese Weise sind einige antike Alphabete und Schriften entschlüsselt worden.

Schwierig wird es aber, wenn die Schriftzeichen im unbekannten Text nicht so einfach den Buchstaben und Wörtern der bekannten Sprache zuzuordnen sind wie im Fall von Ogham. Dann hilft oft nur der Abgleich mit bekannten Texten oder Schriften, wie in dieser Biberaufgabe. Zum Beispiel wurden die ägyptischen Hieroglyphen jahrhundertelang nicht entschlüsselt, bis durch einen Zufall ein Stein mit Hieroglyphen und zwei bekannten Schriften gefunden wurde, der Stein von Rosetta. Auf dem Stein fand sich dreimal der gleiche Text. Der war zwar in verschiedenen Sprachen geschrieben, enthielt aber immer dieselben Namen. So konnten wesentliche Elemente der Hieroglyphen entschlüsselt werden.

Das gilt aber nicht für alle Schriften: Noch immer sind die etwa 650 Schriftzeichen der Maya-Kultur nicht vollständig entschlüsselt, genau so wenig wie die Schriften Linearschrift A und Linearschrift B aus der Mittelmeerregion.

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

Postfix-Notation

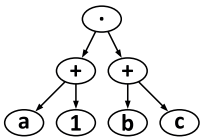
Ein mathematischer Ausdruck besteht aus

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

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

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

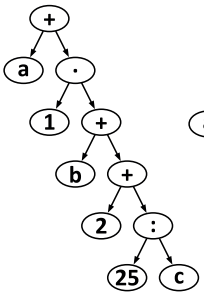
Die Tabelle zeigt für zwei Ausdrücke ihre Strukturbäume und Postfix-Notationen:

Mathematischer Ausdruck	a + b	(a + 1) · (b + c)
Strukturbaum		
Postfix-Notation	a b +	a 1 + b c + ·

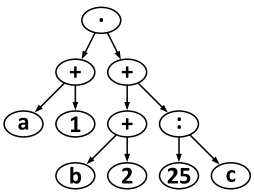
Hier ist die Postfix-Notation eines anderen Ausdrucks: a 1 + b 2 + · 25 c : +

Welchen Strukturbaum hat dieser Ausdruck?

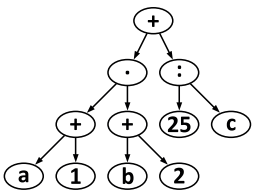
A



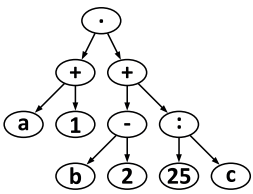
B



C



D



Antwort C ist richtig:

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

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

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

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

Das ist Informatik!
Die *Postfix-Notation,* auch *umgekehrte polnische Notation* (engl. *reverse Polish notation RPN*) genannt, wird oftmals verwendet, um mathematische oder andere Ausdrücke (z. B. in Programmiersprachen) missverständnisfrei und kompakt zu formulieren. Würde man den durch den Strukturbaum aus Antwort C gegebenen Ausdruck in normaler Notation schreiben (also mit Operatoren zwischen den Operanden, deshalb auch Infix-Notation genannt), müsste man Klammern setzen (a + 1) · (b + 2) + 25 : c, die man für die Postfix-Notation nicht braucht.

Die Postfix-Notation wurde von Jan Łukasiewicz (1878-1956) zunächst als Präfix-Notation eingeführt, mit dem Operator vor den Operanden. So schreibt man u.a. die Anwendung von Funktionen auf: in der Mathematik *f(x, y),* beim Programmieren `funktionsname(argument1, argument2, argument3)`. Im Computer wird sie unter anderem beim Parsen von Ausdrücken einer Programmiersprache verwendet.

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

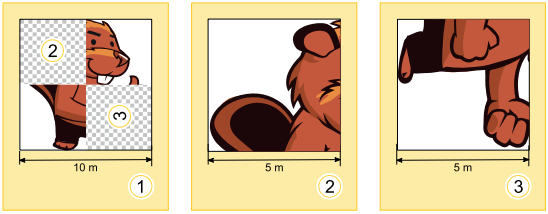
Rekursive Malerei

Tina und Tom helfen bei der Vorbereitung einer Sonderausstellung im Informatik-Museum. Auf den Boden eines Ausstellungsraums sollen sie ein 16 x 16 Meter großes Bild malen.

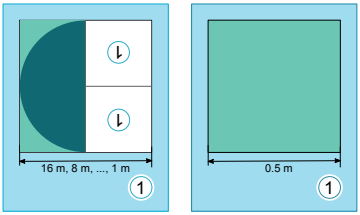
Vom Künstler bekommen sie einen Satz Malanweisungskarten in dessen berühmter Malkartensprache, mit Hinweisen zu den Bildelementen, Maßen und Drehungen.

Auf manchen Malanweisungskarten sind nummerierte Felder, die auf andere Karten verweisen.

Hier ein Beispiel aus einem früheren Malkartenprojekt. Wenn man diese drei Karten richtig ausführt, entsteht ein Bild des Bibers:



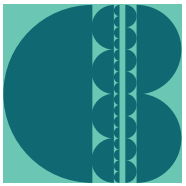
Für die Sonderausstellung bekommen Tina und Tom nun diese zwei Karten:



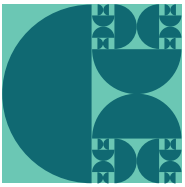
Tom runzelt die Stirn. „Wie soll das gehen? Die linke Karte verweist auf sich selbst, und ausserdem haben beide Karten dieselbe Nummer!" Tina lacht: „Wir kriegen das hin! Zuerst verwenden wir nur die linke Karte. Die rechte Karte wird uns später anweisen, wann wir mit dem Malen aufhören sollen."

Wie wird der Boden des Ausstellungsraums aussehen?

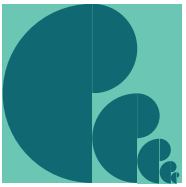
A



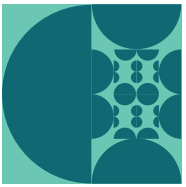
B



C



D



Antwort A ist richtig:

Die linke Malanweisungskarte besagt, dass die linke Hälfte des Bodens mit einer Halbkreisfläche gefüllt werden soll, deren runde Seite bei normaler Orientierung nach links zeigt. Für die rechte Hälfte soll dieselbe Malanweisungskarte zwei Mal verwendet werden. Die Orientierungen der Bildelemente muss den Orientierungen der Einsen entsprechen.

In beiden Hinweisen auf Karte 1 ist die 1 um 180° gedreht, nach unten. Deshalb müssen die dort eingesetzten Bildelemente ebenfalls so gedreht werden, so dass die Rundung der dann gemalten Halbkreisflächen genau in die Gegenrichtung zeigt. Bei der ersten Verwendung der linken Karte 1 (bei der Breite 16 m) zeigt die Rundung der Halbkreisfläche nach links, bei 8 m nach rechts, bei 4 m wieder nach links usw. Bei 0,5 m wird die zweite Karte 1 verwendet: Tom und Tina malen die verbleibende freie Fläche noch aus und können danach aufhören.

