Biber der Informatik 2022
Aufgaben und Lösungen

OESTERREICHISCHE COMPUTER GESELLSCHAFT®
AUSTRIAN COMPUTER SOCIETY

Ein Wettbewerb der OCG

ocg.at/

Das österreichische Biber der Informatik Team 2022

Liam Bauman, Österreichische Computer Gesellschaft (OCG)
Wilfried Baumann, Österreichische Computer Gesellschaft (OCG)
Gerald Futschek, Technische Universität Wien
Benjamin Hirsch, Österreichische Computer Gesellschaft (OCG)
MartinKandlhofer, Österreichische Computer Gesellschaft (OCG)
Katharina Resch-Schobel, Österreichische Computer Gesellschaft (OCG)
Florentina Voboril, Technische Universität Wien

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

Waël Almoman, Collège Voltaire
Masiar Babazadeh, Scuola universitaria professionale della Svizzera italiana / Schweiz. Verein für Informatik in der Ausbildung (SVIA)
Michael Barot, Kantonsschule Schaffhausen
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)
Benjamin Hirsch, OCG
Juraj Hromkovic, ETH Zürich / ABZ
Martin Kandlhofer, OCG
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
Katharina Resch-Schobel, OCG
Giovanni Serafini, ETH Zürich / ABZ
Bernadette Spieler, PH Zürich
Florentina Voboril, Technische Universität Wien

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.

Der Biber der Informatik

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

Der Informatik-Biber will das allgemeine Interesse für das Fach Informatik wecken und gleichzeitig die Motivation für eine Teilnahme an Informatikwettbewerben stärken.

Der Informatik-Biber findet jährlich im November statt. An der 16. Austragung im Jahr 2022 beteiligten sich 302 Schulen und andere Bildungseinrichtungen mit 44.570 Schülerinnen und Schülern; das sind neue Rekordwerte, eine Steigerung um 45%. Die Möglichkeit, auch in Zweierteams zu arbeiten, wurde gern genutzt.

Die Teilnahme am Biber der Informatik 2022 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 2022 wurde in fünf Altersgruppen durchgeführt. Die Aufgaben jeder Altersgruppe sind in die Schwierigkeitsstufen leicht, mittel und schwer eingeteilt. In den Klassenstufen 3 bis 4 waren innerhalb von 30 Minuten 9 Aufgaben zu lösen, drei in jeder Schwierigkeitsstufe. In den Klassenstufen 5 bis 6 waren innerhalb von 35 Minuten 12 Aufgaben zu lösen, vier pro Schwierigkeitsstufe. In den Klassenstufen 7 bis 8, 9 bis 10 und 11 bis 13 waren innerhalb von 40 Minuten 15 Aufgaben zu lösen, jeweils fünf in jeder Schwierigkeitsstufe.

Die 33 Aufgaben des Bibers der Inforamtik 2022 sind auf Seite 6 gelistet, nach ungefähr steigender Schwierigkeit und mit einer informatischen Klassifikation ihres Aufgabenthemas. Ab Seite 7 folgen die Aufgaben nach ihrem Titel alphabetisch sortiert. Im Kopf sind die zugeordneten Altersgruppen und Schwierigkeitsgrade vermerkt. Eine kleine Flagge gibt an, aus welchem Bebras-Land die Idee zur jeweiligen Aufgabe stammt. Der Kasten am Aufgabenende enthält Erläuterungen zu Lösungen und Lösungswegen sowie eine kurze Darstellung des Aufgabenthemas hinsichtlich seiner Relevanz in der Informatik.

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

Wir laden die Schülerinnen und Schüler ein, auch 2023 wieder beim Informatik-Biber mitzumachen, und zwar in der Zeit vom 6. bis 17. November.

Bebras: International Challenge on Informatics and Computational Thinking

Der deutsche Informatik-Biber ist Partner der internationalen Initiative Bebras. 2004 fand in Litauen der erste Bebras Challenge statt. 2006 traten Estland, die Niederlande und Polen der Initiative bei, und auch Deutschland veranstaltete im damaligen Informatikjahr als „EI:Spiel blitz!“ einen ersten Biber-Testlauf. Seitdem kamen viele Bebras-Länder hinzu. Zum Drucktermin sind es weltweit 77, und weitere Länderteilnahmen sind in Planung. Insgesamt hatte der Bebras Challenge 2022 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 33 Aufgaben des Informatik-Biber 2022 stammen aus 20 Ländern: Belgien, Deutschland, Frankreich, Italien, Kanada, Lettland, Neuseeland, Niederlande, Nordmakedonien, Österreich, Philippinen, Schweiz, Slowakei, Südkorea, Türkei, Ungarn, Usbekistan, Vereinigtes Königreich, Vietnam und Zypern.

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

2022 war auch für Bebras ein besonderes Jahr. Wegen des Angriffskriegs auf die Ukraine wurde die Bebras-Mitgliedschaft von Russland und Belarus suspendiert.

BWINF hat sich bemüht, das Bebras-Mitglied Ukraine zu unterstützen und nach Deutschland geflüchteten ukrainischen Kindern und Jugendlichen eine Teilnahme am ukrainischen „Bober“ zu ermöglichen. 85 Schulen aus Deutschland und 8 aus der Schweiz haben dieses Angebot wahrgenommen. Statt der üblichen drei Biber-Charaktere aus anderen Ländern zeigen wir in diesem Biberheft nur den ukrainischen Biber.

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

Der ukrainische Biber

Die Österreichische Computer Gesellschaft

Die Österreichische Computer Gesellschaft (OCG) ist ein gemeinnütziger Verein zur Förderung der Informationstechnologie unter Berücksichtigung ihrer Wechselwirkungen mit Mensch und Gesellschaft. Der Verein bietet ein interdisziplinäres Forum und ist Dialogpartner für aktuelle und gesellschaftspolitisch relevante IT-Themen. Vernetzung und Förderung der Beziehungen zwischen Wissenschaft und Wirtschaft sind wichtige Anliegen. Darüber hinaus bietet die OCG ein standardisiertes, unabhängiges und qualitativ hochwertiges Weiterbildungsangebot im IT-Bereich und schlägt damit eine wichtige Brücke zur Arbeitswelt.

Österreichische und Internationale Informatik-Olympiade (IOI)

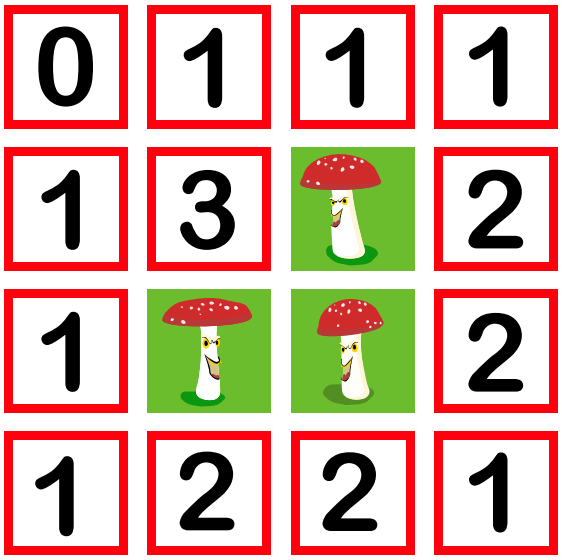
Jene Schülerinnen uns Schüler, die sich beim Computer-Programmmieren fit fuhlen und gerne in einer Gruppe von Gleichgesinnten auch anspruchsvolle Programmieraufgen lösen wollen, sollten sich bei der Österreichischen Inforamtik-Olympiade anmelden. Dabei wird zur Qualifikation verlangt, dass vorgegebene Programmieraufgaben gelöst und hochgeladen werden. Siehe www.ocg.at/ioi. Nach zwei Trainings-Workshops fahren die vier Bestplatzierten der Österreichischen Informatik-Olympiade zur Internationalen Olympiade, um sich mit den Besten aller Länder zu messen.

Aufgabenliste

Das sind die 33 Aufgaben des Informatik-Biber 2022, grob geordnet nach steigender Schwierigkeit und gelistet mit einer Klassifikation ihres informatischen Inhalts.

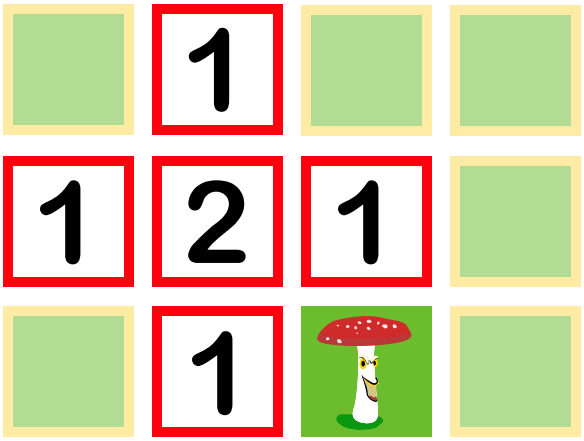
Achtung Fliegenpilz

Beim Spiel „Achtung Fliegenpilz“ ist zu Beginn genau ein Fliegenpilz zu sehen. Alle anderen Felder des Spielbretts sind zugedeckt. Deckst du ein Feld auf, erscheint entweder ein weiterer Fliegenpilz oder die Anzahl der Fliegenpilze auf den Nachbarfeldern. Wenn du alle Felder aufdeckst, auf denen kein Fliegenpilz versteckt ist, hast du gewonnen.
Hier ist ein Beispiel für ein vollständig aufgedecktes Spielbrett:

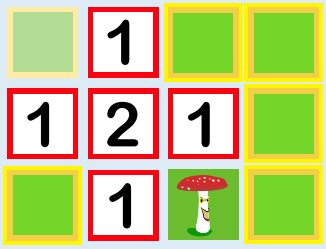


Du hast ein neues Spiel begonnen und bereits einige Felder aufgedeckt – siehe unten.

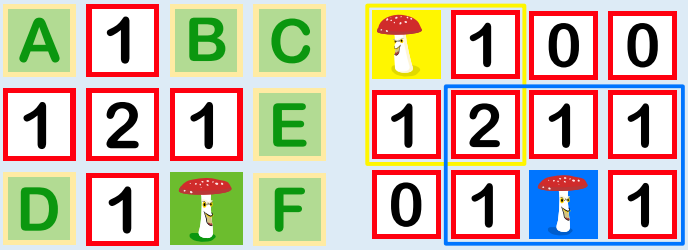
Auf welchen der übrigen Felder ist sicher kein Fliegenpilz?



So ist es richtig:



Um die richtige Antwort zu erklären, versehen wir die zugedeckten Felder mit Buchstaben. Außerdem sagen wir, dass eine Zahl *N* auf einem Feld „verbraucht“ ist, wenn bereits auf *N* Nachbarfeldern dieser Zahl je ein Fliegenpilz aufgedeckt ist; auf anderen Nachbarfeldern kann dann kein Fliegenpilz mehr sein.



- Auf Feld D ist kein Fliegenpilz, weil die Zahl 1 rechts daneben verbraucht ist.
- Auf den Feldern B, C, E und F ist kein Fliegenpilz, weil die gemeinsame Nachbarzahl 1 dieser Felder verbraucht ist.
- Auf Feld A ist ein Fliegenpilz, weil sonst die Nachbarzahlen 1, 2 und 1 die Anzahl der Fliegenpilze auf ihren Nachbarfeldern nicht korrekt angeben würden.

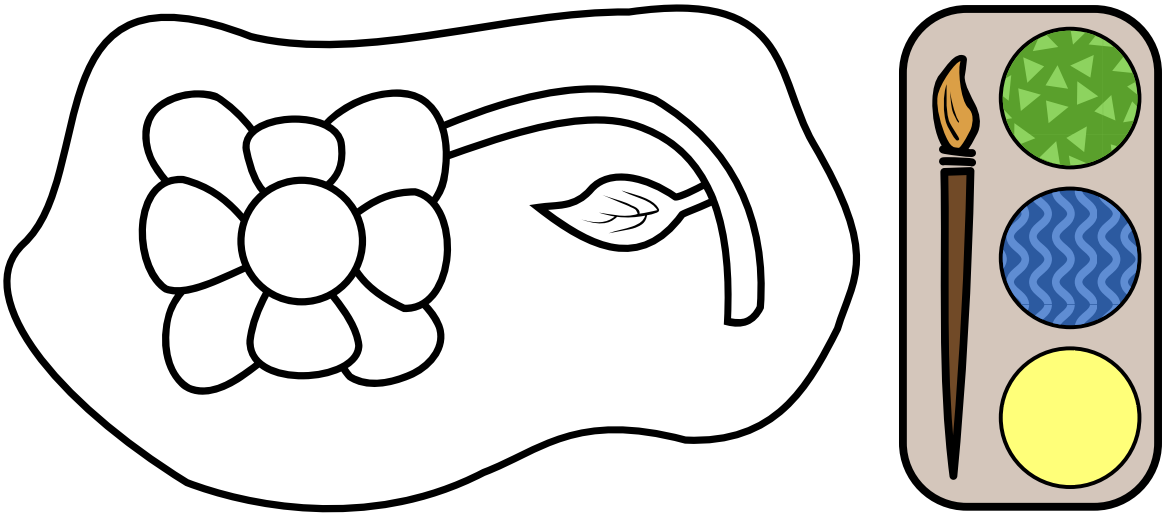
Also ist auf Feld A ein Fliegenpilz versteckt. Die Felder B, C, D, E und F dürfen aufgedeckt werden.
Im rechten Bild siehst du das vollständig aufgedeckte Spielbrett und wie die Zahlen auf den Feldern zustandekommen.

Das ist Informatik!

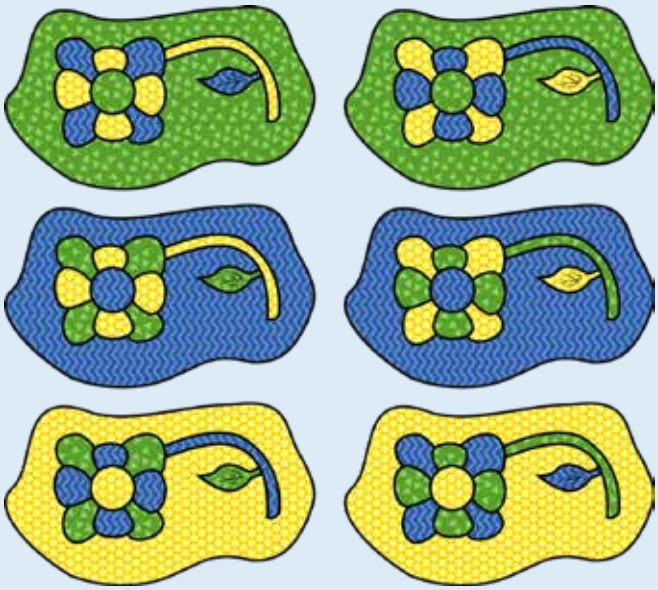
Wie könnte ein Computerprogramm diese Biberaufgabe lösen? Wenn mindestens ein Feld mit einem Fliegenpilz aufgedeckt ist, können einfache Regeln aufgestellt werden: Bei einem Feld mit der Zahl 0 hat kein Nachbarfeld einen Fliegenpilz, man kann sie also alle aufdecken. Wenn ein Feld mit der Zahl 1 bereits ein Nachbarfeld mit einem aufgedeckten Fliegenpilz hat, dann kann es auf allen anderen Nachbarfeldern keinen weiteren Fliegenpilz geben. Wenn solche Regeln für jede Zahl genau formuliert sind, könnte ein Computer immer wieder für jedes Feld des Spielbretts prüfen, ob eine der Regeln anwendbar ist – so lange, bis entweder alle Felder aufgedeckt sind oder es kein Feld mehr gibt, das noch aufgedeckt werden kann. Insgesamt ergibt sich so ein Algorithmus, der ausgeführt werden kann, um im Spiel (mit mindesten einem aufgedeckten Fliegenpilz) so erfolgreich zu sein wie möglich.

Ausmalbild

Male das ganze Bild aus, mit den drei farbigen Mustern.
Zwei Flächen, die sich berühren, dürfen nicht die gleiche Farbe haben.

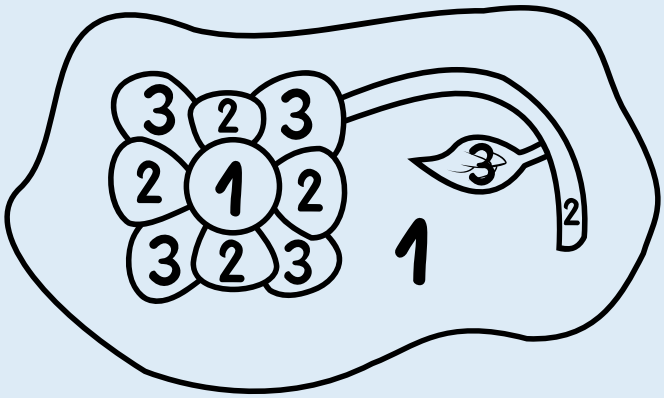


So ist es richtig:
Hier siehst du alle Möglichkeiten, das Bild richtig auszumalen – also so, dass die Bedingung aus der Aufgabe erfüllt ist:



Jede einzelne Möglichkeit ergibt sich, wenn man für die äußere Fläche eine Farbe wählt und dann die anderen Flächen nach und nach so bemalt, dass die Bedingung erfüllt ist. Dabei kann nur die Mitte der Blume die gleiche Farbe bekommen wie die äußere Fläche. Die anderen Flächen muss man mit den beiden anderen Farben abwechselnd bemalen; dabei kann man mit dem Stiel anfangen.

Wenn wir der Farbe für die äußere Fläche die Nummer 1 geben, der Farbe für den Stiel die 2 und der anderen Farbe die 3 geben, sieht das so aus:



Weil es sechs Möglichkeiten gibt, den Nummern 1, 2 und 3 die Farben Gelb, Grün und Blau zuzuordnen, gibt es sechs Möglichkeiten, das Bild richtig auszumalen.

Das ist Informatik!

In dieser Biberaufgabe hast du den Flächen Farben so zugeordnet, dass benachbarte Flächen unterschiedliche Farben haben. Informatik und Mathematik kennen solche „Färbungsprobleme“. Ganz ähnlich wie bei der Blume sollen zum Beispiel Landkarten so gefärbt werden, dass benachbarte Länder nicht gleich gefärbt sind. Es ist für jede Landkarte möglich, eine solche Färbung mit höchstens vier Farben zu finden. Das ist nicht einfach einzusehen. Erst 1976 konnten die Mathematiker Kenneth Appel und Wolfgang Haken das beweisen. Dabei mussten sie Methoden der Informatik verwenden, um eine Vielzahl von Ausnahmen und Gegenbeispielen zu überprüfen. Solche Computerbeweise werden aber kritisiert. Sie sind in der Regel so komplex, dass andere Menschen sie ohne Computer nicht nachvollziehen können.

Auch viele andere Probleme können als Färbungsprobleme aufgefasst werden. Dabei ist es meist wichtig, möglichst wenige Farben zu verwenden. Beispiele sind die Einteilung der Teams bei einem Sportturnier, das Einteilen von Personen in Gruppen oder die Zuteilung von Frequenzen für Radiosender.