Auf diese Weise entsteht genau das Bild von Antwort A.

Das ist Informatik!
Die erste Malanweisungskarte 1 in dieser Biberaufgabe verweist auf sich selbst. Sie ruft Tom und Tina sozusagen dazu auf, sie selbst erneut anzuwenden, mit einer geringeren Breite. In der Informatik werden Anweisungen, die sich selbst aufrufen, als *rekursiv* bezeichnet. Der Begriff kommt von lateinisch „recurrere“ (deutsch: zurücklaufen). Rekursion ist ein mächtiges Konzept. Für manche komplexen Aufgaben kann man kurz und überschaubar eine rekursive Anweisung zu ihrer Lösung formulieren.

Eine rekursive Anweisung muss eine Bedingung enthalten, die festlegt, wann die Rekursion abgebrochen werden soll. Sonst arbeitet die Rekursion solange weiter, bis irgendeine Ressource erschöpft ist. Etwa der Computerspeicher oder die Geduld des Benutzers. In dieser Biberaufgabe hat die zweite Karte 1 diese Funktion: Unter der Bedingung, dass noch eine Fläche der Breite 0,5 m bemalt werden soll, wird sie „aufgerufen“. Da sie keinen Verweis auf eine andere Karte enthält, bricht sie die Rekursion ab.

Riccas

Evelyn hat fünf Bilder von Riccas. Sie beschreibt in Sätzen, wie Riccas aussehen.

Ihre Freundin Lydia zeigt ihr ein sechstes Bild von einem Ricca:

Nun stellt Evelyn fest: Einer ihrer Sätze über Riccas ist sicher falsch.

Welcher dieser Sätze über Riccas ist nun sicher falsch?

A) Alle Riccas haben Zähne.

B) Einige Riccas haben Flügel.

C) Riccas haben entweder Hörner oder drei Augen.

D) Wenn Riccas genau zwei Arme haben, dann haben sie auch genau zwei Beine.

Antwort D ist richtig:

Antwort A enthält eine Aussage, die für alle Riccas gelten muss. Wenn auch nur ein Ricca keine Zähne hätte, wäre sie falsch. Jedoch haben alle sechs Riccas, die Evelyn nun kennt, Zähne. Also kann Evelyns Satz nicht sicher falsch sein.

Antwort B enthält eine Aussage, die nur für einige Riccas gelten soll. Da eines der sechs Riccas, die Evelyn nun kennt, Flügel hat, ist dieser Satz für die sechs Riccas richtig. Aber selbst wenn keines der sechs Riccas Flügel hätte, könnten andere, weitere Riccas Flügel haben, und der Satz könnte noch richtig sein. Dieser Satz kann nur dann sicher falsch sein, wenn Evelyn alle Riccas kennen würde und keines Flügel hätte.

Antwort C verknüpft zwei Aussagen mit „entweder – oder". Diese verknüpfte Aussage ist genau dann wahr, wenn genau eine der beiden Aussagen wahr ist. Das ist für alle sechs Bilder der Fall: vier Riccas haben Hörner, aber keine drei Augen, die übrigen beiden Riccas haben keine Hörner, aber dafür drei Augen. Damit der Satz falsch wäre, müsste es mindestens ein Ricca mit drei Augen und Hörnern oder ein Ricca ohne Hörner und mit einer anderen Zahl als drei Augen geben. Unter den sechs Riccas, die Evelyn nun kennt, ist kein solches Ricca. Also ist der Satz für die sechs Riccas richtig, und nicht sicher falsch.

Bleibt der Satz von Antwort D. Er ist in Form einer „Wenn – Dann"-Aussage formuliert. Nur wenn die Wenn-Bedingung stimmt, muss auch die Dann-Aussage wahr sein. Die Bedingung ist wahr für alle sechs Riccas, die Evelyn nun kennt: sie alle haben genau zwei Arme. Alle Riccas auf Evelyns ersten fünf Bildern haben ebenfalls genau zwei Beine; für sie ist Evelyns Satz also richtig. Jedoch hat das Ricca auf Lydias Bild mehr als zwei Beine, nämlich fünf. Deshalb ist dieser Satz nun sicher falsch.

Das ist Informatik!
Die Anzahl der Arme, Beine und Augen, und ob Riccas Zähne, Hörner oder Flügel haben, sind *Eigenschaften* von Riccas. Wenn man Riccas beschreibt, formuliert man *Aussagen* über diese Eigenschaften. Das führt zu einem *Modell* davon, was Riccas sind.

Computer haben ebenfalls viele Modelle. Einige sind explizit formuliert wie beispielsweise ein Modell von Schülerinnen und Schülern, das aus Name, Geburtsdatum und Wohnadresse in einer Datenbank besteht. Andere Modelle bilden Computer aus Daten, wenn man ihnen beispielsweise Bilder zum Vergleich beim Training eines neuronalen Netzes gibt.

Evelyns Sätze – also ihr Modell der Riccas in dieser Biberaufgabe – sind als *logische Ausdrücke* formuliert. Einige haben *Quantoren* („(für) alle" oder „es gibt"/„einige"), andere nutzen *logische Operatoren* („entweder – oder" bzw. „wenn – dann"). Diese logischen Ausdrücke sind *formalisiert:* Das heißt, dass es eine Festlegung gibt, wie man sie verwendet und was sie bedeuten.

Anhand dieser Festlegungen
- können (einfache) Ausdrücke mit Hilfe von Quantoren und Operatoren zu komplexeren Ausdrücken verknüpft werden.
- kann die Bedeutung der komplexeren Ausdrücke aus den einfacheren Ausdrücken berechnet werden.

Logische Ausdrücke sind eine in der Informatik verbreitete Methode, Modelle zu beschreiben.

Schatzkisten

Auf einer Insel gibt es drei Schatzkisten: Eine Kiste ist am Fuß des Vulkans, die zweite ist unter einer Palme, und die dritte ist am Strand.
Alle Kisten sind leer.

An einem Tag kreuzt der Pirat Biberbart auf, füllt eine der Kisten mit Gold und verschließt sie. Am gleichen Tag sind drei Touristinnen auf der Insel: Anita, Britta und Carla. Jede macht ein Foto: eine, bevor Biberbart Gold in eine Kiste gefüllt hat, die anderen beiden danach.

Anitas Foto	Brittas Foto	Carlas Foto
… zeigt die Kiste am Strand.	… zeigt die zwei Kisten unter der Palme und am Strand.	… zeigt die zwei Kisten unter der Palme und am Fuß des Vulkans.
		

Auf den Fotos sind alle Kisten leer. Biberbart hatte also Glück, dass keine Touristin sein Gold gefunden hat.

In welcher Schatzkiste ist das Gold?

So ist es richtig:

Das Gold ist in der Schatzkiste am Fuß des Vulkans.

Wir prüfen für jede Kiste, ob darin das Gold sein kann. Dazu untersuchen wir jeweils, ob in diesem Fall die Fotos mit der Geschichte übereinstimmen.

Die Kiste unter der Palme: Brittas und Carlas Fotos zeigen die leere Schatzkiste unter der Palme. Wäre dies die Kiste mit dem Gold, müssten beide Fotos gemacht worden sein, bevor die Kiste gefüllt wurde. Wir wissen aber, dass nur eine Touristin ihr Foto gemacht hat, bevor Biberbart Gold in eine Kiste gefüllt hat. Die Annahme, dass das Gold in der Kiste unter der Palme ist, ergibt also einen Widerspruch. Daraus schliessen wir, dass in der Kiste unter der Palme kein Gold ist.

Die Kiste am Strand: Auf Anitas und Brittas Fotos ist die Schatzkiste am Strand leer. Wäre dies die Kiste mit dem Gold, müssten beide Fotos gemacht worden sein, bevor die Kiste gefüllt wurde. Wir wissen aber, dass nur eine Touristin ihr Foto gemacht hat, bevor Biberbart Gold in eine Kiste gefüllt hat. Die Annahme, dass das Gold in der Kiste am Strand ist, ergibt also einen Widerspruch. Daraus schließen wir, dass in der Kiste am Strand kein Gold ist.

Die Kiste am Fuß des Vulkans … ist nur auf Carlas Foto und ist dort leer. Wäre dies die Kiste mit dem Gold, könnte Carla die Touristin sein, die ihr Foto gemacht hat, bevor Biberbart Gold in eine Kiste gefüllt hat. Auf Anitas und Brittas Fotos ist die Kiste am Fuß des Vulkans nicht zu sehen. Anita und Britta können also die Touristinnen sein, die ihre Fotos danach gemacht haben. Die Annahme, dass das Gold in der Kiste am Fuß des Vulkans ist, ergibt keinen Widerspruch.

Da sich das Gold in einer der drei Kisten befinden muss, können wir insgesamt schlussfolgern, dass das Gold tatsächlich in der Kiste am Fuß des Vulkans ist.

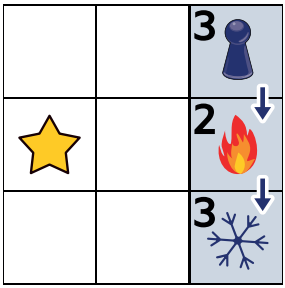
Das ist Informatik!
Beim Beantworten dieser Biberaufgabe hat *logisches Schlussfolgern* geholfen. Wir haben die drei Fotos und unser Wissen über die Situation auf der Insel verwendet, um Gründe zu finden, wieso bestimmte Annahmen zutreffen könnten oder eben nicht. Widersprüche zu konstruieren spielt beim logischen Schlussfolgern eine besonders wichtige Rolle. Wenn aus einer Annahme rein logisch eine Aussage folgt, aber Annahme und Aussage nicht gleichzeitig wahr sein können, dann können wir mit Sicherheit sagen, dass die Annahme falsch ist.

Logik spielt eine überaus wichtige Rolle in vielen Bereichen der Informatik: Die Schaltungen in der Computer-Hardware, ob in den Recheneinheiten oder in Speichermedien, sind Realisierungen von logischen Operationen. Mit logischen Verknüpfungen können komplexe Bedingungen in Programmen oder komplexe Abfragen an Datenbanken formuliert werden. Das Verhalten von Programmen kann mit Hilfe logischer Kalküle beschrieben und verifiziert werden. Und logische *Programmiersprachen* arbeiten direkt mit logischen Aussagen und logischem Schlussfolgern, um Berechnungen durchzuführen.

Schatzsuche

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

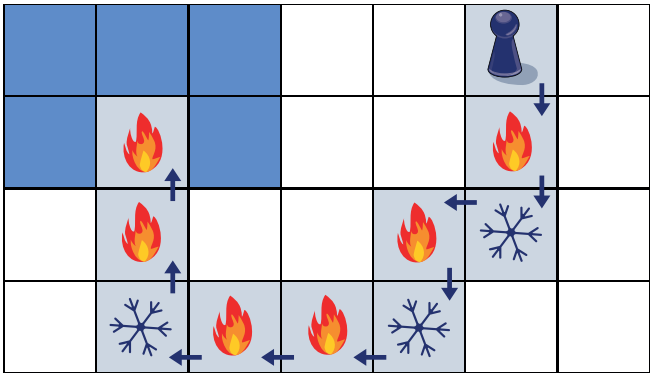
Daniel wählt ein Startfeld aus. Von dort geht er schrittweise mit seiner Spielfigur um je ein Feld weiter: nach links, rechts, oben oder unten.



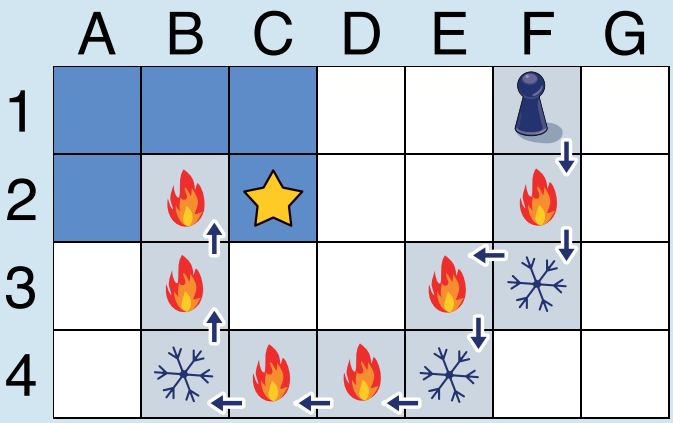
Beim ersten Versuch nehmen sie ein kleines Spielbrett.
Nina versteckt den Schatz auf dem Feld mit dem Stern.
Daniel startet rechts oben und macht zwei Schritte entlang der Pfeile. Nach jedem Schritt sagt Nina, ob Daniel nun näher (❄) am Schatz oder weiter weg (🔥) vom Schatz ist als vor dem Schritt – siehe Bild. Die Zahlen zeigen Daniels Entfernungen vom Schatz: jeweils die kleinste Anzahl Schritte, mit denen Daniel aktuell zum Schatz gehen könnte.

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

Wo ist der Schatz versteckt?



So ist es richtig:



Wir verfolgen Davids Weg und Ninas Rückmeldungen. David startet in Zeile 1 des Spielbretts. Nach dem ersten Schritt ist er in Zeile 2 und näher am Schatz als in Zeile 1. Nach dem nächsten Schritt ist er in Zeile 3 und wieder weiter weg vom Schatz. Da er in der gleichen Spalte geblieben ist, muss der Schatz auf einem Feld in Zeile 2 sein. Denn egal, in welcher Spalte der Schatz versteckt ist: Den kürzesten Weg zum Schatz von einer anderen Spalte aus hat man, wenn man in der gleichen Zeile ist.