Bei realistischen Anwendungen von Färbungsproblemen gibt es meistens sehr viel mehr Flächen als bei der Blume in dieser Biberaufgabe. Dann kann man die Lösung nicht ohne Hilfsmittel finden. Wie gut, dass die Informatik Methoden kennt, Färbungsprobleme mit Computerhilfe zu lösen.

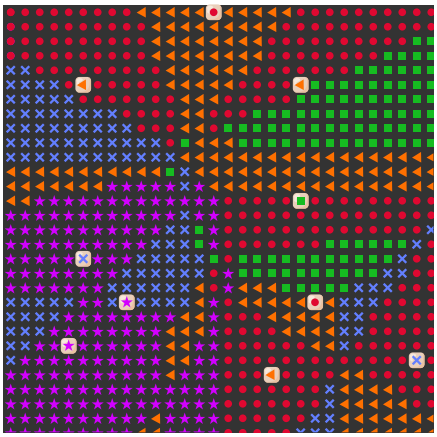
Bemalte Böden

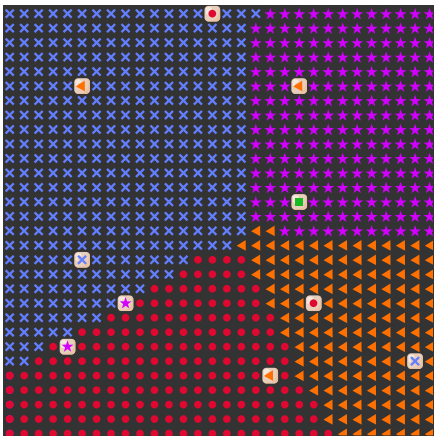
Der Boden eines quadratischen Raumes ist in 30 x 30 Felder unterteilt. Auf zehn Feldern liegen solche Chips mit farbigen Symbolen: 

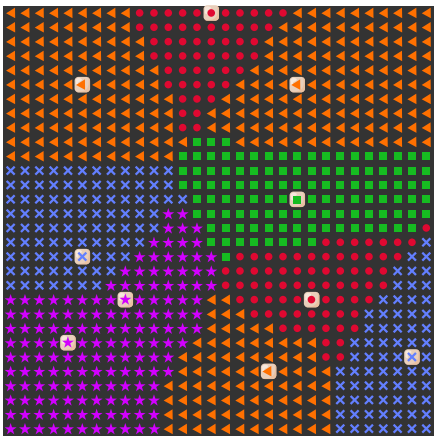
Ein Roboter soll den Boden mit diesen Symbolen bemalen,
Feld für Feld. Er hat dafür vier verschiedene Regeln.

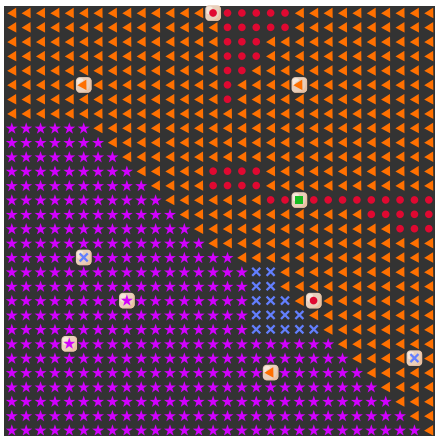
Auf einem Feld, auf dem kein Chip liegt, malt er …
1 … das Symbol des Chips, der ihm am nächsten ist.
2 … das Symbol des Chips, der am weitesten von ihm entfernt ist.
3 … das Symbol des Chips, der ihm am zweitnächsten ist.
4 … das Symbol, das bei den 6 am nächsten liegenden Chips am häufigsten vorkommt.
Der Roboter bemalt alle Felder nach derselben Regel. Wenn die Regel für ein Feld mehrere mögliche Symbole ergibt, sucht der Roboter sich zufällig eines davon aus.

Unten siehst du für jede Regel, wie der Boden am Ende bemalt ist.
Welcher Boden passt zu welcher Regel?



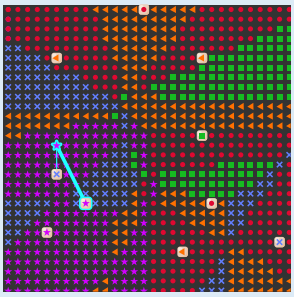
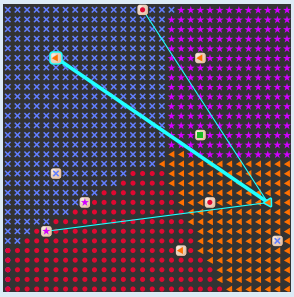
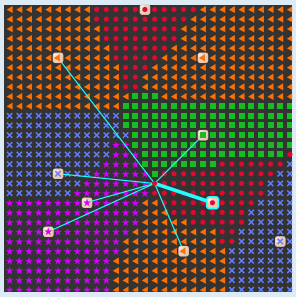
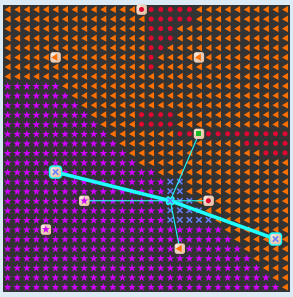






So ist es richtig:

Da alle Felder eines Bodens nach derselben Regel bemalt werden, genügt es, jeweils ein Feld zu überprüfen. Für jeden Boden betrachten wir ein anderes Feld:

Regel 3	Regel 2	Regel 1	Regel 4
			
Das Feld ist mit dem Symbol ★ bemalt, weil ein Chip ★ am zweitnächsten ist.	Das Feld ist mit dem Symbol ◀ bemalt, weil ein Chip ◀ am weitesten entfernt ist.	Das Feld ist mit dem Symbol ● bemalt, weil ein Chip ● am nächsten liegt.	Das Feld ist mit dem Symbol ✕ bemalt, weil bei den 6 am nächsten liegenden Chips ✕ am häufigsten vorkommt.

Das ist Informatik!

Die „Mal-Regeln“ in dieser Biberaufgabe sind Algorithmen, nach denen der Roboter die Bodenfläche des Raumes in Bereiche einteilen kann. Die algorithmische Teilung von Flächen bzw. Ebenen oder auch dreidimensionalen Räumen ist auch in der Informatik von Bedeutung, etwa bei Simulationen und in der Computergrafik.

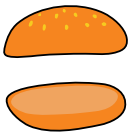
Besonders interessant ist Regel 1: Sie unterteilt die Fläche in *Voronoi-Regionen,* benannt nach dem ukrainischen Mathematiker Georgi Feodosjewitsch Woronoi (1868-1908). Jede Region wird durch ein Zentrum bestimmt, und alle Punkte einer Region liegen keinem anderen Zentrum näher als dem eigenen. *Voronoi-Diagramme* stellen die Grenzen von Voronoi-Regionen dar. Voronoi-Regionen finden vielfach Anwendung, beispielsweise in der Mobilfunkversorgung. Oder man nutzt sie für die Analyse von Fußballspielen oder anderem sozioökonomischen Verhalten, etwa den Beziehungen zwischen der Bevölkerung und den nächstgelegenen Schulen, Krankenhäusern oder bestimmten Dienstleistern.

In der Informatik sind verschiedene Algorithmen bekannt, um aus einer Menge von Punkten einer Ebene die Voronoi-Regionen bzw. das Voronoi-Diagramm zu berechnen, das diese Punkte als Zentren hat. Der Algorithmus von Fortune ist besonders effizient und folgt der allgemeinen und besonders für geometrische Probleme bedeutenden Idee der *Sweep-Line-Algorithmen.*

https://de.wikipedia.org/wiki/Voronoi-Diagramm

Biber-Burger

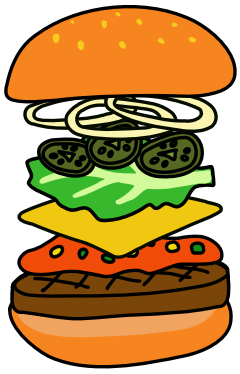
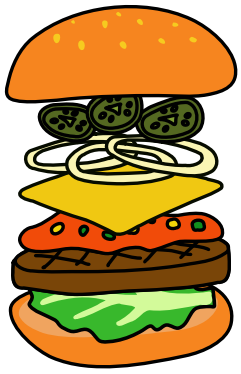
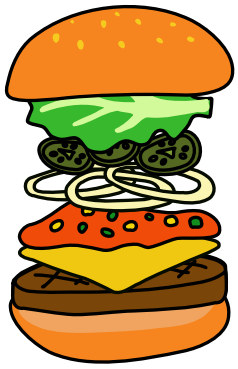
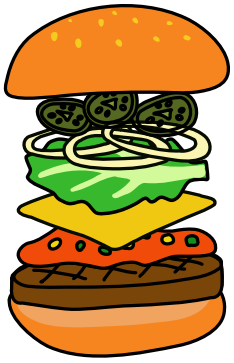
Für Biber-Burger gibt es diese Zutaten:

Brötchen	Patty	Soße	Gurken	Salat	Zwiebeln	Käse
						

Echte Biber-Burger erfüllen alle diese Bedingungen:

1. Die Soße ist direkt auf dem Patty.
2. Das Patty und der Käse liegen unter den Gurken, dem Salat und den Zwiebeln.
3. Die Zwiebeln berühren nicht das Brötchen.

Nur einer dieser Burger ist ein echter Biber-Burger. Welcher?

A	B	C	D
			

Antwort D ist richtig:

Für jeden Burger muss man prüfen, ob er alle drei Bedingungen erfüllt. Nur dann ist er ein echter Biber-Burger.

Antwort A: Dieser Burger erfüllt die Bedingungen 1 und 2. Aber die Zwiebeln berühren oben das Brötchen, also erfüllt er nicht die Bedingung 3.

Antwort B: Dieser Burger erfüllt die Bedingung 1. Aber der Salat ist unter dem Fleisch und dem Käse, also ist Bedingung 2 nicht erfüllt.

Antwort C: Dieser Burger erfüllt die Bedingungen 2 und 3. Allerdings ist die Soße nicht direkt auf dem Patty, also ist Bedingung 1 nicht erfüllt.

Antwort D: Dieser Burger erfüllt alle Bedingungen und ist ein echter Biber-Burger!

Das ist Informatik!

Ein echter Biber-Burger muss also die drei in dieser Biberaufgabe genannten Bedingungen erfüllen. Erfüllt ein Burger nur eine Bedingung nicht, ist er kein echter Biber-Burger. Bedingungen spielen in der Informatik eine besonders wichtige Rolle. In allen Programmiersprachen gibt es eine oder mehrere Anweisungen zur Überprüfung von Bedingungen. Ihre englischen Namen sind häufig „if“ und „case“. Mit Hilfe solcher *bedingter Anweisungen* könnte man also prüfen, ob ein Burger ein echter Biber-Burger ist – sofern man es schafft, den Burger in einen Computer zu stecken.

Es ist relativ leicht, ein Objekt wie einen Biber-Burger auf vorgegebene Bedingungen zu überprüfen. Es ist sehr viel schwieriger, in einer Menge von Objekten (die möglicherweise unendlich sein kann) eines zu finden, das eine potenziell große Menge komplexer Bedingungen erfüllt. In der Informatik ist dieses schwierige Problem als „Constraint Satisfaction Problem“ (dt.: Bedingungserfüllungsproblem, kurz: CSP) bekannt. Es ist interessant, sich mit CSPs zu beschäftigen, weil viele andere schwierige Probleme als CSP formuliert werden können – oder auch logische Rätsel wie Sudokus. Erkenntnisse über CSPs führen dann zu Erkenntnissen über diese anderen Probleme.

Bibermeisterschaft

An der Bibermeisterschaft nehmen 8 Biber teil. Es gibt drei Runden. In jeder Runde sammelt jeder Biber Punkte.

- In Runde 1 werden zwei Teams aus je 4 Bibern gebildet. Die Punkte der einzelnen Biber im Team werden aufsummiert. Das Team mit den meisten Punkten gewinnt. Diese Biber sind in Runde 2 die Spitzengruppe und spielen um die Plätze 1 bis 4. Die Verlierer machen die Plätze 5 bis 8 unter sich aus.

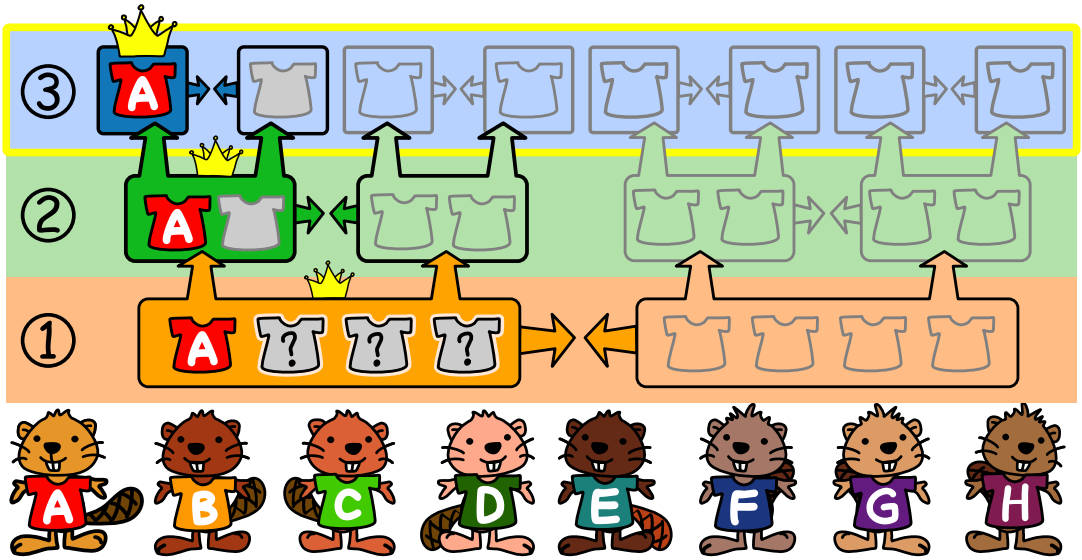
- Runde 2 wird mit denselben Regeln durchgeführt. Die Teams bestehen jetzt aus 2 Bibern. Das Gewinnerteam der Spitzengruppe spielt in Runde 3 um Platz 1. Die Verlierer machen die Plätze 3 und 4 unter sich aus.

- In Runde 3 treten die Biber einzeln gegeneinander an.

Biber Ada  erreicht Platz 1 der Bibermeisterschaft. Hier siehst du die Punkte, die jeder Biber in jeder Runde erzielt hat:

Biber								
Runde 1	15	16	19	18	17	20	19	19
Runde 2	20	27	30	24	28	24	30	30
Runde 3	10	14	11	15	16	13	9	12

Welche drei Biber waren in Adas Team in Runde 1?



So ist es richtig:

Die Biber 

, 

und 

waren in Adas Team in Runde 1.
In Runde 3 wird einzeln gegeneinander gespielt. Biber G ist der einzige, der in Runde 3 weniger Punkte als Ada hat. In Runde 3 hat Ada also gegen ihn gespielt, und in Runde 2 müssen sie demnach im selben Team gewesen sein.

In Runde 2 haben sie zusammen 50 Punkte erreicht. Dieser Wert muss höher sein als die Gesamtpunktzahl des Zweierteams, gegen das sie in Runde 2 gespielt haben. Die beiden Biber D und F sind das einzige Paar, dessen Punktsumme kleiner als 50 ist. Deshalb müssen sie in Runde 1 in demselben Team gewesen sein wie Ada und Biber G.

Das ist Informatik!

Um diese Aufgabe zu lösen, könnten wir so vorgehen: Zuerst zählen wir alle möglichen Teams in der ersten Runde systematisch auf.

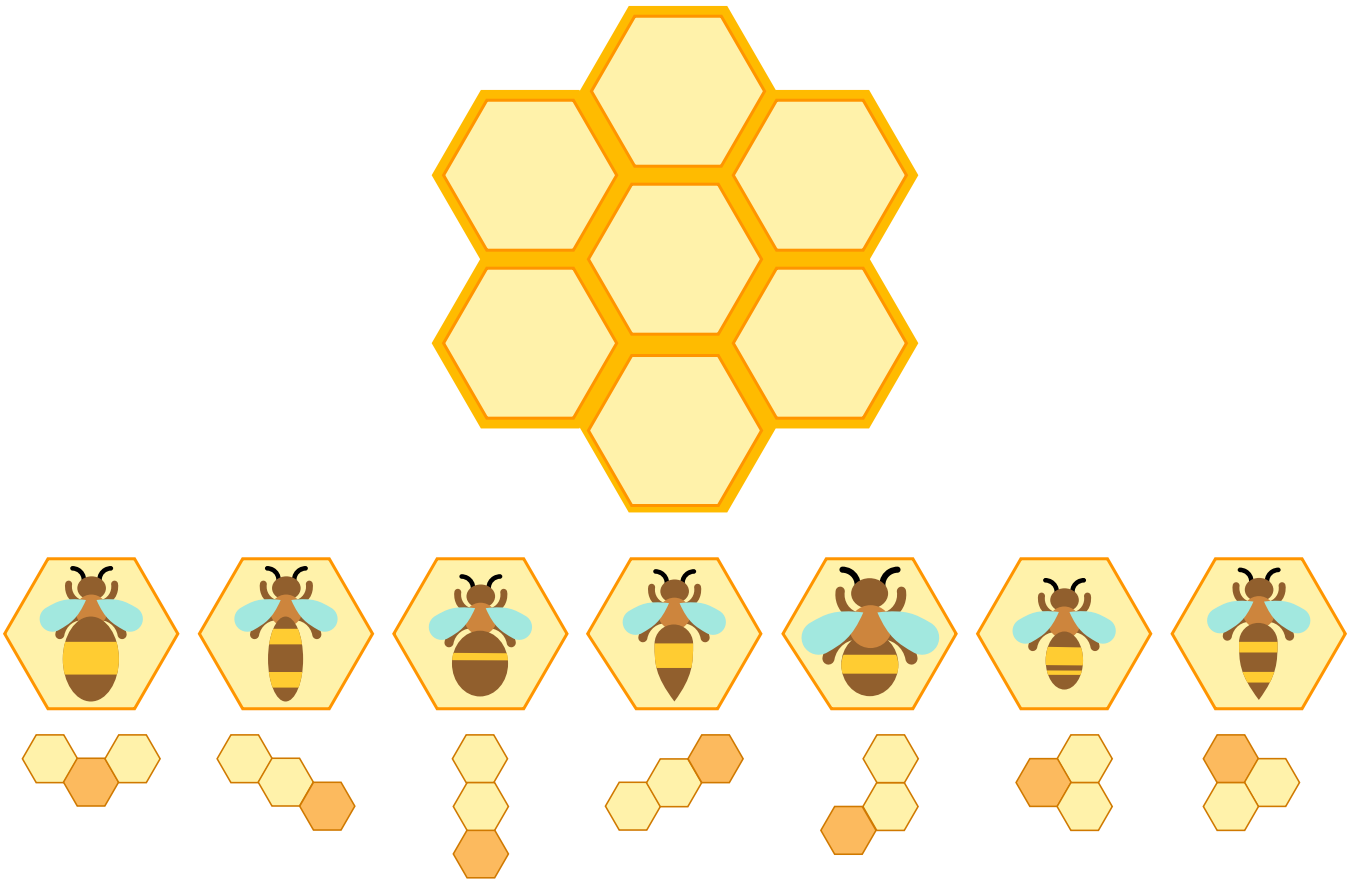
Dazu genügt es, alle $\binom{7}{3} = 35$ Möglichkeiten zu betrachten, aus den sieben Bibern ohne Ada die drei freien Plätze in Adas Team zu besetzen; die jeweils verbleibenden vier Biber bilden das andere Team. Für jede dieser Team-Paarungen betrachten wir die Ergebnisse in Runde 1, in Runde 2 und im Finale, bevor wir entscheiden können, welche drei Biber in Runde 1 tatsächlich in Adas Team waren. Das dauert, und die Buchführung über alle möglichen Ergebnisse ist aufwändig.