Aber in welcher Spalte ist der Schatz versteckt? Auf seinem weiteren Weg kommt Daniel in Zeile 4 mit einigen Schritten nach links dem Schatz zunächst näher; insbesondere ist er in Spalte C näher am Schatz als in Spalte D. Aber nach dem letzten Schritt in der Zeile ist Daniel in Spalte B weiter weg vom Schatz als in Spalte C. Der Schatz muss also auf einem Feld in Spalte C sein. Denn was wir oben für Spalten gesagt haben, gilt auch für Zeilen: Den kürzesten Weg zum Schatz von einer anderen Zeile aus hat man, wenn man in der gleichen Spalte ist.

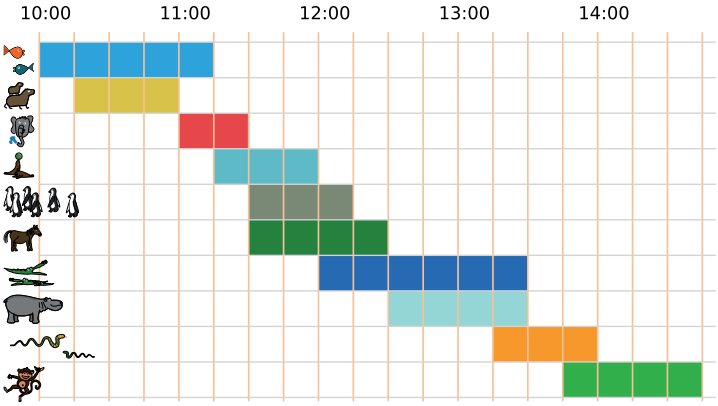
Das ist Informatik!
Daniel geht (mit seiner Spielfigur) über das Spielbrett. Von jedem Feld, auf dem er gerade steht, misst Nina die Entfernung zum Feld mit dem Schatz und nutzt dies für ihre Rückmeldung. Üblicherweise wird die Entfernung zwischen zwei Punkten als Länge der geradlinigen Verbindung zwischen den Punkten gemessen *(euklidische Distanz).* Doch die zwei Felder sind genau genommen keine Punkte. Deshalb bemisst Nina die Entfernung zwischen zwei Feldern in der Anzahl von Schritten, die Daniel für einen kürzesten Weg vom einen Feld zum anderen gehen müsste. Dieses *Maß* kann man generell bei Rastern anwenden und ist in der Informatik als *Manhattan-Distanz* bekannt – abgeleitet vom gerasterten Straßenplan des New Yorker Stadtteils Manhattan.

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

Spaß im Zoo

Heute ist Ali im Zoo. Er will möglichst viele verschiedene Vorführungen besuchen.

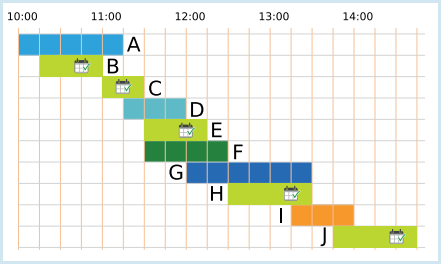
Hier ist ein Plan mit allen Vorführungen. Zum Beispiel siehst du ganz unten:
Die Vorführung der Affen beginnt um 13:45 Uhr und endet um 14:45 Uhr.

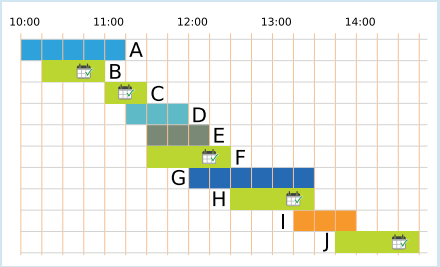


Ali besucht eine Vorführung immer ganz, von Anfang bis Ende. Kannst du Ali helfen?

Wähle so viele Vorführungen wie möglich aus, die Ali nacheinander besuchen kann.

So ist es richtig:
Ali kann höchstens 5 Vorführungen nacheinander besuchen.
Das sind die beiden richtigen Antworten:





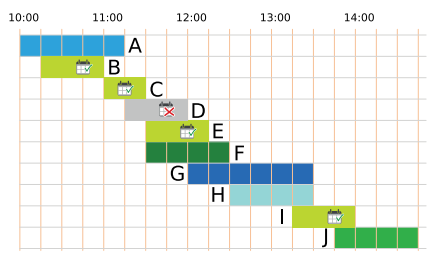
Es gibt unterschiedliche Wege, die richtigen Antworten zu finden.

Ein Besuchsplan für Ali ist eine Auswahl von Vorführungen, die er nacheinander besuchen kann. Ein Weg zu den richtigen Antworten ist es, alle Besuchspläne aufzulisten. In dieser Liste sind die Pläne mit den meisten Vorführungen die richtigen Antworten. Alle Besuchspläne zu finden, ist leider sehr zeitaufwändig.

Zeiteinheiten	Vorführungen
2	C
3	B, D, E, I
4	F, H, J
5	A
6	G

Aber könnte es nicht auch einen Besuchsplan mit 6 Vorführungen geben? Wir versuchen einmal, einen zu erstellen. Vorher schauen wir uns die Dauer der Vorführungen genauer an: Der gesamte Tag ist auf dem Plan in 19 Zeiteinheiten zu je einer Viertelstunde unterteilt. Die Vorführungen dauern 2, 3, 4, 5 oder 6 Zeiteinheiten.

Um möglichst viele Vorführungen in einen Besuchsplan zu packen, wählen wir so *kurze* Vorführungen wie möglich. Die 6 kürzesten Vorführungen dauern zusammen 18 Zeiteinheiten (2 + 3 + 3 + 3 + 3 + 4). Zu diesen kurzen Vorführungen gehören auch die Vorführungen C, D und E. Da die Vorführungen C und E aber genau hintereinander liegen, kann Ali Vorführung D dazwischen nicht besuchen.



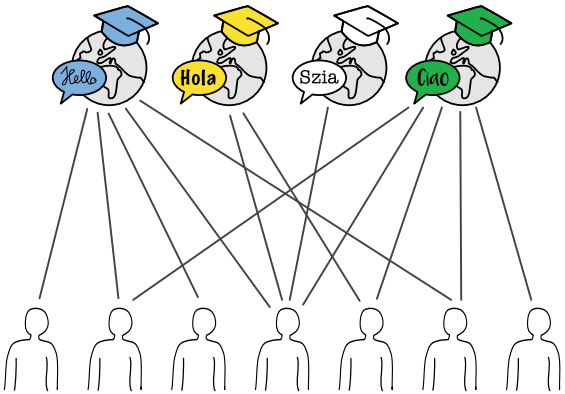
Wir müssen also die Vorführung D durch eine andere möglichst kurze ersetzen. Es sind nur noch Vorführungen mit mindestens 4 Zeiteinheiten übrig. Ohne die Vorführung D benötigen wir deshalb insgesamt mindestens 19 Zeiteinheiten für 6 Vorführungen: 2 + 3 + 3 + 3 + 4 + 4. Aber: Welche zwei Vorführungen mit 4 Zeiteinheiten wir auch wählen, eine davon überschneidet sich immer mit einer Vorführung mit 3 Zeiteinheiten. Wir müssten auch diese durch eine Vorführung mit mindestens 4 Zeiteinheiten ersetzen und würden dann mindestens 20 Zeiteinheiten für 6 Vorführungen benötigen. Es stehen aber nur 19 Zeiteinheiten zur Verfügung! Wir schlussfolgern, dass es keinen Besuchsplan geben kann, der mehr als 5 Vorführungen enthält.

Das ist Informatik!
Diese Biberaufgabe enthält einen Zeitplan der Vorführungen im Zoo. Solche Zeitpläne herzustellen, ist nicht einfach und wird in der Informatik als *Scheduling-Problem* bezeichnet. Natürlich möchte der Zoo seinen Besuchern ermöglichen, möglichst viele Vorführungen zu sehen, aber es müssen auch andere Bedingungen beachtet werden. Beispielsweise können Vorführungen nur angeboten werden, wenn die Tierpfleger Zeit haben, die verfügbaren Arenas frei sind und die Vorführungen sich mit den Lebensrhythmen der Tiere vereinbaren lassen.

Es gibt viele ähnliche Probleme im Leben, auf die sich dieselben Überlegungen anwenden lassen. Ein Beispiel ist die Erstellung eines Stundenplanes in der Schule, oder die Zuteilung von Kinofilmen zu Kinosälen. Die Erstellung dieser Zeitpläne ist so aufwändig, dass man dies schon für relativ kleine Beispiele (die Stundenpläne deiner Schule) oft nicht mehr von Hand erarbeiten kann. Auch die *Prozessoren* deines Computers müssen viele Aufgaben übernehmen und diese nacheinander abarbeiten. Der Zeitplan, wann welcher Prozessor was tut, wird vom *Betriebssystem* blitzschnell und ohne dass man es merkt erstellt. Scheduling ist eines der großen Themen der Informatik, mit welchen sich die Forschung auch heute noch beschäftigt.

Sprachkurse

Eine Sprachschule plant vier Sommerkurse. Die Linien im Bild zeigen, welche Lehrperson der Schule (unten) für welchen Kurs (oben) geeignet ist.

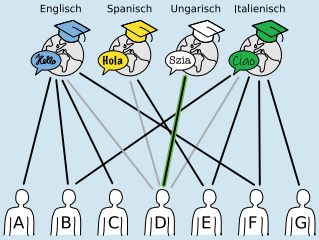


Eine Lehrperson kann nur einen Kurs halten. Trotzdem gibt es mehrere Möglichkeiten, jedem Kurs eine geeignete Lehrperson zuzuordnen.

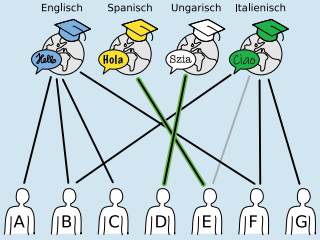
Ordne jedem Kurs eine geeignete Lehrperson zu.
Markiere dazu die Linie zwischen Person und Kurs.

Um die Antwort leichter erklären zu können, markieren wir die Lehrpersonen mit den Buchstaben A bis G. Die Sprachen, die in den Kursen unterrichtet werden, sind (von links nach rechts) Englisch, Spanisch, Ungarisch und Italienisch.

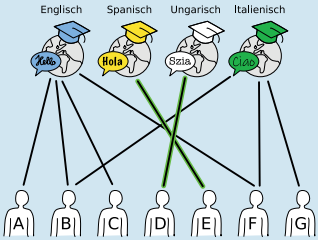
D ist die einzige Lehrperson, die für den Ungarischkurs geeignet ist. Sie muss diesem Kurs zugeordnet werden und kann keine weiteren Kurse übernehmen.



E ist jetzt die einzige Lehrperson, die für den Spanischkurs geeignet ist. Sie muss diesem Kurs zugeordnet werden und kann keine weiteren Kurse übernehmen.



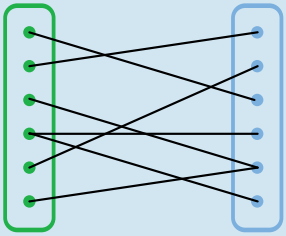
Bei den beiden verbleibenden Kursen (Englisch und Italienisch) kann man recht frei wählen. B und F dürfen aber nur einem Kurs zugeordnet werden, auch wenn sie für beide geeignet sind



Dadurch gibt es insgesamt 10 Möglichkeiten, jedem Kurs eine geeignete Lehrperson zuzuordnen:

Englisch	Italienisch	Ungarisch	Spanisch
A	B	D	E
A	F	D	E
A	G	D	E
B	F	D	E
B	G	D	E
C	B	D	E
C	F	D	E
C	G	D	E
F	B	D	E
F	G	D	E

Das ist Informatik!
Ein Graph besteht aus Knoten (Punkten), die durch Kanten (Linien) verbunden sind. Eine spezielle Klasse von Graphen sind *bipartite Graphen:* Die Knoten lassen sich in zwei getrennte Teilmengen teilen, sodass es nur Kanten zwischen Knoten verschiedener Teilmengen gibt.



Die Situation in dieser Biberaufgabe kann durch einen bipartiten Graphen dargestellt werden: Eine Teilmenge besteht aus den Kursen und die andere aus den Lehrpersonen. Bipartite Graphen eigenen sich sehr gut, *Zuordnungsprobleme* zu modellieren und zu lösen. Zuordnungsprobleme begegnen uns häufig im Alltag, z.B. bei Stundenplänen oder bei der Verteilung von Arbeit an Angestellte oder Maschinen. Bei kleineren Problemen ist es einfach möglich, eine optimale Zuordnung zu finden; bei größeren wird es jedoch relativ schnell sehr komplex. Aus diesem Grund wurden in der Informatik verschiedene Algorithmen entwickelt, um möglichst schnell möglichst viele passende Paare zu finden.