Wenn in der Informatik ein Problem gelöst werden soll, ist es meist keine gute Idee, eine solche „Brute-Force-Strategie“ anzuwenden, also alle Möglichkeiten aufzuzählen, die eine Lösung des Problem sein könnten, und jeweils zu überprüfen, ob es sich auch wirklich um eine bzw. die beste Lösung handelt. Für die meisten Probleme und Eingabedaten dauert das viel zu lange, und es werden effizientere Ansätze benötigt. Wie die obige Erklärung zeigt, lässt sich diese Biberaufgabe effizienter lösen, wenn die Lösung nicht vorwärts (von Runde 1 aus), sondern rückwärts gesucht wird, also von Runde 3 aus. Eine *Rückwärtssuche* ist besonders geeignet, wenn eine Lösung gesucht wird, die bestimmte Bedingungen erfüllen muss.

Bienenwaben

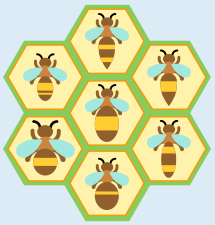
Die Bienen wollen auf die sieben Waben.
Unter jeder Biene zeigt ein Hinweis, wann eine Wabe für sie richtig ist.

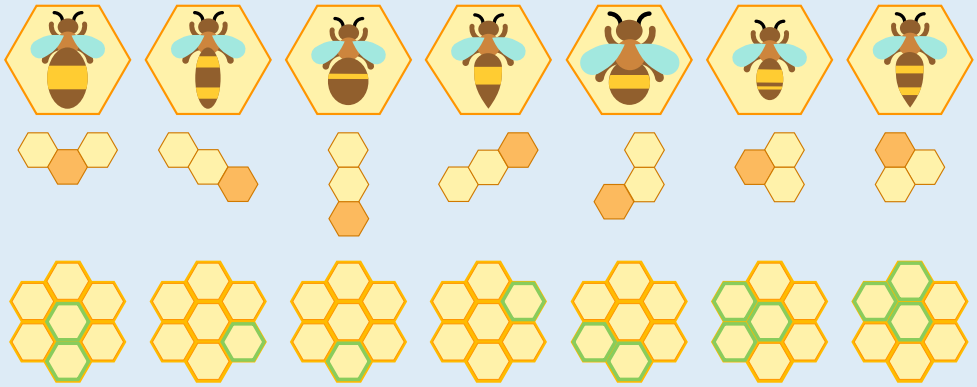
Ziehe jede Biene auf eine richtige Wabe!



So ist es richtig:

Wie kannst du diese Biberaufgabe geschickt lösen? Schau dir die Hinweise für die Bienen genauer an. Unten siehst du noch einmal die Bienen und die Hinweise. Außerdem wird für jede Biene angezeigt, welche Waben wegen des Hinweises für sie richtig sein können.





Du siehst, dass es für drei Bienen jeweils nur eine richtige Wabe gibt. Diese Bienen kannst du direkt auf diese Waben ziehen. Danach bleibt für die erste Bienen ganz links und die fünfte Biene auch nur noch je eine richtige Wabe übrig, und du kannst sie dorthin ziehen. Und dann ist auch für die beiden rechten Bienen nur noch je eine richtige Wabe übrig.

Du kannst auch versuchen, die Bienen von links der Reihe nach auf die Waben zu ziehen; zuerst Biene 1, dann 2 usw. Wenn du dabei aber Biene 1 in die Wabe ganz unten ziehst, ist diese Wabe für Biene 3 nicht mehr frei. Damit Biene 3 auf ihre einzige richtige Wabe kann, musst du Biene 1 also noch einmal „umziehen“. Das ist unnötig umständlich. Es lohnt sich also, die Hinweise genau zu betrachten und geschickt vorzugehen.

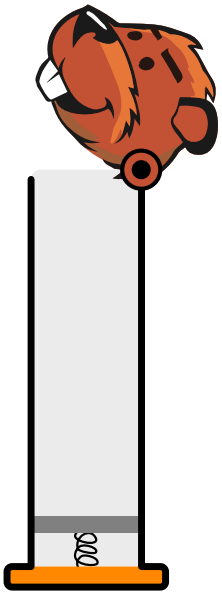
Das ist Informatik!

In dieser Biberaufgabe müssen sieben Bienen auf sieben verschiedenen Waben untergebracht werden. Dafür gibt es über 5.000 Möglichkeiten! Wenn man die Hinweise berücksichtigt, verringert sich die Anzahl der Möglichkeiten stark. Es wäre aber immer noch aufwändig, alle verbliebenen Möglichkeiten auszuprobieren. Der Schlüssel zu einer schnellen Lösung der Aufgabe ist die richtige Reihenfolge bei der Platzierung der Bienen. Konkret hatten wir dazu diese Idee: Wenn man zuerst diejenigen Bienen platziert, für die es nur eine richtige Wabe gibt, macht man nichts falsch, und mit ein wenig Glück sind danach die Möglichkeiten für die anderen Bienen noch weiter eingeschränkt.

Unsere Idee hat also geholfen, die Aufgabe schneller zu lösen, und gleichzeitig nicht verhindert, überhaupt eine Lösung zu finden. Solch eine hilfreiche Idee nennt man in der Informatik Heuristik. Bei besonders schwierigen Problemen geht es nicht ohne Heuristiken. Wenn bei einem Problem aber nicht nur irgendeine, sondern die beste Lösung gesucht ist, kann eine Heuristik zu Lasten der Lösungsqualität gehen. Die Informatik untersucht dann, wie stark die Heuristik-Lösung von einer besten Lösung abweichen kann. Dann kann man sich gründlich überlegen, ob man sich mit der Heuristik-Lösung zufrieden gibt. Ein Beispiel ist das Planen einer Route in einem Navigationssystem. Um garantiert die beste Route zu finden, müsste jede der sehr vielen möglichen Routen berechnet und mit den anderen verglichen werden. Das wäre mit hohem Rechenaufwand verbunden. Man kann aber auch heuristisch vorgehen und z.B. nur die Möglichkeiten ausprobieren, die mit hoher Wahrscheinlichkeit zu einer guten Lösung führen. So wird in kurzer Zeit eine gute Route ermittelt, anstatt sehr lange Zeit für die Berechnung der garantiert besten Route zu benötigen.

Bonbon-Spender

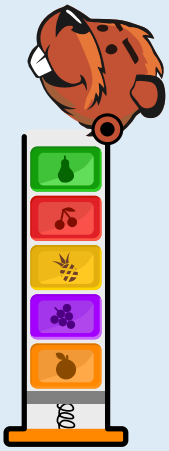
Anna füllt fünf Bonbons in einen Spender. Danach isst sie die Bonbons so nacheinander, wie sie oben aus dem Spender kommen. Sie möchte die Bonbons so nacheinander essen:



Wie muss Anna die Bonbons in den Spender füllen?
Ziehe die Bonbons richtig in den Spender.

So ist es richtig:

Der Spender gibt immer nur das oberste Bonbon aus. Das Bonbon, das ganz unten in den Spender gefüllt wird, kommt also als letztes heraus. Damit die Bonbons so nacheinander heraus kommen, wie Anna sie essen möchte, muss der Spender so gefüllt werden:



Das ist Informatik!

Wenn Anna die Bonbons in den Spender füllt, muss sie auf die richtige Reihenfolge achten. Bei der Herausgabe der Bonbons kehrt der Spender die Reihenfolge, in der die Bonbons hineingefüllt wurden, nämlich um. Das liegt daran, dass der Spender das oberste Bonbon – und damit das letzte Bonbon, was eingefüllt wurde – zuerst ausgibt. Das könnte man auf Englisch so ausdrücken: „last in, first out“ (zuletzt hinein, zuerst heraus), kurz: LIFO.

Die Informatik kennt eine Struktur zur Speicherung von Daten, die ähnlich funktioniert wie der Spender, nämlich nach dem LIFO-Prinzip. Diese Struktur heißt Stack, auf Deutsch: Stapel. Natürlich kann man sie nur dann sinnvoll verwenden, wenn man auf die gespeicherten Daten nach dem LIFO-Prinzip zugreifen will. Das scheint eine starke Einschränkung zu sein. Aber weil Stacks in Computern sehr einfach zu realisieren sind, werden sie dennoch recht häufig verwendet.

Das LIFO-Prinzip kommt beim Informatik-Biber 2022 auch in der Aufgabe „Muttern und Schrauben“ vor.

Computerviren

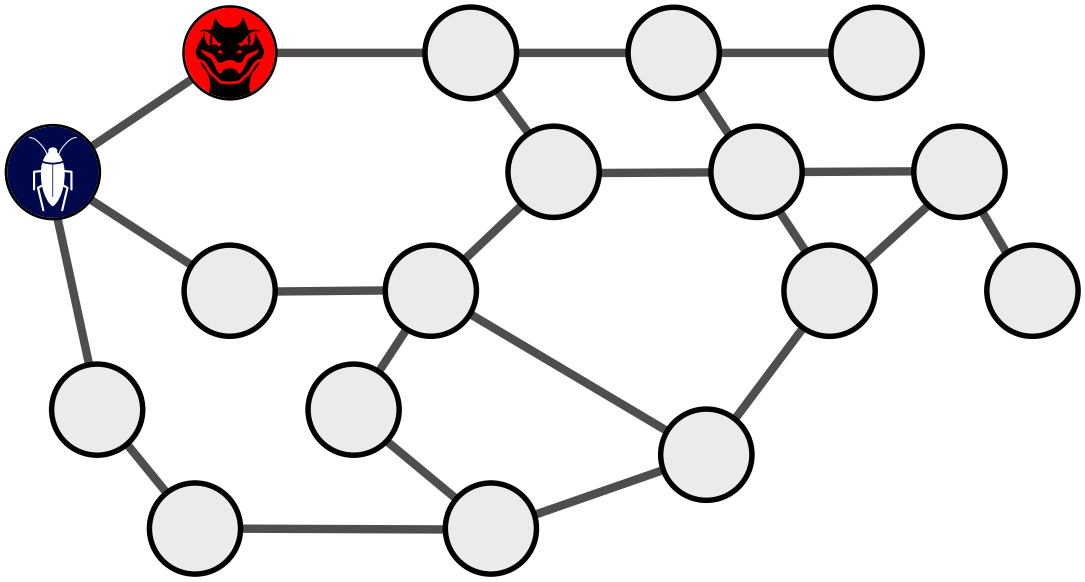
In einem Computernetz haben sich zwei Netzknoten mit Computerviren infiziert: einer mit dem Virus **BlueBug**, ein anderer mit dem Virus **RedRaptor**.

Immer am Morgen breiten sich beide Viren aus. Jedes Virus infiziert dann zusätzlich alle Knoten, die mit den von ihm bereits infizierten Knoten direkt verbunden sind. Wenn ein Knoten mit beiden Viren infiziert ist, schaltet er nach einigen Stunden wegen Überlastung ab.

Die Viren können sich an den folgenden Tagen von dort also nicht weiter ausbreiten.

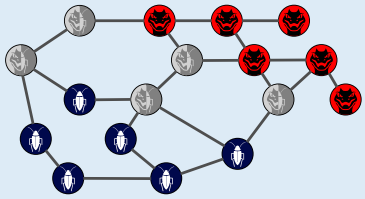
Unten siehst du das Computernetz mit den Knoten und ihren direkten Verbindungen.
Die beiden zu Beginn infizierten Knoten sind markiert. Nach einigen Tagen sind alle Knoten mit einem Virus infiziert oder sogar abgeschaltet.

Welche Knoten sind dann mit welchem Virus infiziert oder abgeschaltet?
Wähle für jeden Knoten die richtige Markierung.

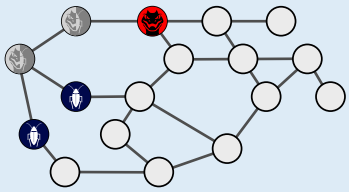
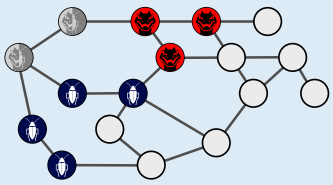
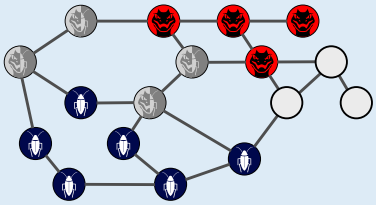
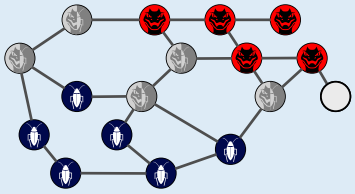


So ist es richtig:

Nach fünf Tagen sind alle Netzwerkknoten infiziert oder abgeschaltet, und zwar so:



Die Viren haben sich in dieser Zeit so ausgebreitet:

Nach **1 Tag** sind fünf Netzknoten infiziert. Die beiden zu Beginn infizierten Knoten sind nun mit beiden Viren infiziert und deswegen abgeschaltet:	Nach **2 Tagen** sind vier weitere Knoten infiziert:
	
Nach **3 Tagen** sind zwei Knoten mit beiden Viren infiziert und nun ebenfalls abgeschaltet. Zudem sind drei weitere Knoten mit „BlueBug“ und zwei mit „RedRaptor“ infiziert:	Nach **4 Tagen** ist ein weiterer Netzwerkknoten ausgeschaltet. „BlueBug“ kann sich nun nicht mehr weiter ausbreiten. Der letzte Knoten wird am fünften Tag von „RedRaptor“ infiziert werden.
	

Das ist Informatik!

In Computernetzen stellen Viren und andere Schadsoftware (allgemein englisch: *Malware*) eine große Bedrohung dar. Sie beeinflussen nicht nur die Leistungsfähigkeit der betroffenen Computer, sondern in der Regel haben sie eine „Nutzlast“ (*payload)*, die gezielt Schaden anrichtet. In manchen Fällen werden übertragene Daten mitgelesen und so sensible Informationen wie Passwörter oder Benutzerdaten herausgefunden und an einen Auftraggeber übermittelt (*Spyware*). In anderen Fällen werden Daten auf dem befallenen Computer verschlüsselt. Damit der Zugriff auf die Daten wieder möglich wird, muss erst ein Geldbetrag auf ein anonymes Konto überwiesen werden (*Ransomware*). Manchmal werden Gruppen infizierter Computer ferngesteuert, um Angriffe auf andere Computer durchzuführen (*Botnets*).

Dass eine Malware einen Computer zur Abschaltung bringt, wie in dieser Biberaufgabe, ist meist nicht beabsichtigt, denn dadurch wird die Verbreitung gestoppt. Manche Malware wird aber gezielt für Sabotage und Cyberkrieg (*Cyberwarfare*) entwickelt. Dadurch können betroffene Computer sogar dauerhaft beschädigt werden.

Die Einspielung aktueller Sicherheitsupdates ist eine wichtige Voraussetzung für die Abwehr von Malware. „Virenschutz“-Software kann den Schutz verbessern, ist aber in manchen Betriebssystemen schon enthalten. Regelmäßige Datensicherungen und Wachsamkeit im Bezug auf ungewöhnliches Verhalten von Informatiksystemen sind aber unabdingbar.

Edelsteine

Peter hat einige Edelsteine. Sie sind alle unterschiedlich wertvoll. Sarah kennt Peters Edelsteine, aber nicht deren Wert. Sie will wissen, welcher Stein der wertvollste ist. Dazu macht sie Folgendes dreimal:

- Sie wählt vier von Peters Steinen aus und fragt ihn, welcher davon der wertvollste Stein ist.

Jedesmal wählt sie die vier Steine beliebig neu aus, und Peter gibt ihr jedesmal eine ehrliche Antwort. Danach weiß Sarah, welcher Stein der wertvollste ist.

Wie viele Edelsteine kann Peter höchstens haben? **A** 8 **B** 10 **C** 11 **D** 12

Antwort B ist richtig:

Wenn Peter 10 Edelsteine hat, kann Sarah bei den ersten beiden Fragen insgesamt acht verschiedene Edelsteine auswählen. Die beiden „Gewinner" der einzelnen Fragen (also die Steine, die jeweils die wertvollsten der vier gewählten Steine sind) können auch „Gesamtsieger" sein, also der insgesamt wertvollste Stein. Die anderen sechs Steine scheiden aus. Bei der letzten Frage wählt sie die beiden Gewinner und die zwei bisher noch nicht gewählten Steine aus. Der Gewinner dieser Frage muss der Gesamtsieger sein.

Für 10 Steine kann Sarah also (unter anderem) so vorgehen, um den wertvollsten Stein zu finden.
Wenn Peter 11 Steine hat, kann sie das leider nicht schaffen:
Wenn Sarah, wie oben, bei den ersten beiden Fragen insgesamt acht verschiedene Steine vergleicht, verbleiben die beiden Gewinner und drei weitere Steine, also einer zu viel, um den Gesamtsieger mit der dritten Frage zu ermitteln. Wenn Sarah hingegen den Gewinner der ersten Frage bei der zweiten Frage mit 3 „neuen" Steinen vergleicht, kennt sie danach den wertvollsten der sieben gewählten Steine. Diesen Stein muss sie mit den vier weiteren Steinen vergleichen. Auch das ist ein Stein zu viel für die dritte Frage.
Wenn Sarah bei 11 Steinen für die ersten beiden Fragen nur sechs oder noch weniger verschiedene Steine auswählt, oder wenn Peter mehr als 12 Steine hat, kann Sarah nach drei Fragen erst recht nicht wissen, welcher Stein der wertvollste ist.

Das ist Informatik!

Sarahs Vorgehen bei der Bestimmung des wertvollsten Steins sieht zunächst aus wie ein Algorithmus. Aber wenn man genau hinsieht, fehlt noch etwas: nämlich eine Vorschrift, wie sie die vier Steine für ihre Frage auswählt. Dennoch ist es möglich, ihr Vorgehen zu analysieren: In dieser Biberaufgabe wird betrachtet, für wie viele Edelsteine ihr Vorgehen funktionieren kann, und zwar unabhängig von der Auswahl der Steine. Das Ergebnis – nämlich, dass ihr Vorgehen nur für maximal 10 Edelsteine funktioniert – gilt also gleich für eine ganze Klasse von Algorithmen, die sich aus der Kombination von Sarahs grundsätzlichem Vorgehen mit den verschiedenen Möglichkeiten zur Auswahl der Steine ergeben.

Die möglichst genaue Analyse von Algorithmen ist ein wichtiges Gebiet der Informatik. Für einen Algorithmus (bzw. für das in der realen Welt genutzte Informatiksystem, das eine Implementierung des Algorithmus verwendet) wollen wir nicht nur wissen, ob er funktioniert, also ob er ein gegebenes Problem löst, sondern auch, wie gut er funktioniert. Dieses „wie gut" kann dabei ganz unterschiedliche Bedeutungen haben: Wie gut sind die gefundenen Ergebnisse, etwa in Bezug auf ein Optimierungskriterium? Wie viele Ressourcen benötigt der Algorithmus, z.B. Zeit oder Speicherplatz? Wenn wir nicht wissen, wie lange wir auf die Reaktion eines Informatiksystems warten müssen bzw. mit wie viel Speicher wir es ausstatten müssen, nützt uns eine Lösungsgarantie nicht viel.

Die Analyse von Algorithmen ist nicht einfach. Wenn es also möglich ist, einen Teil eines zu analysierenden Algorithmus' zu verallgemeinern – wie bei Sarahs Algorithmus die Auswahl der Steine – und die Analyse dann für eine ganze Klasse von Algorithmen durchführen können, ist das Ergebnis der Analyse noch einmal deutlich wertvoller.

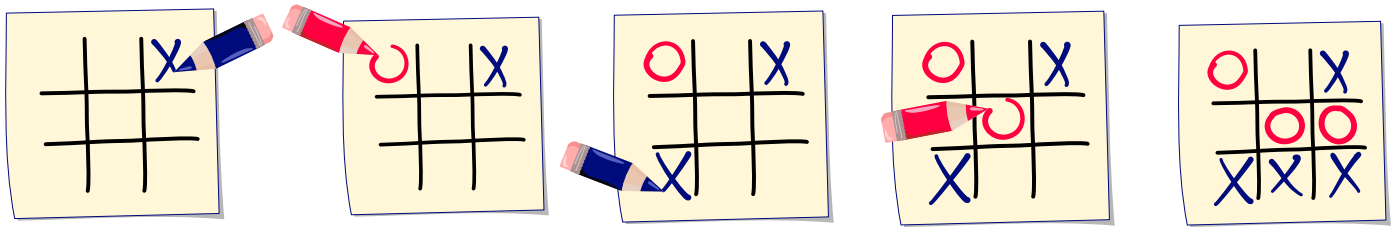
Endstand

Tic-Tac-Toe ist ein Spiel für zwei Personen.

In ein Raster mit 3 x 3 Feldern setzen die Spieler abwechselnd je ein Zeichen in ein freies Feld: immer zuerst der eine Spieler ein X, der andere ein O.

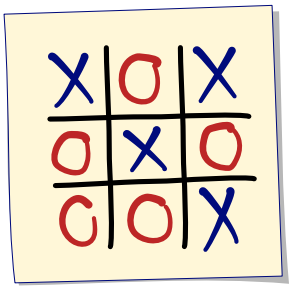
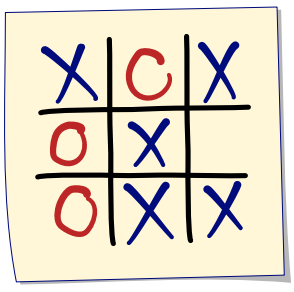
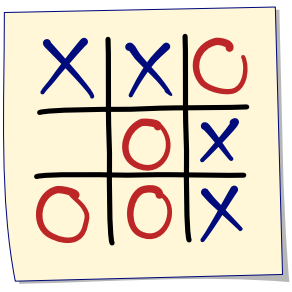
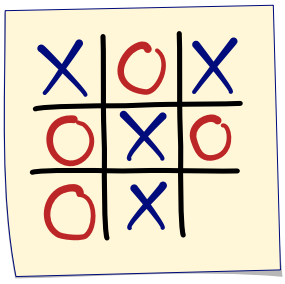
Wer zuerst drei (gleiche) Zeichen in eine Zeile, Spalte oder Diagonale setzen kann, gewinnt; dann ist das Spiel beendet. Wenn alle Felder besetzt sind und niemand gewonnen hat, endet das Spiel unentschieden.

Hier siehst du die Spielstände aus einem möglichen Spielverlauf: nach den ersten vier Spielzügen und nach dem letzten Zug. Der Spieler mit X gewinnt.



Den Spielstand am Ende eines Spiels nennen wir Endstand. Die Spielregeln legen genau fest, wie ein Endstand aussehen kann.

Nur eines der vier Bilder zeigt einen Endstand von Tic-Tac-Toe. Welches?

A	B	C	D
			

Antwort C ist richtig:

Antwort C zeigt einen Endstand. Beide Spieler haben vier Zeichen gesetzt. Der zweite Spieler hat mit seinem vierten O drei gleiche Zeichen in einer Diagonale und damit gewonnen. Damit war das Spiel beendet, keine weiteren Zeichen wurden gesetzt.

Antwort A zeigt keinen Endstand. Spieler X hat das Spiel gewonnen, aber Spieler O hat weitere Os in leere Felder gesetzt. Da der Gewinner immer das letzte Zeichen setzt, kann er niemals weniger Zeichen als der Verlierer setzen.

Antwort B zeigt keinen Endstand. Der Spielstand enthält 5 X, aber nur 3 O. Die Anzahlen der O und X können sich bei regelmäßigem Spielverlauf aber höchstens um 1 unterscheiden.

Antwort D zeigt keinen Endstand. Beide Spieler haben vier Zeichen gesetzt, aber es gibt noch keinen Gewinner und noch sind nicht alle Felder besetzt. Spieler X kann noch ein Zeichen setzen und wird das Spiel damit gewinnen.

Das ist Informatik!

Bei der Lösung dieser Biberaufgabe haben wir geprüft, ob die vier Antwort-Bilder einen Endstand des Spiels zeigen. Dabei haben wir Bedingungen für Endstände geprüft, die aus den Spielregeln von Tic-Tac-Toe logisch abgeleitet werden können:

- Die Differenz zwischen den Anzahlen von X und O muss 0, -1 oder 1 sein.
- Wenn kein Spieler gewonnen hat, müssen alle Felder gefüllt sein.
- Der Verlierer kann höchstens so viele Zeichen setzen wie der Gewinner.
- Ein Endstand kann höchstens eine Folge von drei gleichen Zeichen in einer Zeile, Spalte oder Diagonalen enthalten.

So wie diese „Endstand-Bedingungen“ lässt sich auch ein Endstand und insbesondere auch ein Gewinn-Spielstand aus den Spielregeln und dem Anfangs-Spielstand (das leere Raster) logisch ableiten. Das funktioniert, weil die Spielregeln von Tic-Tac-Toe als Logik-Regeln formuliert werden können. Wenn man diese Regeln mit Hilfe der Programmiersprache Prolog, in die logisches Ableiten sozusagen eingebaut ist, formuliert, erhält man ein Programm, das Tic-Tac-Toe spielen kann.

In den frühen Zeiten der Forschung zur „Künstlichen Intelligenz“ wurde logisches Ableiten als wichtiges Werkzeug gesehen, „intelligentes“ Verhalten mit Informatik-Systemen zu implementieren. Heute verwenden KI-Systeme vor allem statistische Verfahren, zu denen letztlich auch die „künstlichen neuronalen Netze“ gehören. Logisches Ableiten bzw. das „logische Programmieren“ mit Sprachen wie Prolog hat aber auch heute noch Anwendungen.

https://de.wikipedia.org/wiki/Prolog_(Programmiersprache)

Fiat Lux

Das Spiel “Fiat Lux” hat 8 Schalter, die an oder aus sein können.

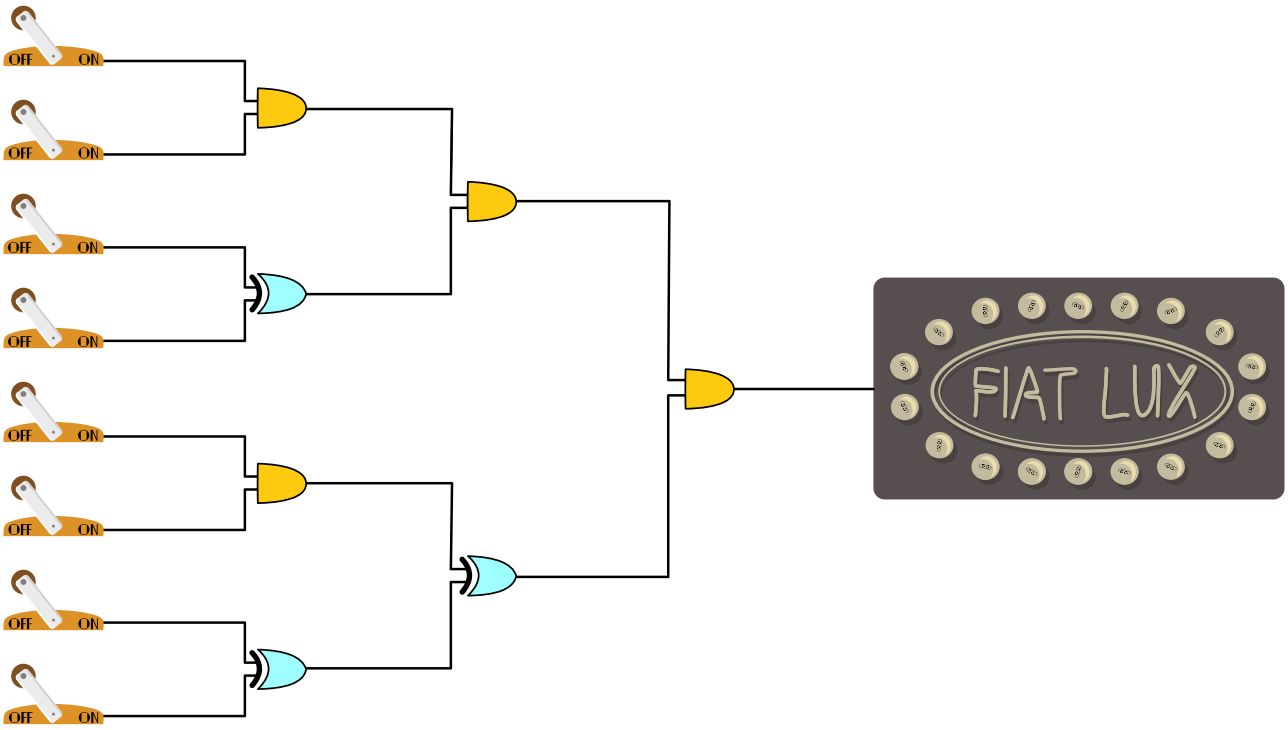
Von den Schaltern aus gehen Drähte zu Bauteilen, von diesen wieder zu anderen Bauteilen und schließlich zu einem Leuchtschild.

Der Ausgangsdraht eines Schalters hat Strom, wenn der Schalter an ist: 

Der Ausgangsdraht eines Bauteils  hat Strom, wenn beide Eingangsdrähte Strom haben.
Der Ausgangsdraht eines Bauteils  hat Strom, wenn genau ein Eingangsdraht Strom hat.

Das Leuchtschild leuchtet, wenn sein Eingangsdraht Strom hat.

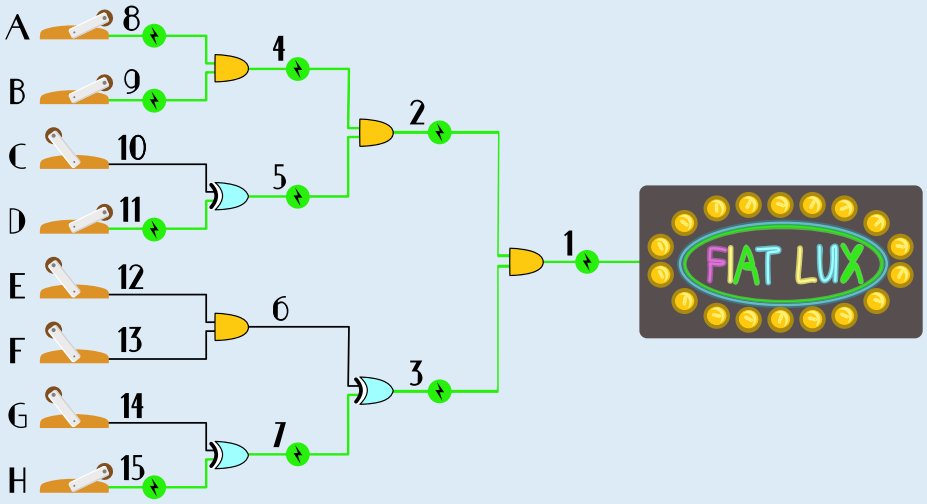
Stelle die Schalter so, dass das Leuchtschild leuchtet.



So ist es richtig:

Wenn man die Schalter so stellt wie im Bild, leuchtet das Schild.

Um alle richtigen Schalterstellungen zu ermitteln, geht man am besten vom Leuchtschild aus rückwärts. Die Drähte wurden nummeriert und die Schalter mit Buchstaben bezeichnet, um über sie sprechen zu können.



Damit das Schild leuchtet, muss Draht 1 Strom haben. Draht 1 hat Strom, wenn die Drähte 2 und 3 beide Strom haben.

Draht 2 hat Strom, wenn die Drähte 4 und 5 beide Strom haben. Das ist der Fall, wenn (a) Schalter A und B beide an sind und (b) genau einer der Schalter C und D an ist. Insgesamt hat Draht 2 Strom, wenn entweder die Schalter A, B und C oder die Schalter A, B und D an sind.

Draht 3 hat Strom, wenn genau einer der Drähte 6 und 7 Strom hat. Draht 6 hat Strom, wenn Schalter E und F an sind. Draht 7 hat Strom, wenn entweder Schalter G oder Schalter H an sind – und keinen Strom, wenn G und H gleichzeitig an oder gleichzeitig aus sind. Insgesamt hat Draht 3 Strom, wenn die Schalter in einer dieser Kombinationen an sind: E, F oder E, F, G, H (dann hat nur Draht 6 Strom) oder G oder H (dann hat nur Draht 7 Strom). Wenn nur einer von G oder H an ist (und nur Draht 7 Strom hat), kann zusätzlich auch noch einer von E oder F an sein, ohne dass Draht 6 Strom bekommt: E, G oder F, G oder E, H oder F, H. Für Draht 3 gibt es also insgesamt acht Möglichkeiten.

Die beiden Möglichkeiten für Draht 2 lassen sich mit den acht Möglichkeiten für Draht 3 beliebig kombinieren. Insgesamt gibt es also 16 Möglichkeiten, die Schalter so zu stellen, dass das Schild leuchtet.

Das ist informatik!

Die Drähte im Spiel „Fiat Lux“ haben entweder Strom oder nicht; man könnte sagen, sie sind entweder an oder aus, wie die Schalter. Es gibt also für Drähte und Schalter jeweils nur zwei Zustände.
Die Unterscheidung zwischen zwei Zuständen ist die kleinstmögliche Informationseinheit. In der Informatik wird sie als Bit bezeichnet. Mit Bits kann man rechnen, ähnlich wie mit Zahlen: Die Zustände nennt man dann meist 0 und 1 (statt „aus“ und „an“), und man verwendet Bit-Operationen, wie zum Beispiel AND (deutsch: „und“) oder XOR („exklusives oder“). Diese beiden Operationen „verrechnen“ zwei Bits b1 und b2, und zwar so:

b1 AND b2 gleich 1 genau dann, wenn b1 und b2 gleich 1 sind
b1 XOR b2 gleich 1 genau dann, wenn genau eines der Bits b1 und b1 gleich 1 ist.

Das Bauteil  in dieser Biberaufgabe realisiert also das AND, das Bauteil  das XOR.

Bits und auch Bit-Operationen können auch in Wirklichkeit mit einfachen und günstigen Bauteilen realisiert werden; insbesondere gilt das für die kombinierte Operation NAND („nicht und“). Interessant ist, dass alle Berechnungen, die mit Bits überhaupt möglich sind, alleine mit Schaltungen aus NANDs realisiert werden können. Das ist ein wichtiger Grund für die Bedeutung von Computern: dass sie aus wenigen einfachen, billigen und heutzutage mikroskopisch kleinen Elementen gebaut werden können.

Filmabend

Ein paar Freunde möchten einen Film miteinander anschauen. Zur Auswahl stehen sieben Filme. Um eine Entscheidung zu fällen, bewertet jede Person jeden Film, und zwar mit

gut , mittel oder schlecht .

Das Ergebnis siehst du unten. Leider gibt es keinen Favoriten für den Filmabend.
Ein Film ist ein „Favorit“, wenn jede Person diesem Film die eigene beste Bewertung gegeben hat. Film 1 zum Beispiel ist kein Favorit, weil Niklaus seine beste Bewertung einem anderen Film gegeben hat, nämlich Film 4.

Ada möchte nun so wenige Freunde wie möglich überzeugen, ihre Bewertung zu ändern, damit es doch einen Favoriten gibt. Aber welche?

Hilf Ada und ändere so wenige Bewertungen wie möglich, so dass es einen Favoriten gibt.

	1	2	3	4	5	6	7
Ada	😊	😊	😊	😊	😊	😊	😊
Nancy	😐	😊	😊	😐	😐	😊	😊
Niklaus	☹	☹	☹	😐	☹	☹	☹
Grace	☹	😐	😐	😐	☹	😐	☹
Edsger	😊	😐	☹	☹	😐	😊	😊
Rozsa	😐	☹	😐	☹	😊	😐	😐

So ist es richtig:

Film	Freunde, die andere Filme besser bewerten
1	4: Nancy, Niklaus, Grace und Rozsa
2	3: Niklaus, Edsger und Rozsa
3	3: Niklaus, Edsger und Rozsa
4	3: Nancy, Edsger und Rozsa
5	4: Nancy, Niklaus, Grace und Edsger
6	2: Niklaus und Rozsa
7	3: Niklaus, Grace und Rozsa

Zu Beginn gibt es keinen Favoriten. Für jeden Film findet Ada Freunde, die andere Filme besser bewerten.

Bei Film 6 gibt es nur zwei Freunde, die andere Filme besser bewerten. Bei allen anderen Filmen sind es mehr als zwei. Wenn nur ein Freund eine Bewertung ändert, wird es danach immer noch keinen Favoriten geben. Also muss Ada mindestens zwei Freunde überzeugen, ihre Bewertung zu ändern: Wenn Niklaus und Rozsa je eine Bewertung so verändern, dass Film 6 ihre beste Bewertung erhält, wird Film 6 ein Favorit.

Welche Bewertung können Niklaus und Rozsa jeweils ändern, damit Film 6 ihre beste Bewertung erhält? Die beiden haben jeweils drei Möglichkeiten:

- Niklaus kann seine Bewertung von Film 6 verbessern (zu 😐 oder 🙂) oder seine Bewertung von Film 4 verschlechtern (zu ☹). In allen drei Fällen erhält Film 6 danach seine beste Bewertung.
- Rozsa kann ihre Bewertung von Film 6 verbessern (zu 🙂) oder ihre Bewertung von Film 5 verschlechtern (zu 😐 oder ☹). In allen drei Fällen erhält Film 6 danach ihre beste Bewertung.