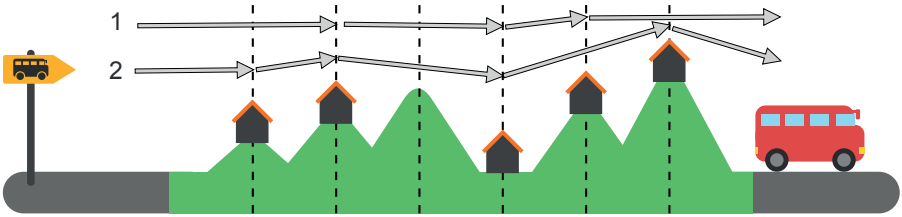
Zum Beispiel wird auch das sogenannte Heiratsproblem mithilfe eines bipartiten Graphen dargestellt. Dabei steht eine Menge von heiratswilligen Männern einer Menge von heiratswilligen Frauen gegenüber. Ziel des Verfahrens ist, unter Berücksichtigung der jeweiligen Wünsche, alle Männer beziehungsweise alle Frauen zu verheiraten. Der englische Mathematiker Philip Hall formulierte im Heiratssatz 1935 die Bedingungen, unter denen so eine Zuordnung möglich ist.

In unserer Variante geht es nicht um diese vollständige Zuordnung, sondern darum, möglichst jeden Knoten der einen Teilmenge (die Kurse) einem Knoten der anderen Teilmenge zuzuordnen.

Wanderungen

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

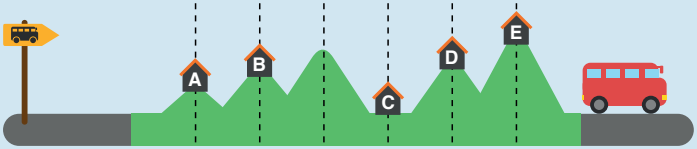
Mia hat die Region mit gestrichelten Linien in Abschnitte eingeteilt. Sie kann immer nur einen oder zwei Abschnitte an einem Tag wandern. Zwei verschiedene Wanderungen, die sie machen kann, hat sie bereits in die Karte eingetragen. Wanderung 1 hat 3 Übernachtungsorte; Wanderung 2 hat 4 Übernachtungsorte.



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

6 ist die richtige Antwort:

Wir benennen die Orte mit Buchstaben:



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

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

C → D → E → ; C → E → ; C → D →

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

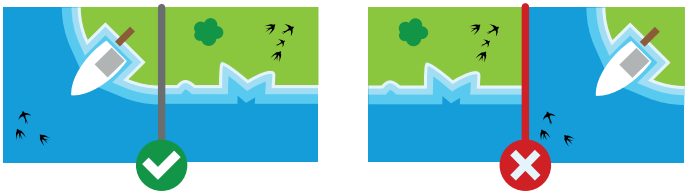
Das ist Informatik!
Manchmal kann die Zahl aller Möglichkeiten, eine gegebene Aufgabe zu erledigen, sehr groß sein. Zum Beispiel gibt es etwa 14 Millionen Möglichkeiten, 6 verschiedene Zahlen aus den Zahlen 1 bis 49 auszuwählen. Und es gibt etwa eine halbe Milliarde Möglichkeiten, die Zahlen von 1 bis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

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

Nach diesem Prinzip funktionieren zum Beispiel einige wichtige Sortieralgorithmen. Auch die *dynamische Programmierung,* eine Methode zur algorithmischen Lösung von von Optimierungsproblemen (beschrieben 1957 von Richard Bellman), basiert auf diesem Prinzip: Wenn man erkennt, dass die optimalen Lösungen eines Problems sich aus den optimalen Lösungen von Teilproblemen zusammensetzen, kann man dies nutzen, um sozusagen „klein anzufangen“: Zunächst werden die Lösungen für die kleinsten Teilprobleme direkt berechnet und anschließend zu Lösungen für die nächstgrößeren Teilprobleme zusammengesetzt. Dies wird wiederholt, bis die optimale Lösung des vollständigen Problems gefunden ist. Da gefundene Teil-Lösungen häufig zu Lösungen vieler größerer Teile beitragen, werden sie gespeichert, um wiederholte gleiche Berechnungen einzusparen. Auch beim Zählen von Möglichkeiten kann dynamische Programmierung sehr hilfreich sein.

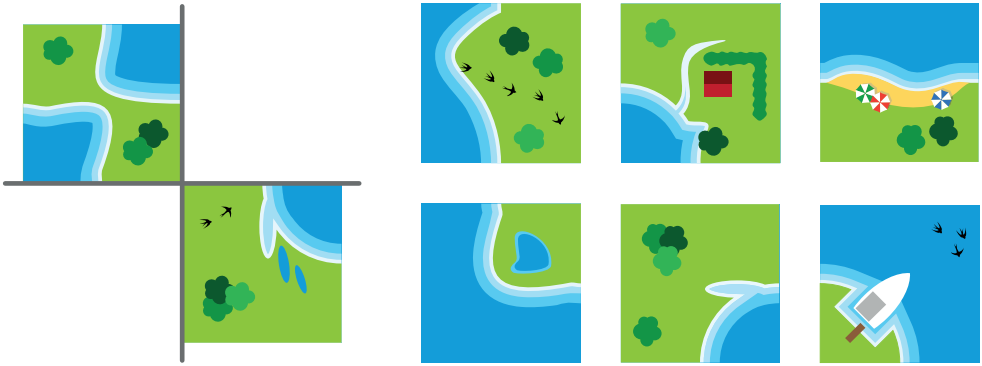
Wasser – Land

Edu hat ein neues Spiel. Das Spiel hat Kärtchen mit Wasser und Land. Edu legt die Kärtchen so aneinander, dass sie passen: Land an Land, Wasser an Wasser. Dann entstehen schöne Landschaften.



Edu legt zwei Kärtchen hin und lässt zwei Lücken.

Welche Kärtchen passen in die Lücken?



So ist es richtig:

Das Kärtchen mit dem Boot und das mit den fünf Vögeln passen in die Lücken: Überall liegt Wasser an Wasser und Land an Land. Von den sechs möglichen Kärtchen passen nur diese beiden Kärtchen in die Lücken.

Nur wenn man die Kärtchen drehen dürfte, würden noch weitere Kärtchen in die Lücken passen.

Das ist Informatik!
Schauen wir uns Edus Kärtchen genauer an. Alle Kärtchen können in vier Bereiche geteilt werden. Dabei zeigen alle äußeren Ränder jedes Bereichs entweder Land oder Wasser.

Es gibt also nur zwei verschiedene Arten von Bereichen, Wasser (W) oder Land (L). Zwei Kärtchen passen aneinander, wenn benachbarte Bereiche die gleiche Art haben. Daher können wir für je drei Bereiche der beiden Lücken die erforderliche Art eintragen. Der vierte Bereich kann Wasser oder Land sein, deswegen tragen wir „*“ ein.

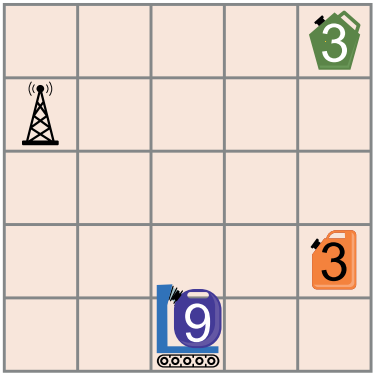
Auf diese Weise erstellen wir für jede Lücke ein Muster. Die Kärtchen, die die Lücken füllen sollen, müssen zu diesen Mustern passen: Wo ein Muster einen Buchstaben hat, muss das Kärtchen den gleichen Buchstaben haben; wo das Muster einen Stern hat, kann das Kärtchen irgendeinen Buchstaben haben. Zu jedem Muster passt nur ein Kärtchen auf der rechten Seite.