Diese jeweils drei Möglichkeiten können beliebig miteinander kombiniert werden. Insgesamt gibt es also 3 x 3 = 9 Möglichkeiten, nur zwei Bewertungen so zu ändern, dass es einen Favoriten gibt.

Das ist Informatik!

Wie kann Ada vorgehen, um ihr Problem zu lösen? Eine Idee besteht darin, für jeden Film und für jede Person einzeln zu prüfen, ob diese Person andere Filme besser bewertet hat oder nicht. Dabei entsteht eine Tabelle wie oben. Diese Tabelle hilft herauszufinden, welche Personen ihre Bewertungen ändern müssen, damit man tatsächlich mit der kleinstmöglichen Anzahl von Veränderungen zu einem Favoriten kommt. Ada kann tatsächlich diesen Algorithmus verwenden, um ihr Problem effektiv zu lösen. Ist dieser Algorithmus jedoch auch effizient? Könnte Ada noch schneller sein?

Wir bezeichnen im Folgenden die Anzahl Filme mit M und die Anzahl Freunde mit F. Ada muss alle $M * F$ Einträge einzeln betrachten, und für jeden Eintrag muss sie alle anderen M-1 Bewertungen derselben Person berücksichtigen. Insgesamt muss Ada $M * (M - 1) * F$ Bewertungen betrachten.

Aber: Um zu entdecken, ob eine bestimmte Bewertung problematisch ist, muss Ada diese Bewertung nur mit der besten Bewertung vergleichen, die die bewertende Person vergeben hat. Wenn diese Person einen anderen Film besser findet, dann kann der von Ada gerade betrachtete Film gar kein Favorit sein. Anders gesagt, wenn Ada zunächst die besten Gesamtbewertungen für jede einzelne Person herausfindet (indem sie sich einmal alle $M * F$ Bewertungen betrachten), kann sie bei einer zweiten Betrachtung aller $M * F$ Bewertungen feststellen, ob sie schlechter ausfallen als die beste Bewertung der jeweiligen Person.

Dieser alternative Algorithmus mit einer gezielten Vorberechnung der besten Bewertungen führt dazu, dass Anna sich $2 * M * F$ Bewertungen betrachtet. Bei $M = 7$ und $F = 6$ wie in dieser Biberaufgabe sind das 84 Tabellenzugriffe, während der erste Algorithmus 252 Tabellenzugriffe erfordert. Der zweite Algorithmus löst Adas Problem ebenfalls korrekt, ist aber effizienter als der erste Algorithmus.

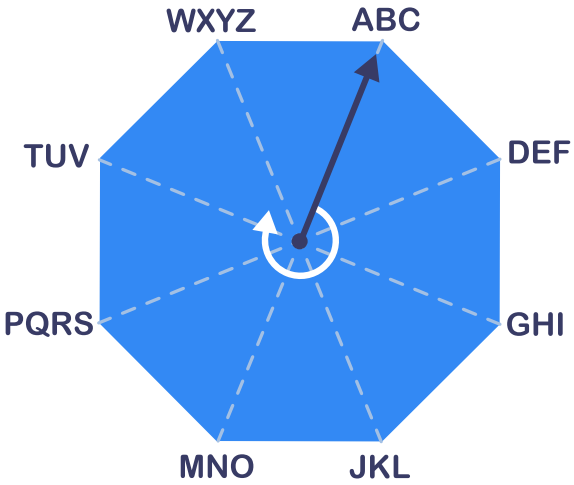
Eine der wichtigsten Aufgaben in der Informatik besteht darin, Probleme nicht nur korrekt, sondern auch so effizient wie möglich zu lösen. Mit schnelleren Computern werden Lösungen schneller berechnet. Sind aber keine effizienten Algorithmen bekannt, um ein Problem zu lösen, werden auch schnellere Computer an ihre Grenzen kommen.

Geheimes Achteck

Mit dieser achteckigen Scheibe werden Worte verschlüsselt. Der Zeiger auf der Scheibe kann auf acht verschiedenen Positionen stehen. An jeder Position ist eine Folge von Buchstaben.

Am Anfang steht der Zeiger immer bei der Folge ABC. Dann wird jeder Buchstabe des Wortes einzeln verschlüsselt, und zwar mit zwei Ziffern:

- Die erste Ziffer gibt an, um wie viele Positionen der Zeiger im Uhrzeigersinn weiter gedreht wird, damit er bei der Folge mit diesem Buchstaben steht.
- Die zweite Ziffer gibt an, der wievielte Buchstabe in der Folge verschlüsselt wird.



Ein Beispiel: Das Wort PAAR wird so verschlüsselt:
51 31 81 53

Welches Wort wird so verschlüsselt: 22 61 62 74 ?

A) HANS **B)** HAUS **C)** HALLO **D)** HALS **E)** HAUT

Antwort B ist richtig:

Bei der Verschlüsselung entsteht 22, wenn der Zeiger von ABC um zwei Positionen zu GHI weiter gedreht wird (erste Ziffer 2), und dass der zweite Buchstabe in dieser Folge verschlüsselt wird (zweite Ziffer 2), also: H

61 entsteht, wenn der Zeiger nun von GHI zu ABC gedreht wird (erste Ziffer 6) und davon der erste Buchstabe verschlüsselt wird (zweite Ziffer 1): A

62 entsteht, wenn der Zeiger nun von ABC zu TUV gedreht wird (erste Ziffer 6) und davon der zweite Buchstabe verschlüsselt wird (zweite Ziffer 2): U

74 entsteht, wenn der Zeiger nun von TUV zu PQRS gedreht wird (erste Ziffer 7) und davon der vierte Buchstabe verschlüsselt wird (zweite Ziffer 4): S

Das Wort HAUS wird also so verschlüsselt: 22 61 62 74

Man kann die richtige Antwort auch schneller bestimmen: Zunächst kann man Antwort C (HALLO) ausschließen, da nur vier Buchstaben verschlüsselt wurden. Nun kann man die Verschlüsselung des letzten Buchstabens betrachten: (a) Weil sie eine 4 als zweite Ziffer hat, kann es sich nur um S oder Z handeln; nur S kommt als letzter Buchstabe vor, nämlich in den Antworten A, B und D. (b) Weil sie eine 7 als erste Ziffer hat, muss der vorletzte Buchstabe von der Position sieben Drehungen gegen den Uhrzeigersinn stammen, also aus der Folge TUV. Von den Antworten A, B und D gilt das nur für Antwort B: HAUS

Das ist Informatik!

Seit Tausenden von Jahren versuchen Menschen, Informationen für die Übermittlung an andere so zu verschlüsseln, dass nur die beabsichtigten Empfänger sie entschlüsseln können. Was mit Pergamentstreifen anfing, die um Stäbe (*Skytale*) gewickelt wurden, entwickelte sich über Substitutionsverfahren wie die Caesar-Verschlüsselung (monoalphabetische Substitution) und die *Vigenère-Verschlüsselung* (polyalphabetische Substitution) zu modernen Verfahren der Public-Key-Kryptografie, wie etwa dem *RSA-Verfahren.*

Das Verschlüsselungsverfahren aus dieser Biberaufgabe ist ein polyalphabetisches Substitutionsverfahren: Zum einen werden die Zeichen des Klartextes durch andere Zeichen ersetzt (substituiert). Dasselbe Zeichen kann aber, wenn es mehrfach auftritt, durch unterschiedliche Zeichen ersetzt werden. Zum Beispiel wird der Buchstabe A beim ersten Auftreten im Klartext PAAR durch 31, beim zweiten Auftreten aber durch 81 ersetzt.

Das „Geheime Achteck" ist aber nicht sonderlich geheim, denn die damit erzeugten Geheimtexte sind leicht zu entschlüsseln – insbesondere, weil der Schlüssel, nämlich die Anordnung der Buchstaben auf dem Achteck, nicht variiert werden kann. Nun kann man noch argumentieren, dass das Geheime nicht der Schlüssel sondern das Verschlüsselungsverfahren ist. Aber bis heute ist diese Maxime von Auguste Kerckhoffs (1835-1903) allgemein anerkannt: Die Sicherheit eines Verschlüsselungsverfahrens darf nicht auf seiner Geheimhaltung beruhen.

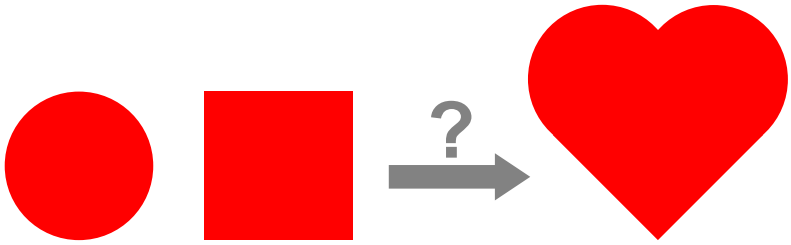
Herzbild

Tina hat zuerst zwei Formen gezeichnet: einen Kreis und ein Quadrat.
Daraus hat sie ein Herz gemacht. Sie hat dabei nur diese Umwandlungen benutzt:

- drehe: eine Form beliebig drehen
- verschiebe: eine Form beliebig verschieben
- verdopple: eine Form an gleicher Stelle verdoppeln

Wie hat Tina das Herz gemacht?

A verdopple Kreis, drehe Quadrat, verschiebe Kreis, verschiebe Kreis
B verdopple Quadrat, drehe Quadrat, verschiebe Quadrat, verschiebe Kreis
C verdopple Kreis, drehe Kreis, verschiebe Kreis, verschiebe Quadrat
D verschiebe Kreis, verschiebe Kreis, verdopple Kreis, verschiebe Quadrat

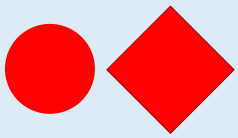
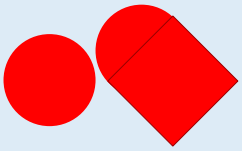
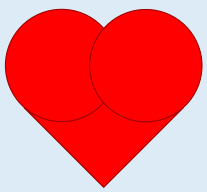


Antwort A ist richtig:

Wenn man sich das Herz genau anschaut, stellt man fest, dass es aus zwei Kreisen und einem um 1/8 (45 Grad) gedrehten Quadrat besteht. Es braucht in den Umwandlungen also ein „verdopple Kreis“, damit man zwei Kreise hat, und ein „drehe Quadrat“, damit das Quadrat um 1/8 gedreht werden kann. Damit fallen die Antworten B, C und D weg, denn:

- In der Antwort B wird ein Quadrat verdoppelt und kein Kreis.
- In der Antwort C wird ein Kreis gedreht, aber nicht das Quadrat.
- In der Antwort D wird gar keine Form gedreht, also insbesondere nicht das Quadrat.

Mit den Umwandlungen aus Antwort A kann Tina das Herz machen:

Aus	wird durch *verdopple* Kreis	wird durch *drehe* Quadrat	wird durch *verschiebe* Kreis	wird durch *verschiebe* Kreis
				

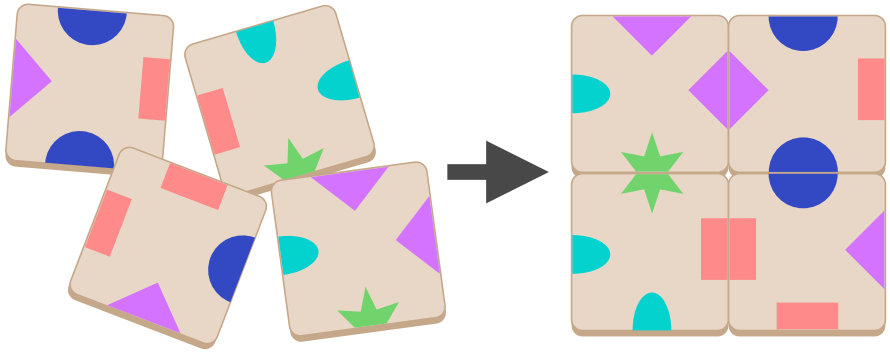
Das ist Informatik!

Wir kennen Tina nicht näher und wissen nicht, ob sie vielleicht ein Roboter sein könnte – zum Beispiel der aus der Biberaufgabe „Roboter Tina“ vom Informatik-Biber 2022. Auf jeden Fall hat sie das Herzbild so erstellt, wie es auch ein Computerprogramm machen würde: Für ihr Kunstwerk hatte sie einfache Bilder (Kreis und Quadrat) und einige Operationen zur Verfügung. Sie hat nacheinander einige Operationen ausgeführt und sie dabei auf die einfachen Bilder angewandt. So entstand ein aus einfachen Bildern zusammengesetztes neues Bild, nämlich das Herz.

Auch in Computerprogrammen gibt es einfache Dinge wie Zahlen oder einzelne Buchstaben (und andere Schriftzeichen) und daraus zusammengesetzte Dinge, wie mehrere Zeichen hintereinander (auch *Zeichenketten* genannt), Listen mit Zahlen darin und vieles mehr. Auf diese Dinge – in der Informatik oft *Objekt* genannt – können Operationen angewandt werden: Für Zahlen gibt es die üblichen Rechenoperationen, eine Zeichenkette kann man in mehrere neue aufteilen oder mehrere Zeichenketten zu einer zusammenfügen, und so weiter. Letztlich besteht ein Computerprogramm aus einer Folge von Anweisungen, und die meisten Anweisungen sind Anwendungen von Operationen auf Objekte.

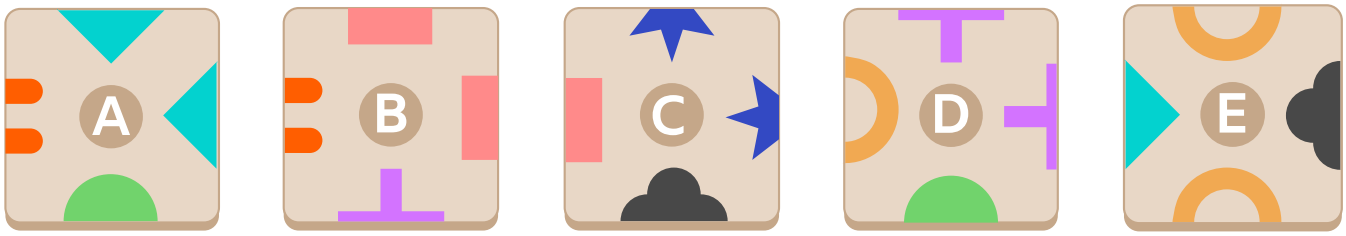
Kartenquadrat

Du sollst mit vier Karten ein Quadrat legen.
Dabei müssen je zwei sich berührende Ränder das gleiche Symbol zeigen.
Mit diesen vier Karten kannst du ein solches Quadrat legen:



Mit vier der fünf folgenden Karten kannst du ebenfalls ein solches Quadrat legen.

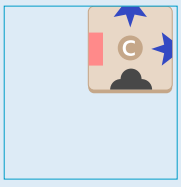
Welche Karte kannst Du dabei NICHT verwenden?



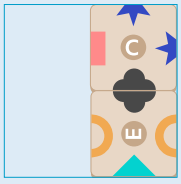
Antwort C ist richtig:

Es gibt $\binom{5}{4} \cdot 4! \cdot 4^4 = 30720$ verschiedene Möglichkeiten, vier von den fünf Karten auszuwählen. Das sind viel zu viele, um sie alle auszuprobieren und dann jede Auswahl zu prüfen, ob sie sich zu einem Kartenquadrat legen lässt. Es ist also klug, sich die Karten näher anzusehen und dabei insbesondere die Verteilung der Symbole. Diese ist nämlich dafür zuständig, ob und wie sich die Karten aneinander legen lassen.

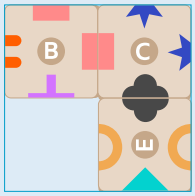
Zunächst fällt auf, dass der blaue halbe Stern nur auf Karte C vorkommt. Da er zudem gleich zweimal und zwar über Eck vorkommt, können nur an den anderen Seiten Karten angelegt werden. In einem Kartenquadrat könnte C also nur in der Ecke liegen.



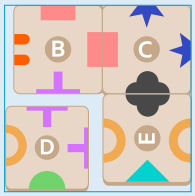
An die Seite mit der halben schwarzen Wolke kann man nun allein Karte E anlegen.



Für das rote Rechteck gibt es ebenfalls nur eine Karte, nämlich B. Diese hat zwar rote Rechtecke an zwei Seiten, aber da es keine weitere Karte mit roten Rechtecken gibt, muss sie so gedreht werden, dass das zweite rote Rechteck nach außen zeigt.



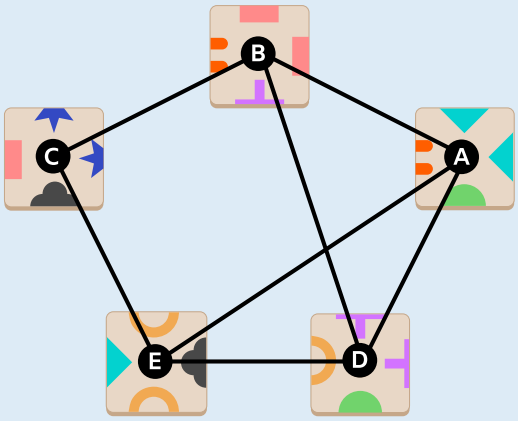
Als letztes braucht es nun eine Karte mit einem halben lila Kreuz und einem hellbraunen Halbkreis. Karte D hat sogar diese Symbole, sie sind jedoch in der falschen Reihenfolge: Der Halbkreis müsste im Uhrzeigersinn direkt nach dem Kreuz kommen, es ist aber umgekehrt.



Da es in keinem Schritt eine andere Karte hätte angelegt werden können, kann also Karte C nicht verwendet werden, um ein Kartenquadrat zu legen.
Aus den anderen Karten lässt sich jedoch ein solches Quadrat legen. Also ist Karte C auch die einzige Karte, die nicht für ein Kartenquadrat verwendet werden kann.

Das ist Informatik!

Wenn zwei Karten das gleiche Symbol zeigen, können sie aneinander gelegt werden. Das kann man mit Hilfe eines *Graphen* darstellen: Die Karten sind die *Knoten,* und wenn zwei Karten das gleiche Symbol zeigen, dann sind sie durch eine *Kante* verbunden. Für die fünf Karten dieser Biberaufgabe sieht der Graph so aus:

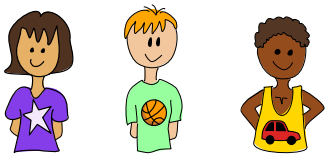


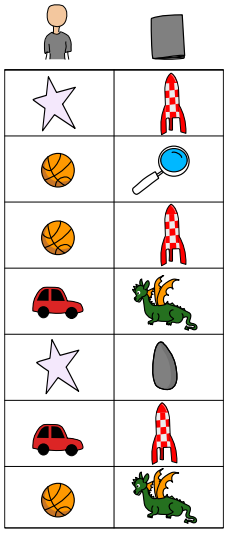
Ein Rundweg über die Kanten des Graphen, an dem vier Knoten (also Karten) beteiligt sind, entspricht einem Kartenquadrat, in dem immer zwei aneinander liegende Karten das gleiche Symbol zeigen. Das ist aber nicht zwingend ein „echtes“ Kartenquadrat im Sinne dieser Biberaufgabe: Ein Rundweg ist nämlich auch dann möglich, wenn die sich im Quadrat berührenden Ränder *nicht* zueinander passen.
Andersherum gilt aber: Für ein echtes Kartenquadrat gibt es im Graphen einen Rundweg mit vier Karten.

Deshalb kann man sich auf die Rundwege mit vier Karten konzentrieren, wenn man ein echtes Kartenquadrat finden will. Während es viele Tausend Möglichkeiten gibt, aus den fünf Karten ein Quadrat zu legen, gibt es in dem Graphen nur drei Rundwege mit vier Karten: A-E-C-B-A, A-E-D-B-A und B-D-E-C-B. Nur für den Rundweg A-E-D-B-A lassen sich die Karten so drehen, dass auch die Ränder der Karten zueinander passen.

Kinder lieben Bücher

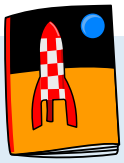
Die Kinder leihen in der Bibliothek Bücher aus. Die Bibliothek schreibt in einer Tabelle auf, wer welches Buch ausgeliehen hat.





Welches Buch haben die Kinder am häufigsten ausgeliehen?

So ist es richtig:

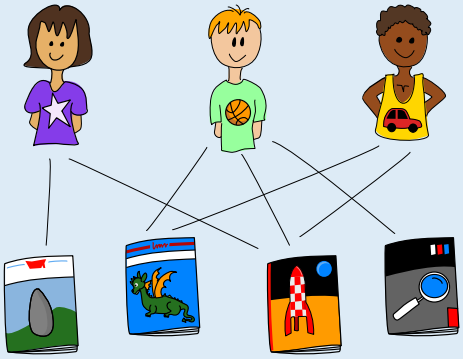


Die Tabelle enthält für jede Ausleihe ein Paar: Ausleihendes Kind und geliehenes Buch. Um die richtige Antwort zu finden, genügt es, bei den geliehenen Büchern nachzuzählen:

- Das Buch mit der Rakete kommt dreimal vor.
- Das Buch mit der Lupe kommt einmal vor.
- Das Buch mit dem Drachen kommt zweimal vor.
- Das Buch mit dem Hinkelstein kommt einmal vor.

Also haben die Kinder das Buch mit der Rakete am häufigsten ausgeliehen.

Das ist Informatik!



Es ist schön, dass die Kinder in dieser Biberaufgabe gerne Bücher lesen! Aber braucht die Bibliothek wirklich eine Tabelle, um die Ausleihen darzustellen? Man könnte auch ein Bild mit Linien zeichnen:
Das wäre für Menschen einfacher (wenn es nicht zu viele Ausleihen sind), aber nicht für Computer. Computer können Bilder mit Linien nicht gut lesen, sind aber sehr gut bei der Arbeit mit Tabellen. Wenn wir wollen, dass Computer uns helfen – zum Beispiel wenn wir uns merken und wieder nachschauen wollen, welches Kind welches Buch ausgeliehen hat – dann ist es eine gute Idee, die Daten so zu speichern, dass der Computer sie gut verarbeiten kann: zum Beispiel in Tabellen.

Tabellen stellen Beziehungen (bzw. Relationen, in der Sprache der Mathematik) dar. In der Bibliothek kann es eine Kundentabelle geben, in der die Ausleihenden mit ihren persönlichen Daten wie Name und Anschrift in Beziehung gesetzt werden. Außerdem ist eine Büchertabelle sinnvoll, in dem für jedes Buch u.a. Titel und Autorin/Autor enthalten sind. Eine Ausleihtabelle wie in dieser Biberaufgabe beschreibt dann eine Relation zwischen Kunden und Büchern. Dass diese Relation in einer eigenen Tabelle gespeichert wird, erlaubt es, dass man mehrere Bücher ausleihen kann.

Die Informatik hat Systeme entwickelt, die große Datenmengen in Tabellen effizient speichern und verarbeiten können: die *relationalen Datenbanken.* Sie bilden die Grundlage vieler wichtiger Informatik-Anwendungen.

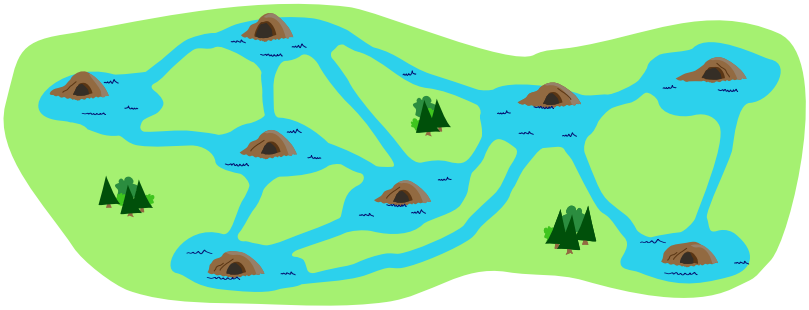
Lilis Nachbarn

Auf der Karte siehst du acht Biberburgen. In jeder Burg wohnt genau ein Biber.
Zwei Biber sind Nachbarn, wenn ein Kanal ihre Burgen verbindet.

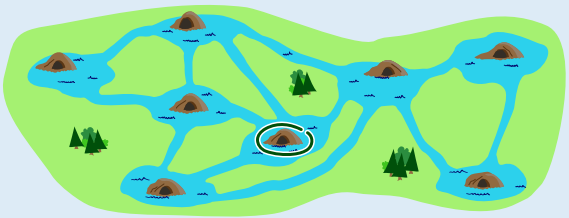
Das ist bekannt:

- Lili, Simon und Peter haben je vier Nachbarn.
- Simon und Peter sind Ninas einzige Nachbarn.

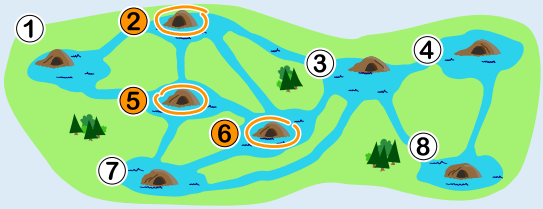
Wo wohnt Lili?



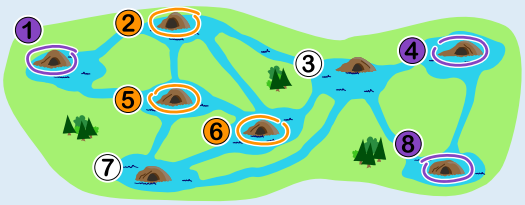
So ist es richtig:



Um die Frage zu beantworten, ist es hilfreich, die Kanäle zwischen den Burgen zu betrachten. Es gibt drei Burgen, von denen vier Kanäle abgehen – deren Bewohner also vier Nachbarn haben: 2, 5 und 6. Lili, Peter und Simon leben in je einer dieser drei Burgen.



Nun müssen wir herausfinden, in welcher der drei Burgen Lili wohnt. Wir wissen, dass von Ninas Burg genau zwei Kanäle abgehen. Also lebt Nina in einer dieser Burgen: 1, 4 oder 8.



Weil Simon und Peter die beiden Nachbarn von Nina sind, können wir folgern, dass

- Nina in der Burg 1 wohnt und
- Simon und Peter in den Burgen 2 und 5 wohnen (oder anders herum).

Also kann Lili nur in Burg 6 wohnen.

Das ist Informatik!

Burgen und Kanäle bilden ein Netzwerk, in dem jeder Kanal zwei Biberburgen miteinander verbindet. Ein solches Netzwerk von Verbindungen oder Beziehungen zwischen Objekten wird in Informatik und Mathematik gerne als *Graph* modelliert. Ein *Graph* besteht aus einer Menge von Knoten und einer Menge von Kanten, die jeweils zwei Knoten miteinander verbinden. Das Burgen-Kanäle-Netzwerk in dieser Biberaufgabe kann also als Graph modelliert werden, und zwar mit den Burgen als Knoten und den Kanälen als Kanten.

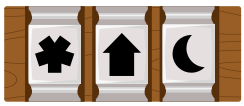
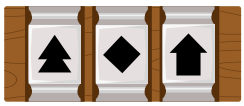
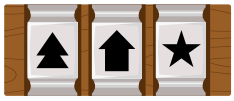
Graphen sind ein sehr vielseitiges Modell, auch weil es interessante Varianten gibt: Die Kanten können eine Richtung haben (dann sind die Kanten geordnete Paare) oder nicht (dann sind sie Mengen mit zwei Elementen). Außerdem können Kanten Werte haben, die auch als Gewichte bezeichnet werden; damit könnte man im Burgen-Kanäle-Netzwerk die Länge der Kanäle modellieren.

Weitere Anwendungsbeispiele für Graphen sind soziale Strukturen, Straßennetze, Computernetze, elektrische Schaltungen, Versorgungsnetze oder chemische Moleküle. Diese Vielseitigkeit bringt es mit sich, dass sich in Bezug auf Graphen viele Fragen ergeben, zum Beispiel: Was ist der kürzeste Weg – über möglichst wenige Kanten, oder über Kanten mit möglichst geringer Gewichte-Summe – zwischen zwei Knoten? Gibt es einen Rundweg über alle Knoten? Die Informatik kann viele „Graphen-Fragen“ (wie etwa die erste Beispielfrage) mit effizienten Algorithmen gut beantworten, während für einige andere Fragen (wie die zweite Beispielfrage) keine effizienten Algorithmen bekannt sind.

Marias Kiste

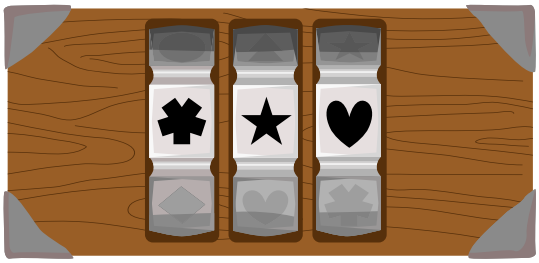
Maria findet eine geheimnisvolle Kiste. Ob ein Schatz darin versteckt ist? Leider ist die Kiste verschlossen. Um sie zu öffnen, muss Maria den „Schlüssel" herausfinden: die richtige Kombination aus drei Symbolen.

Zum Glück findet sie neben der Kiste auch diese Hinweise zu einigen falschen Kombinationen:

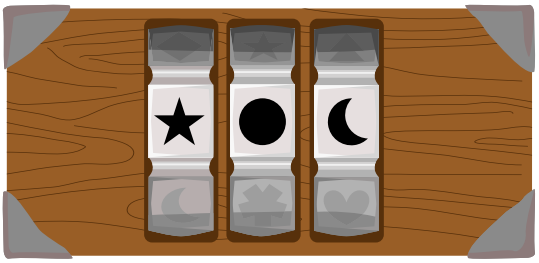
1		Eines der Symbole ist Teil des Schlüssels und steht an der richtigen Position.
2		Keines der Symbole ist Teil des Schlüssels.
3		Zwei Symbole sind Teil des Schlüssels. Beide stehen aber an der falschen Position.
4		Ein Symbol ist Teil des Schlüssels. Es steht aber an der falschen Position.
5		Ein Symbol ist Teil des Schlüssels. Es steht aber an der falschen Position.

Eine dieser Kombinationen ist der Schlüssel für Marias Kiste. Welche?

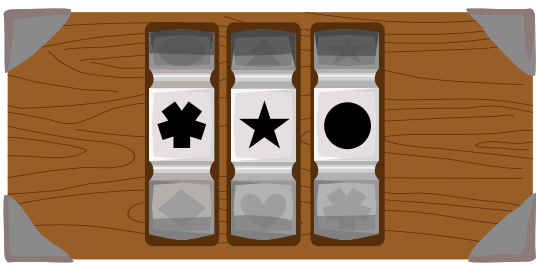
A)



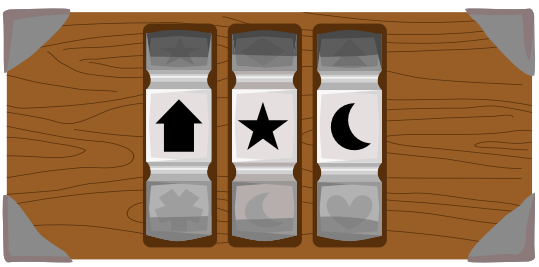
B)



C)



D)



Antwort B ist richtig:

Können wir den Hinweisen entnehmen, welche Symbole Teil des Schlüssels sind und an welchen Positionen sie stehen?

Nach Hinweis 2 sind diese Symbole nicht Teil des Schlüssels: Tanne, Diamant und Haus.
Hinweis 5 gibt an, dass ein Symbol Teil des Schlüssels ist, sich aber an der falschen Position befindet.

Da Tanne und Haus bereits ausgeschlossen sind, wissen wir, dass Stern Teil des Schlüssels ist, aber nicht hinten im Schlüssel steht.

Aus Hinweis 3 kann man schließen, dass der Stern auch nicht in der Mitte steht. Er steht also vorne im Schlüssel:

Das ist nur bei Antwort B der Fall; deshalb kann nur sie richtig sein.
Aus den Hinweisen kann man auch auf die beiden noch fehlenden Symbole und deren Positionen schließen. Hinweis 1 zeigt, dass ein Symbol Teil des Schlüssels ist und an der richtigen Position steht. Das Haus ist ausgeschlossen, und vorne im Schlüssel steht nicht der Asterisk. Also steht der Mond hinten im Schlüssel.

Auch ein Symbol aus Hinweis 4 ist Teil des Schlüssels, steht aber an der falschen Position. ist bereits ausgeschieden, und nur die Mitte ist noch frei. Daher kann das Herz nicht Teil des Schlüssels sein – aber der Kreis, und der steht in der Mitte.
Der Schlüssel sieht insgesamt also so aus:

Der Schlüssel kann auch anders ermittelt werden. Jeder Weg führt aber zum gleichen Ergebnis.

Das ist Informatik!

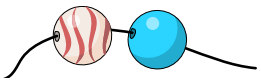
Bei dieser Biberaufgabe konnten wir ganz ähnlich vorgehen wie bei den „Bienenwaben“, um die richtige Antwort zu finden: Wir sind vorrangig den Hinweisen gefolgt, welche die Menge der möglichen richtigen Antworten stark einschränken. Und durch die Hinweise wurden eine Reihe von Bedingungen angegeben, die der Schlüssel für Marias Kiste erfüllen muss, wie beim „Biber-Burger“. Da wir hier das Objekt bestimmen sollten, dass alle Bedingungen erfüllt, haben wir bei dieser Aufgabe wirklich ein *Constraint Satisfaction* Problem gelöst.

Wie kann ein Computerprogramm so ein Problem lösen? Es gibt insgesamt 336 Möglichkeiten, die 8 Symbole auf die 3 Positionen des Schlüssels zu platzieren. Ein Computerprogramm könnte all diese möglichen Antworten auf die Frage nach dem Schlüssel aufzählen und jede prüfen, ob sie alle durch die Hinweise gegebenen Bedingungen erfüllt. Bei nur 336 Möglichkeiten würde diese *vollständige Suche* nach der richtigen Antwort sehr schnell erfolgreich beendet sein. In den meisten realistischen Problemen gibt es aber sehr viel mehr mögliche Antworten, und dann ist eine solche *Brute-Force-Strategie* nicht brauchbar (s. auch die „Bibermeisterschaft“).

Besser geht es mit *Backtracking*: Bei dieser Suchstrategie werden alle möglichen Antworten schrittweise konstruiert und dabei schon für Teilantworten die Bedingungen überprüft. Erfüllt eine Teilantwort eine Bedingung nicht, können alle Antworten, die diese Teilantwort enthalten, bei der Suche ignoriert werden. Ein Beispiel: Wir platzieren den Asterisk auf die erste Position im Schlüssel. Das steht in Widerspruch zu Hinweis 4: Wenn der Asterisk Teil des Schlüssels wäre, stünde er nicht auf der ersten Position. Damit können alle möglichen Antworten mit dem Asterisk an erster Position ignoriert werden; sie können nicht die richtige Antwort sein. Beim Backtracking werden in der Regel sehr viel weniger mögliche Antworten untersucht als bei der vollständigen Suche.

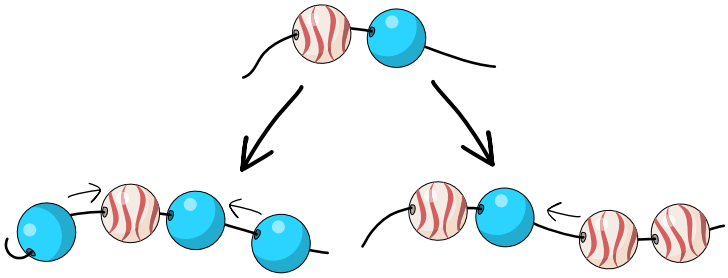
Matrosenketten

Monika macht Matrosenketten mit weiß-roten Wellenperlen und einfarbigen blauen Perlen. Sie beginnt immer mit einer Wellenperle links und einer blauen Perle rechts:



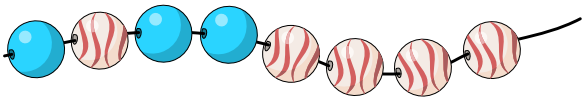
Dann verlängert sie die Matrosenkette mehrmals. Jedesmal fügt sie

- entweder an beiden Enden der Schnur jeweils eine blaue Perle hinzu
- oder zwei Wellenperlen am rechten Ende der Schnur hinzu.

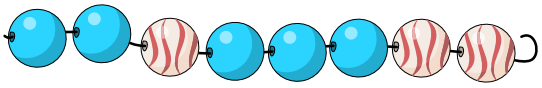


Welche dieser Ketten ist keine von Monikas Matrosenketten?

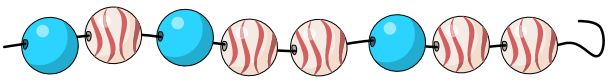
A



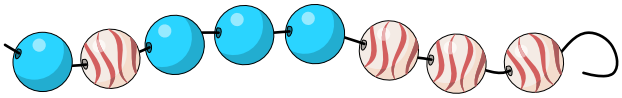
B



C



D



Antwort D ist richtig:

Um festzustellen, ob eine Kette eine von Monikas Matrosenketten ist, kannst du in jeder Kette zuerst die beiden Perlen suchen, mit denen Monika begonnen haben muss. Danach prüfst du, auf welche Weise Monika die Kette verlängert haben kann. Die beiden möglichen Verlängerungsaktionen nennen wir B (zwei **B**laue Perlen links und rechts anfügen) und W (zwei **W**ellenperlen rechts anfügen).