Wir haben eine Eigenschaft der Kärtchen entdeckt. Diese Entdeckung haben wir genutzt, um sie durch eine Anordnung der Zeichen L und W zu beschreiben. Durch diesen Schritt haben wir die in den Bildern enthaltenen Informationen erheblich reduziert. Wir konzentrieren uns auf die Informationen, die zur Lösung dieser Aufgabe erforderlich sind. Informatiker würden sich auf die Anordnung der Zeichen in den Bildern beziehen. Durch die Reduktion der Bilder auf die Bereichsarten L und W entsteht ein Modell für die fehlenden Kärtchen. *Modellierung* bedeutet *Abstraktion* (oder Vereinfachung), und Abstraktion reduziert die Information. Computer müssen mit Modellen von der Realität arbeiten. Bei der Erstellung solcher Modelle muss darauf geachtet werden, dass wichtige Eigenschaften der Realität nicht verloren gehen.

Zerobots Mission

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

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



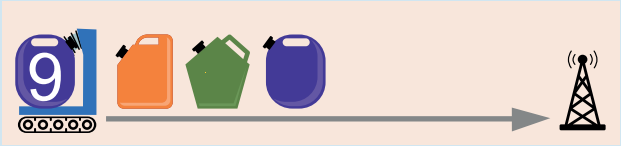
Zerobots aktuelle Position und der Füllstand seines Tanks werden im Bild so angezeigt: 

Alarm: Die Tanks sind fehlerhaft und könnten explodieren!

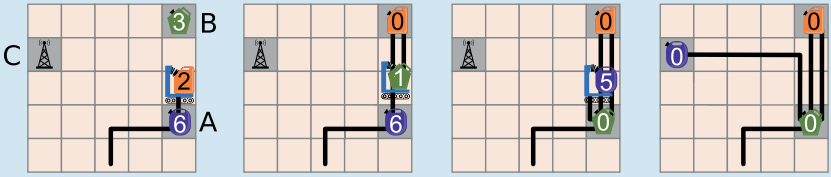
Das ist Zerobots Mission:  Er soll so zur Basisstation fahren, dass am Ende alle Tanks leer sind (Füllstand 0).

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

So ist es richtig:



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



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

Zerobot fährt 3 Felder bis A und tauscht  (Füllstand 6) gegen  (Füllstand 3) aus. Dann fährt er 3 Felder bis B und tauscht  (Füllstand 0) gegen  (Füllstand 3) aus. Damit fährt er wieder zu A und tauscht  (Füllstand 0) gegen  (Füllstand 6) aus.

Damit fährt er 6 Felder bis zur Basistation C.  hat dann den Füllstand 0. Mission erfüllt!

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

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

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

Zerteile den Code

In einem speziellen Code für Texte wird jeder Buchstabe durch ein Codewort aus den Ziffern **0** bis **9** kodiert. Dabei gilt diese Regel: Kein Codewort darf mit dem Codewort eines anderen Buchstabens beginnen.

Der Buchstabe **X** wird beispielsweise durch **12** kodiert.
Nun kann **Y** durch **2** kodiert werden, denn **12** beginnt nicht mit **2** (und **2** nicht mit **12**).
Jetzt kann **Z** durch **11** kodiert werden; denn weder **12** noch **2** beginnen mit **11** und **11** beginnt weder mit **12** noch mit **2**. **21** wäre jedoch nicht als Codewort für **Z** erlaubt, weil es mit **2**, also dem Codewort von **Y** beginnt.

Das Wort **MEMORY** wird durch die Ziffernfolge **12112233321** kodiert.

Teile die Ziffernfolge in die Codewörter der einzelnen Buchstaben!

So ist es richtig: 1 21 1 22 33 321

Man beginnt links am Anfang der Ziffernfolge. Falls **M** durch 12 kodiert würde, hätte **E** zwangsläufig das Codewort 1, denn dahinter käme wieder 12 für das zweite **M**. Das würde jedoch der Regel widersprechen: Das Codewort für **M** würde dann mit 1 beginnen, dem Codewort für **E**. Längere Anfangsstücke der Ziffernfolge (121, 1211, 12112 etc.) können auch nicht das Codewort für **M** sein, weil dieses Codewort zwei Mal vorkommen muss, diese Stücke aber jeweils nur einmal in der Ziffernfolge enthalten sind. Folglich ist das Codewort für **M** die Ziffer 1.

Nun muss das Codewort für **E** folgen – und dahinter wieder das für **M**, also die 1. Somit kann das Codewort für **E** nur eine der folgenden Ziffernfolgen sein: 2, 21 oder 211223332. Es kann nicht 2 sein; dann würde das Wort mit **MEMM** beginnen. Es kann nicht 211223332 sein; dann wäre das Wort insgesamt nur **MEM**. Folglich ist das Codewort für **E** die Ziffernfolge 21. Nun ist klar, dass 1 21 1 die Kodierung für **MEM** ist.

Der Rest der Ziffernfolge, also 2233321, kodiert die Buchstaben **ORY**. Die 2 alleine kann nicht das Codewort für **O** sein, sonst hätten wir **OO** zu Beginn. Das Codewort für **O** muss also mindestens die 22 beinhalten. Am Ende wiederum sind 1 und 21 schon als Codewörter für **M** bzw. **E** vergeben. Das Codewort für **Y** muss also mindestens die Folge 321 sein. Zwischen 22 und 321 steht 33. Das muss das Codewort für **R** sein: Das einzig andere noch möglich Codewort für **R** wäre 3. Dann müsste 3321 das Codewort für **Y** sein – und würde mit dem Codewort für **R** beginnen; das ist gegen die Regel. Die Aufteilung des hinteren Teils ist also 22 33 321.

Das ist Informatik!
Der Code, der in dieser Biberaufgabe beschrieben wird, ist ein Beispiel für einen *Präfixcode.* Ein Präfix ist eine Zeichenfolge zu Beginn einer anderen Zeichenfolge. Bei einem Präfixcode darf kein Codewort Präfix eines anderen Codeworts sein. Das heisst: kein Codewort darf mit einem anderen Codewort beginnen.

Bei Präfixcodes haben die Codewörter unterschiedliche Länge. Der Vorteil der Präfix-Regel ist, dass man keine Trennsymbole zwischen Codewörtern benötigt. Man kann immer erkennen, an welcher Stelle das nächste Codewort beginnt. Wenn man kurze Codewörter für häufig vorkommende Buchstaben wählt, kann man Texte sehr effizient kodieren und große Textmengen platzsparend speichern.
Die Huffman-Kodierung ist eine Methode, einen optimalen Präfixcode zu finden. Sie ist weit verbreitet und steckt z.B. hinter bekannten Datenformaten wie JPEG und MP3.