- Bei Kette A hat Monika mit der zweiten und dritten Perle begonnen und mit den Aktionen B, W und W verlängert.
- Bei Kette B hat Monika mit der dritten und vierten Perle begonnen und mit den Aktionen B, B und W verlängert.
- Bei Kette C hat Monika mit der zweiten und dritten Perle begonnen und mit den Aktionen W, B und W verlängert.
- Bei Kette D müssen die zweite und dritte Perle den Anfang bilden. Mit Aktion B kann man die erste und vierte Perle hinzufügen, aber danach gibt es keine Aktionen mehr, um die gesamte Kette zu erhalten.

Wer sich die Verlängerungsaktionen genau ansieht, kann feststellen, dass sowohl die Anzahl der blauen Perlen als auch die der Wellenperlen in einer Matrosenkette immer ungerade sein muss. Kette D hat als einzige eine gerade Anzahl beider Arten von Perlen und kann daher keine von Monikas Matrosenketten sein.

Das ist Informatik!

Wenn man wie Monika in dieser Biberaufgabe Perlenketten herstellt, muss man beachten: Man kann die Perlen nur an den Enden der Kette auffädeln oder wieder entfernen. Insbesondere kann man keine Perle in der Mitte einfügen und auch keine Perle aus der Mitte herausnehmen, ohne zuerst die Perlen von den beiden Enden der Kette her abzufädeln.

Die Informatik kennt eine Struktur zur Speicherung von Daten, die ähnlich funktioniert wie eine Perlenkette, bei der man also Elemente nur an den Enden hinzufügen und entfernen kann, aber nicht in der Mitte. Sie wird *double-ended queue* (auf deutsch etwa: Schlange mit zwei Enden) oder kurz *deque* genannt – und wie „Deck“ ausgesprochen.

Deques können in vielen Fällen gut verwendet werden, etwa um den Zugriffsverlauf eines Web-Browsers oder die in Computersoftware oft eingesetzten Undo-Redo-Listen zu verwalten. Sie können auch nützlich sein, wenn die Gültigkeit mathematischer Ausdrücke geprüft werden soll. Dabei kann die Prüfung, ob es zu jeder öffnenden Klammer auch eine passende schließende Klammer gibt, auf ganz ähnliche Weise erfolgen wie die Prüfung, ob es sich bei einer Perlenkette um eine von Monikas Matrosenketten handelt.

Muttern und Schrauben

Ben steht am Fließband und verarbeitet Bauteile: Muttern  und Schrauben  .

Ben geht strikt nach folgendem Verfahren vor:

- Ben nimmt das nächste Bauteil vom Fließband herunter.
- Wenn Ben eine Mutter vom Fließband genommen hat, legt er sie in den Eimer.
- Wenn Ben eine Schraube vom Fließband genommen hat, nimmt er eine Mutter aus dem Eimer, schraubt sie auf die Schraube und legt das fertige Teil in den Kasten.

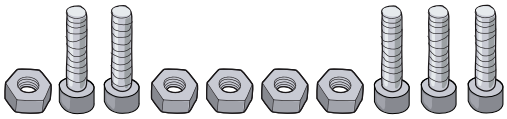
Bei diesem Verfahren können zwei Fehler auftreten:

1. Ben nimmt eine Schraube vom Fließband, aber es ist keine Mutter im Eimer, die er aufschrauben könnte.
2. Ben hat alle Bauteile vom Fließband verarbeitet, aber es sind immer noch Muttern im Eimer.

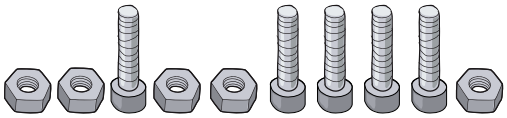
Der Eimer für die Muttern ist ausreichend groß und zu Beginn leer.

Welche Folge von Muttern und Schrauben kann Ben ohne Fehler von links nach rechts verarbeiten?

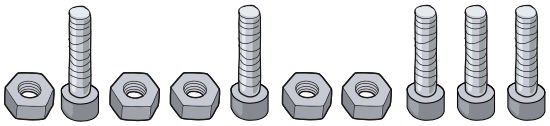
A



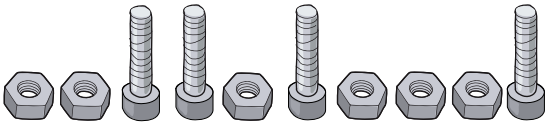
B



C

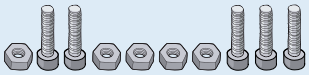
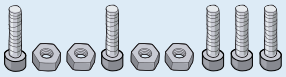
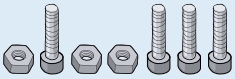
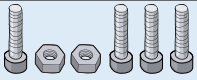


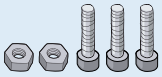
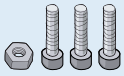
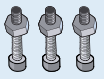
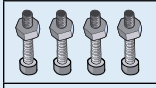
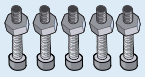
D



Antwort C ist richtig:

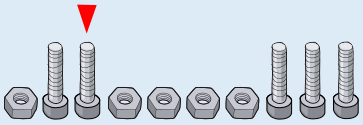
Die Tabelle zeigt für die Bauteilfolge aus Antwort C die Zustände des Kastens für die fertigen Teile, des Eimers für die Muttern und des Fließbands. Ben kann die gesamte Folge ohne Fehler verarbeiten.

Kasten	Eimer	Fließband
leer	leer	
leer		
	leer	
		
		

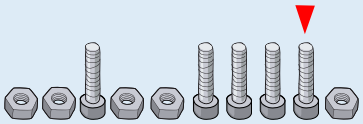
Kasten	Eimer	Fließband
		
		
		
		
		
	**leer**	**leer**

Die Folgen der anderen Antworten kann Ben nicht ohne Fehler verarbeiten:

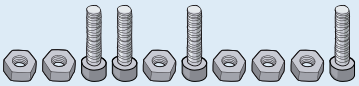
Antwort A:
An der markierten Stelle tritt ein Fehler auf. Dann nimmt Ben eine Schraube, aber es ist keine Mutter im Eimer.



Antwort B:
An der markierten Stelle tritt ein Fehler auf. Ben hat bis dahin 4 Muttern auf 4 Schrauben geschraubt. Der Eimer ist also leer. Nun nimmt er die fünfte Schraube, aber es ist keine Mutter dafür im Eimer.



Antwort D:
Am Ende tritt ein Fehler auf. Nachdem Ben 4 Muttern auf 4 Schrauben geschraubt hat, bleiben 2 Muttern übrig.



Das ist Informatik!
Ben verarbeitet Bauteile, die eins nach dem anderen von dem Fließband geliefert werden. Dabei verwendet er einen großen Eimer zum Zwischenspeichern der Muttern. Eine ähnliche Anordnung wird in der theoretischen Informatik als Modell für Algorithmen verwendet, die eine bestimmte Klasse von Problemen lösen können: *Kellerautomaten.* Ein Kellerautomat verarbeitet Daten (Zahlen oder Zeichen), die er nach und nach als Eingabe erhält. Er besitzt einen einzigen, unendlich großen Speicher: einen Keller. Im Unterschied zum Eimer in der Aufgabe haben die Elemente im Keller eine bestimmte Reihenfolge, und man kann aus einem Keller nur das Element herausnehmen, das man als letztes hineingegeben hat (nach dem Motto „last in first out“, kurz: LIFO).

Ein Kellerautomat kann verwendet werden, um eine *kontextfreie* Sprache zu erkennen. In der Informatik versteht man unter einer Sprache eine Menge von Zeichenketten, die nach bestimmten Regeln geformt worden sind. Ein einfacher Typ von Sprachen sind kontextfreie Sprachen. Ein Beispiel für eine kontextfreie Sprache sind alle wohlgeformten Klammerausdrücke. Bei einem wohlgeformten Klammerausdruck wird jede öffnende Klammer wieder geschlossen. Wohlgeformt sind z.B. ((( ))) und (()()). Nicht wohlgeformt sind dagegen (((() und ())(. Man kann sich die Muttern und Schrauben in der Aufgabe als öffnende und schließende Klammern vorstellen. Dann verarbeitet Ben eine Folge von Bauteilen auf dem Fließband nur dann ohne Fehler, wenn sie einen wohlgeformten Klammerausdruck darstellt. Das Prüfen von Klammerausdrücken ist eine wichtige Aufgabe eines *Compilers,* der Programmtexte in ausführbare Programme übersetzt. Denn in Programmtexten der meisten Programmiersprachen kommen geschachtelte Funktionsaufrufe und arithmetische Ausdrücke mit Klammern vor.

Das LIFO-Prinzip kommt beim Informatik-Biber 2022 auch in der Aufgabe „Bonbon-Spender“ vor.

Roboter Tina

Roboter Tina liefert Post aus. Tina benutzt dazu eine Landkarte, die in Felder eingeteilt ist. Tina bewegt sich der Straße entlang auf ein benachbartes Feld nach links, rechts oder vorne (also nicht diagonal).

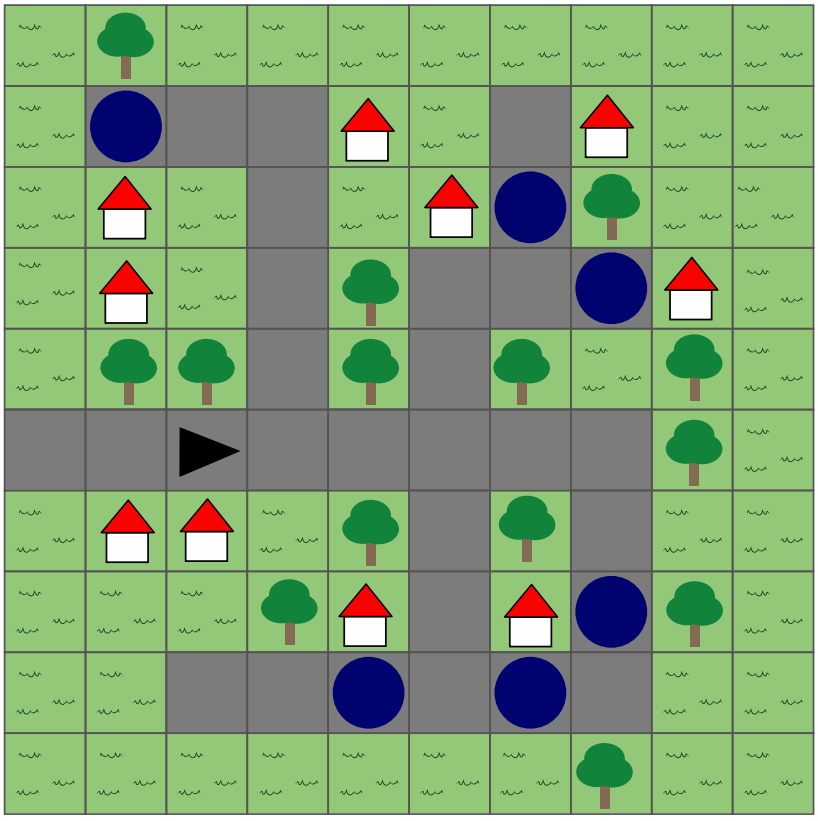
Für die Navigation hat Tina drei Sensoren. Sobald Tina ein Feld betritt (und bevor Tina sich drehen kann), erkennen sie, was sich auf den Feldern links, vor und rechts von Tina befindet:

Straße , Wiese , ein Baum oder ein Haus .

Die Tabelle zeigt, was Tinas Sensoren auf jedem Feld ihres Weges erkannt haben. Tina startet auf dem Feld , in Richtung des Pfeiles.

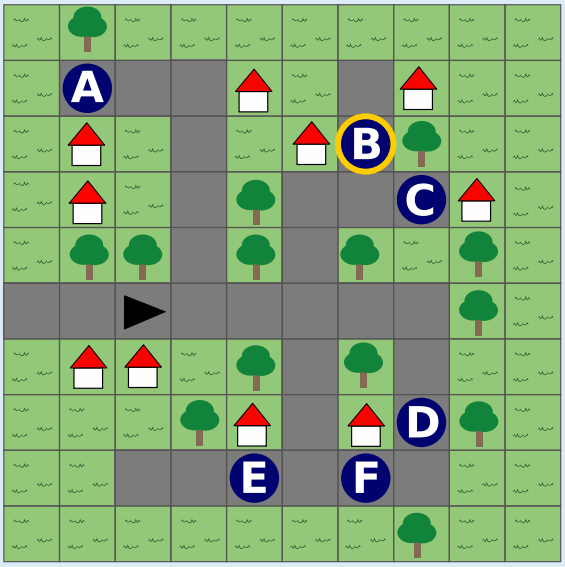
links	vor	rechts

An welchem der dunkelblauen Punkte befindet sich Tina am Ende ihres Weges?



So ist es richtig:

Tina befindet sich am Ende ihres Weges an Punkt B.



Die richtige Antwort lässt sich schnell finden, wenn man schaut, zu welchen Punkten die Sensor-Infos  auf dem letzten Feld von Tinas Weg passen; das sind A, B und D.

Die Sensor-Infos  auf dem vorletzten Feld passen von diesen nur zu B.

Alternativ kann man versuchen, vom Startfeld  aus einen Weg zu gehen, der mit den Sensor-Infos aus der Tabelle immer übereinstimmt. Der Weg zu Punkt B ist der einzige solche Weg.

Das ist Informatik!

In dieser Biberaufgabe begegnen wir Roboter Tina. Roboter sind speziell ausgestattete Computer, die Informationen aus ihrer Umwelt mithilfe von *Sensoren* erfassen, diese Informationen automatisch (d.h. mit einem Programm) verarbeiten und aufgrund des Resultats eine Aktion in ihrer Umwelt mittels sogenannter *Aktoren* selbstständig ausführen.

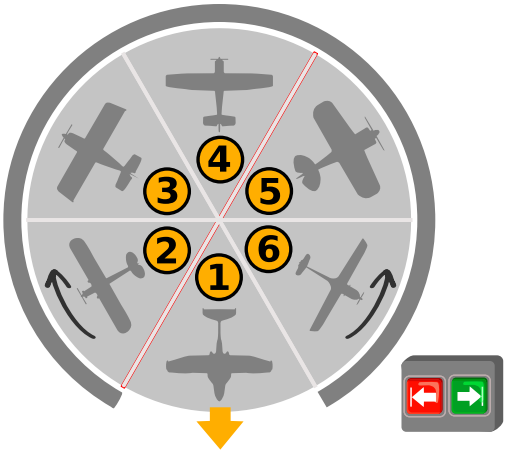
Tinas Sensoren erfassen den Inhalt der Felder links, vorne und rechts. Damit sind die Sensoren schon recht leistungsstark, denn sie können nicht nur Bilder aufnehmen, sondern in diesen Bildern bestimmte Objekte erkennen, z.B. durch eine aufwändige Analyse der Bilddaten. Welche Aktoren Tina hat und welche Aktionen für diese Aktoren aus der Objekterkennung abgeleitet werden, wird in dieser Biberaufgabe nicht beschrieben. Tina hat aber offensichtlich Bewegungsaktoren, z.B. ein Fahrwerk und einen Motor, und ist in der Lage, sich damit in ihrer aktuellen Fahrtrichtung von einem Straßenfeld zum nächsten zu bewegen und ihre Richtung durch Drehen zu ändern.

Selbstfahrende Autos, an denen aktuell auch viele Informatikerinnen und Informatiker arbeiten, sind auch Roboter. Sie sind mit zahlreichen Sensoren ausgestattet, die unter anderem die Geschwindigkeit, die aktuelle Position und den Abstand vom Straßenrand messen sowie Objekte in der Umgebung erfassen. Zur Verarbeitung dieser Informationen werden oft sehr komplexe Algorithmen benötigt, die zum Beispiel Personen erkennen und sie von einem Straßenschild unterscheiden können. Zu den vielen Aktoren selbstfahrender Autos gehören Motor, Lenkung und Bremsen, die selbständig bzw. ohne menschliche Mitwirkung betätigt werden müssen.

Rundhangar

Auf dem Flugplatz von Beavertown parken sechs Flugzeuge in einem Hangar.
Sie stehen auf einer Drehscheibe, in sechs Parkpositionen.

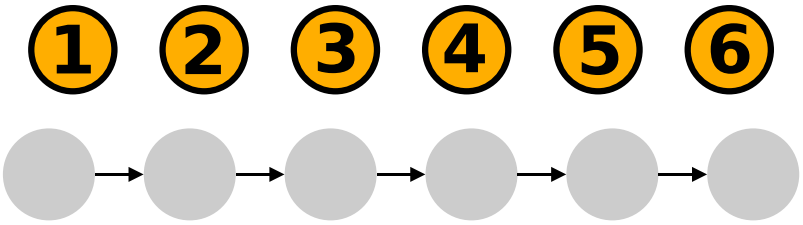
Außen gibt es zwei Pfeiltasten  . Mit einem Tastendruck kann man die Drehscheibe um genau eine Parkposition nach links oder rechts drehen.



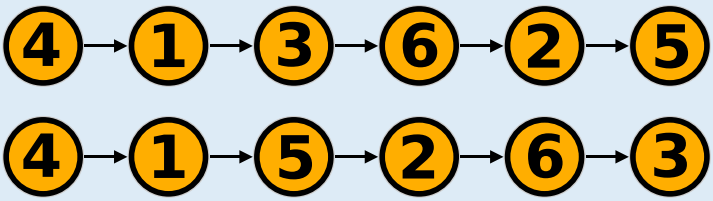
Morgens, wenn die Piloten ihre Flugzeuge abholen, ist die Parkposition 1 immer beim Hangartor, und das Flugzeug darauf kann herausrollen. Im besten Fall müssen die Pfeiltasten dann noch fünfmal gedrückt werden, damit auch alle weiteren Flugzeuge herausrollen können. Wenn beispielsweise die Piloten in der Reihenfolge 1, 6, 5, 4, 3, 2 auf die Parkpositionen zugreifen wollen, genügt es, die Taste  fünfmal zu drücken.

Aber was ist der schlechteste Fall? Bei welcher Reihenfolge müssen die Tasten am häufigsten gedrückt werden? Natürlich drücken die Piloten die Tasten nur so oft wie unbedingt nötig, damit ihr Flugzeug herausrollen kann.

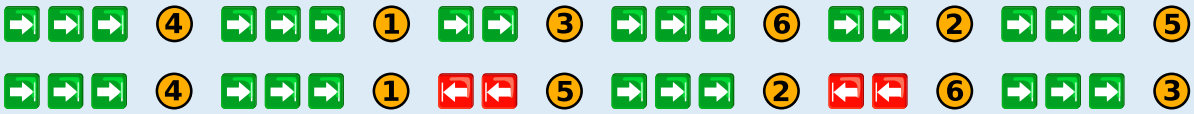
Gib ein Beispiel für eine solche Reihenfolge.



Es gibt zwei richtige Antworten:



Es erfordert maximal häufiges Tastendrücken, wenn man immer auf die Parkposition zugreifen will, auf der noch ein Flugzeug steht und die davon die größte Entfernung von der Position hat, die gerade beim Hangartor ist. Dieser Strategie kann man auf zwei Weisen folgen (➡➡➡ ④ bedeutet, dass die Taste ➡ dreimal gedrückt wird und dann das Flugzeug an Position 4 ausparkt):



In beiden Fällen müssen die Tasten 16-mal gedrückt werden.