Ziffernschloss

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

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

1 >> 3

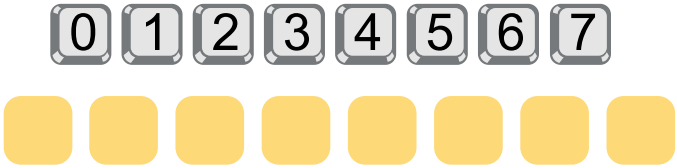
dass links von Ziffer 3 genau eine Ziffer steht (nämlich die 4), die größer ist als 3.
Den aktuellen Zifferncode hat er sich insgesamt so notiert:

0 >> 0 ; 3 >> 1 ; 0 >> 2 ; 1 >> 3 ; 0 >> 4

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

3 >> 0 ; 2 >> 1 ; 4 >> 2 ; 4 >> 3 ; 1 >> 4 ; 1 >> 5 ; 1 >> 6 ; 0 >> 7

Wie lautet der neue Code?



Ziehe die Ziffern an die richtigen Stellen.

So ist es richtig:

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

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

Das ist Informatik!
Bob bescheibt in seiner Notation, wie sich der Code zu einer sortierten Folge der verwendeten Ziffern bzw. Zahlen verhält.

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

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

Die Folge dieser Inversionszahlen heißt *Inversionsssequenz.* (Die Summe der Inversionszahlen beschreibt übrigens den Grad der Unsortiertheit einer Permutation; vergleiche dazu auch die Biberaufgabe „Zug entladen".)

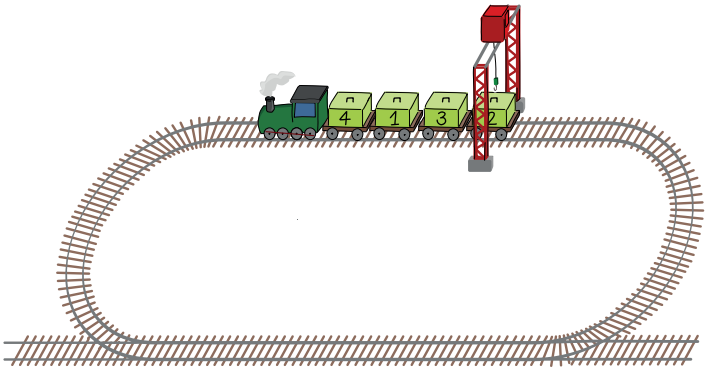
Wir haben nun drei Folgen – den Code (bzw. die Permutation), die sortierte Folge und die Inversionssequenz – und fassen sie in dieser Tabelle zusammen:

Code / Permutation	0	2	4	3	1
Sortierte Folge	0	1	2	3	4
Inversionssequenz	0	3	0	1	0

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

Zug entladen

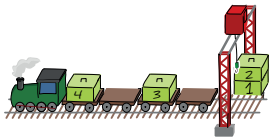
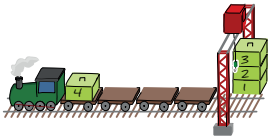
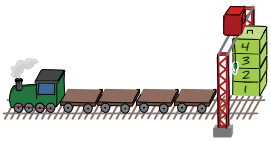
Ein Zug hat Wagen mit nummerierten Kisten. Zum Entladen der Kisten fährt der Zug eine Runde zum Kran. Der Kran ist unbeweglich, und der Zug kann nur vorwärts fahren.



Die Kisten sollen in der Reihenfolge ihrer Nummern entladen werden, Kiste 1 zuerst.

Der Zug fährt Wagen für Wagen die Kisten unter den Kran. Nur wenn eine Kiste an der Reihe ist, entlädt der Kran die Kiste. Wenn der Zug unter dem Kran durch ist, können noch Kisten auf den Wagen sein. Dann muss der Zug noch eine Runde zum Kran fahren.

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

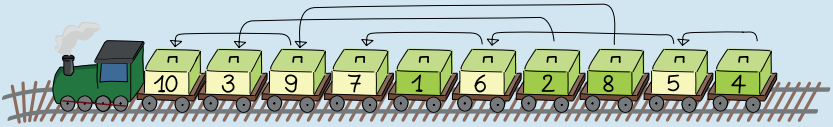
Runde 1	Runde 2	Runde 3
		
Kiste 4 ist nicht an der Reihe, Kiste 1 wird entladen, Kiste 3 ist nicht an der Reihe, Kiste 2 wird entladen.	Kiste 4 ist nicht an der Reihe, Kiste 3 wird entladen.	Kiste 4 wird entladen.

Wie viele Runden muss dieser Zug fahren, bis alle Kisten in der richtigen Reihenfolge entladen sind?

7 ist die richtige Antwort:

Die richtige Reihenfolge für das Entladen ist 1, 2, 3, 4, 5, 6, 7, 8, 9, 10; die Reihenfolge der Kisten auf dem Zug ist aber 10, 3, 9, 7, 1, 6, 2, 8, 5, 4. Nach der ersten Runde kann der Kran die Kisten 1 und 2 entladen. Nach der zweiten Runde kann der Kran die Kisten 3 und 4 entladen, nach weiteren Runden dann die Kisten 5, 6, 7 und 8, 9 und schließlich 10. Insgesamt fährt der Zug 7 Runden.

Zur Beantwortung kann man sich auch Folgendes überlegen: Der Zug muss immer dann eine weitere Runde fahren, wenn zwei in der richtigen Reihenfolge direkt hintereinander stehende Kisten auf dem Zug falsch herum stehen, also die Kiste mit der größeren Nummer vor der Kiste mit der kleineren Nummer kommt.



Da z. B. Kiste 3 vor Kiste 2 kommt, ist Kiste 3 nach der ersten Runde nicht an der Reihe und kann nicht entladen werden. Der Zug muss eine weitere Runde fahren, damit auch Kiste 3 entladen werden kann. Für das Nummernpaar (2,3) stehen also die Kisten „falsch“ auf dem Zug. Beim gegebenen Zug gibt es sechs solcher Paare: (2,3), (4,5), (5,6), (6,7), (8,9) und (9, 10). Deshalb muss der Zug 6 zusätzliche Runden fahren, also insgesamt 7 Runden.

Das ist Informatik!
Wenn für eine beliebige Zahl in der Folge 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 die Kiste mit der nächstgrößeren Zahl weiter vorne auf dem Zug liegt, sprechen Informatikerinnen und Informatiker von einer *Inversion,* das bedeutet Umkehrung. Für jede solche Umkehrung ist eine zusätzliche Runde erforderlich. Wenn wir die Anzahl der Umkehrungen zählen, erhalten wir die richtige Antwort zu dieser Biberaufgabe.

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


BIBER DER INFORMATIK