Es ist nicht möglich, dass die Tasten häufiger als 16-mal gedrückt werden müssen: Nur bei den ersten beiden Zugriffen können Tasten dreimal gedrückt werden müssen (um zuerst auf Position 4 und dann auf Position 1 zuzugreifen). Bei den nächsten vier Zugriffen können sich höchstens zweimal und dreimal Drücken abwechseln.

Das ist Informatik!

Der Rundhangar hat den Vorteil, dass Flugzeuge darin sehr platzsparend geparkt werden können. Das Abholen kann dafür aber länger dauern als bei einem gewöhnlichen Hangar. Da stellt sich die Frage, unter welchen Bedingungen ein Rundhangar für einen Flughafen vorteilhaft sein kann.

Die Informatik beschäftigt sich mit ähnlichen Fragen. Computer speichern zwar Daten statt Flugzeuge, aber auch bei einer Datenstruktur kann es passieren, dass sie besonders platzsparend ist, aber dafür der Zugriff auf die Daten länger dauert. Wenn die Effizienz einer Informatik-Methode zu bewerten ist, kann es also zu Abwägungen zwischen verschiedenen Effizienzkriterien – wie Speicherverbrauch und Laufzeit – kommen. Auch in der Informatik spricht man dann von einem *Trade-Off.*

Diese Biberaufgabe beschäftigt sich mit der Laufzeit des „Speicherzugriffs“ beim Rundhangar. Bei der Laufzeitanalyse spielt der *worst case,* also der schlechteste Fall, eine besondere Rolle. Man möchte ja auf alles vorbereitet sein, nicht nur auf den besonders angenehmen best case. Deshalb werden Algorithmen grundsätzlich nach ihrem worst case klassifiziert. Der kann aber eine besonders unrealistische Ausnahme darstellen. Ein realistisches Bild der Laufzeit ergibt sich oft erst, wenn man zusätzlich überlegt, wie gut ein Algorithmus mit den wahrscheinlich auftretenden Daten funktioniert, dem *average case.*

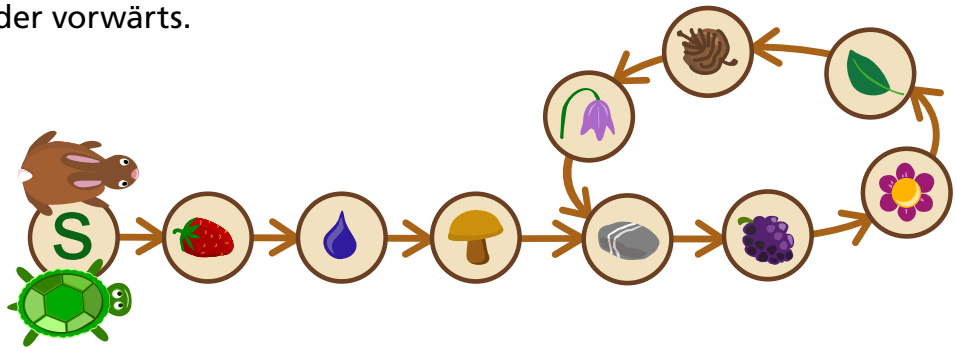
Schildkröte und Hase

Schildkröte und Hase machen einen Wettlauf. Unten siehst du die Laufbahn.

Sie starten gleichzeitig auf dem Feld **S**.
Sie gehen von Feld zu Feld, den Pfeilen entlang. In einer Minute geht …

- … die Schildkröte ein Feld vorwärts und
- … der Hase zwei Felder vorwärts.

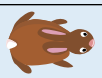
Wo treffen sich Schildkröte und Hase nach dem Start zum ersten Mal wieder?



So ist es richtig:

Schildkröte und Hase treffen sich zum ersten Mal auf dem Feld mit der  wieder. Man kann das leicht mit 2 Fingern nachvollziehen.

Die Tabelle zeigt im Minutentakt, auf welchen Feldern Schildkröte und Hase sich befinden. Nach 6 Minuten treffen sie sich zum ersten Mal wieder:

Minuten nach Start	0	1	2	3	4	5	6	7	8	9	10	11	12	13	…
	S														…
	S														…

Das ist Informatik!

Der Wettlauf zwischen Schildkröte und Hase findet auf einer besonderen Laufbahn statt. Sie besteht aus einzelnen Feldern sowie aus Pfeilen, die jeweils von einem Feld zu höchstens einem nächsten Feld zeigen. Die Laufbahn mündet in einen Kreis aus sechs Feldern, in dem die Läufer endlos laufen können. Schildkröte und Hase können sich in dieser Aufgabe nur begegnen, weil es diesen Kreis gibt.

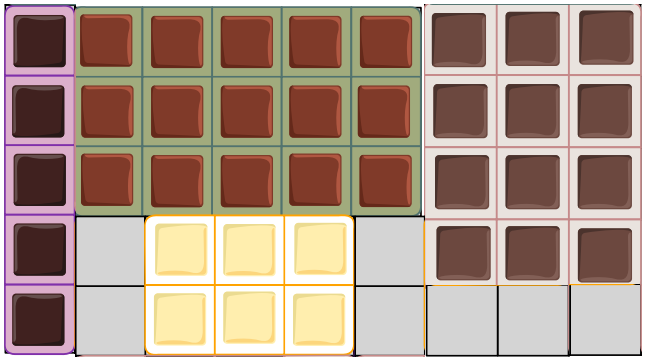
In der Informatik bezeichnet man eine Struktur, wie sie durch die Laufbahn beschrieben ist, als *Liste.* Gibt es in der Liste einen Kreis, wie in dieser Biberaufgabe, bezeichnet man diesen als Zyklus. Es gibt Listen mit einem *Zyklus* (im Bild links) und Listen ohne Zyklus. Hat eine Liste keinen Zyklus, dann besteht die Liste aus einer Kette von Knoten mit einem Endfeld, von dem kein Pfeil ausgeht (im Bild rechts).

Der berühmte Informatiker Robert W. Floyd (1936-2001) hat einen Algorithmus entworfen, der entscheiden kann, ob eine Liste einen Zyklus hat oder nicht. Er lässt genau wie in dieser Biberaufgabe den Hasen und die Schildkröte am Startfeld loslaufen, den Hasen mit der doppelten Geschwindigkeit der Schildkröte. Wenn Schildkröte und Hase gleichzeitig auf einem Feld ankommen, gibt es einen Zyklus. In dem Moment, in dem der Hase keinen weiteren Zug mehr machen kann, hat er das Endfeld oder das Feld davor erreicht, und es ist kein Zyklus vorhanden.

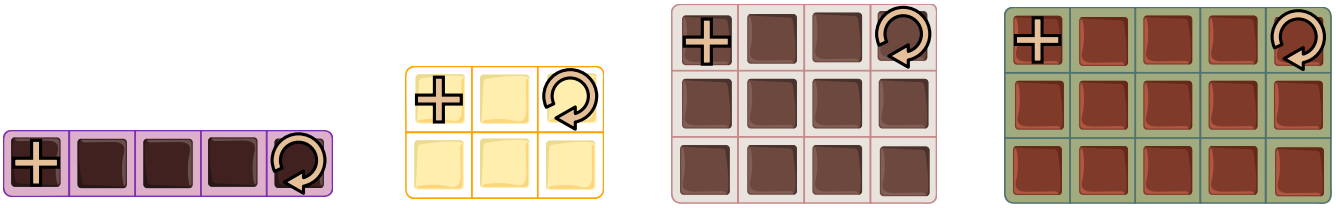
Schokolade packen

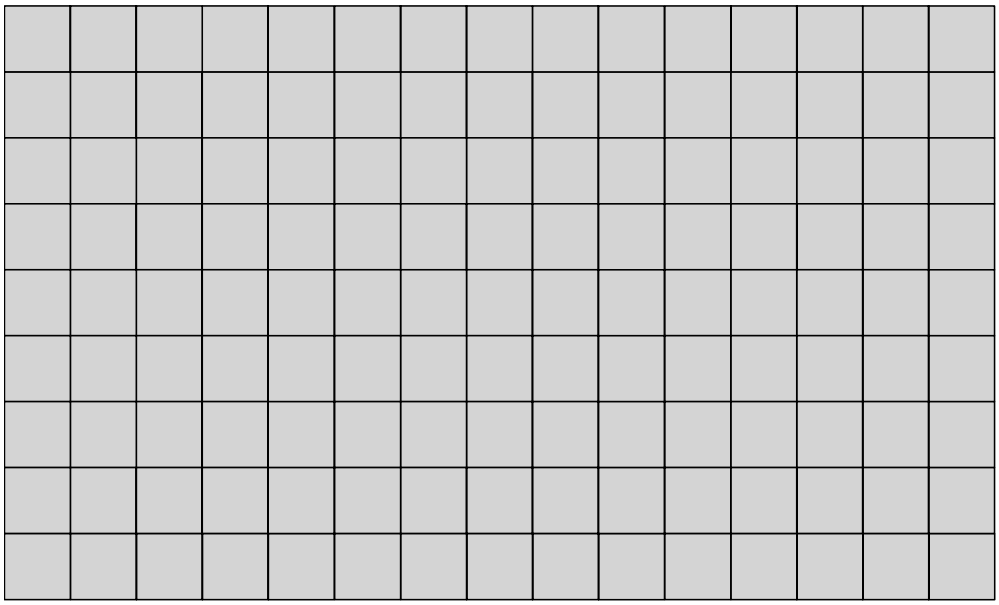
Jeder Kunde einer Schokoladenfabrik soll als Werbung vier Tafeln Schokolade bekommen. Die Tafeln sind unterschiedlich groß, aber ihre Stücke sind alle quadratisch und gleich groß.

Die vier Tafeln müssen nebeneinander in eine Schachtel. In der Schachtel soll so wenig Platz frei bleiben wie möglich. Wenn die Tafeln zum Beispiel so gelegt werden, passen sie in eine Schachtel für 5 x 9 Stücke, mit freiem Platz für 7 Stücke.



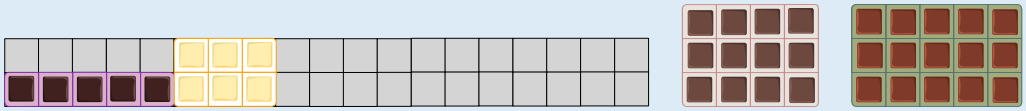
Lege die Tafeln so, dass sie in eine Schachtel mit möglichst wenig freiem Platz passen.



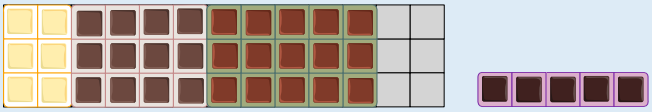


So ist es richtig:

Insgesamt sollen 12 + 15 + 6 + 5 = 38 Schokoladenstücke in die Schachtel. Ein Schachtel, in die 38 einzelne Stücke ohne freien Platz gelegt werden können, müsste entweder die Größe 1 x 38 oder 2 x 19 haben; 2 und 19 sind die einzigen Teiler von 38. Die beiden Tafeln mit den Größen 3 x 4 und 3 x 5 würden in keine dieser Schachteln passen.



Eine Schachtel für 39 Stücke (also mit freiem Platz für genau ein weiteres Stück) hat entweder die Größe 1 x 39 oder 3 x 13. In die Schachtel der Größe 1 x 39 passt nur die Tafel 1 x 5. In die Schachtel der Größe 3 x 13 passen die Tafeln 3 x 5, 3 x 4, 3 x 2, die Tafel 1 x 5 passt aber nicht in den verbleibenden freien Platz der Größe 3 x 2.



Eine Schachtel für 40 Stücke wiederum kann folgende Größen haben: 1 x 40, 2 x 20, 4 x 10, 5 x 8. In die Schachteln der Größe 1 x 40 und 2 x 20 passen nicht alle Tafeln. In die beiden anderen Schachteln passen alle vier Tafeln, z.B. so:

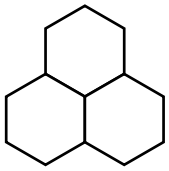
Man kann die Tafeln noch zu einigen anderen Anordnungen legen, die in eine Schachtel für 40 Stücke passen. Platzsparender als mit freiem Platz für 2 Stücke können die vier Tafeln aber nicht gepackt werden.

Das ist Informatik!

In dieser Biberaufgabe sollen Rechtecke so angeordnet werden, dass das umschließende Rechteck die minimale Fläche hat. Dieses Problem ist in der Informatik auch als „rectangle packing“ bekannt, eines von vielen so genannten Verpackungsproblemen. Für wenige Rechtecke lässt sich relativ einfach die optimale Lösung finden (hier die kleinst mögliche Schachtel). Bei größeren Stückzahlen ist es notwendig den Prozess zu automatisieren; es wird also ein Algorithmus benötigt, der als Computerprogramm realisiert werden kann. Leider ist „rectangle packing“, wie viele andere Verpackungsprobleme auch, *NP-vollständig.* Das bedeutet, dass es für das Problem sehr wahrscheinlich keinen effizienten Algorithmus gibt, der optimale Lösungen findet. In der Informatik werden für NP-vollständige Probleme deshalb effiziente Algorithmen entwickelt, die zwar nicht garantiert optimale, aber nachweisbar gute Lösungen finden können.

Unter anderem für Logistikunternehmen sind effiziente Lösungsansätze für Probleme solcher Art von großer Bedeutung, z. B. zum Einlagern in Hochregalen, fürs platzsparende Verpacken von Waren, oder zur Verteilung von Waren auf Container. Außerdem können scheinbar andersartige Probleme als Verpackungsprobleme beschrieben werden. Ein Arbeitsprozess, den *N* Arbeitskräfte in M Stunden bewältigen können, kann zum Beispiel als *N x M* Rechteck dargestellt werden. Mehrere Prozesse kann man also mit möglichst geringem Aufwand an Personen und Zeit bewältigen, wenn man für die entsprechenden Rechtecke das „rectangle packing“ Problem optimal löst.

Sechseck ausmalen

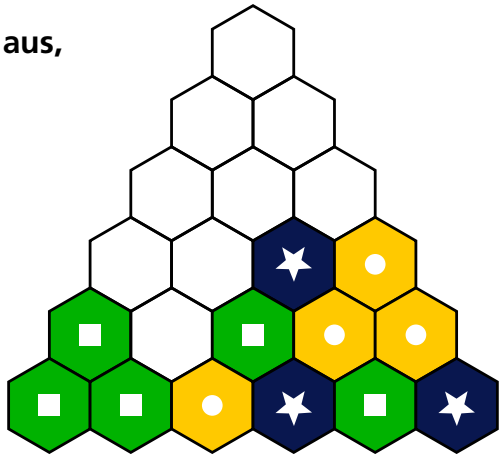


Sami legt weiße Sechsecke aneinander. Dann malt er sie aus, mit drei verschiedenen Farben. Immer wenn drei Sechsecke genau so zusammen liegen (zwei unten und eines oben in der Mitte), müssen sie am Ende …

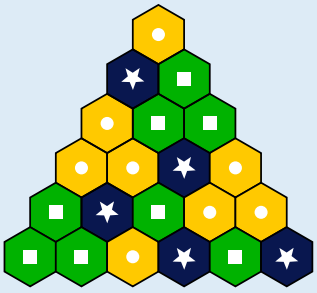
- alle drei die gleiche Farbe **oder**
- alle drei verschiedene Farben haben.

Das gefällt Sami! Sami hat viele Sechsecke aneinander gelegt und schon einige ausgemalt.

Male alle übrigen Sechsecke aus, so wie es Sami gefällt.



So ist es richtig:



Sobald zwei Sechsecke ausgemalt sind, die in der Sechseck-Pyramide nebeneinander liegen, steht die Farbe für das Sechseck darüber fest: Haben beide die gleiche Farbe, bekommt das Sechseck darüber ebenfalls diese Farbe. Haben beide verschiedene Farben, bekommt das Sechseck darüber die dritte Farbe. So muss zum Beispiel das unterste zu Beginn weiße Sechseck die Farbe Blau bekommen.

So kann man die übrigen Sechsecke reihenweise, von unten nach oben, nacheinander ausmalen, so wie es Sami gefällt.

Das ist Informatik!

Wie löst man diese Biberaufgabe? Wenn man ein Sechseck ausmalt, führt man eine *Aktion* aus. Um die richtige Aktion (mit der richtigen Farbe) auszuwählen, muss man sich die Sechsecke darunter anschauen und prüfen, welche *Bedingung* sie erfüllen: Haben sie die gleiche Farbe oder unterschiedliche Farben? Diese Prüfung, mit anschließender Aktion, wird *wiederholt,* nämlich *für jed*es noch weiße Sechseck, das über zwei bereits ausgemalten Sechsecken liegt.

Aktionen, Bedingungen, Wiederholungen: Das sind die Grundbausteine eines jeden Algorithmus, also eines präzise beschriebenen Verfahrens, das auch als Programm für einen Computer realisiert werden kann. Beim Lösen dieser Biberaufgabe hast du also einen Algorithmus erfunden. Das ist eine der wichtigsten Aufgaben von Informatikerinnen und Informatikern: Algorithmen zu erfinden oder bereits erfundene Algorithmen nutzen und in Programme für Computer umzusetzen, um Aufgaben und Probleme mit automatischer Informationsverarbeitung zu lösen.

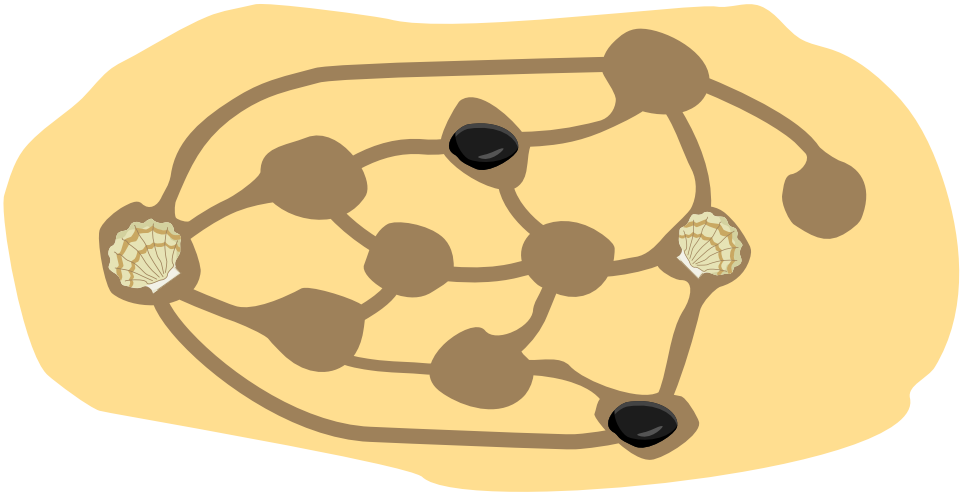
Spiel am Strand

Ann und Bob spielen am Strand. Sie graben einige Mulden und verbinden manche davon mit Furchen. Anns Spielfiguren sind Muscheln . Bobs Spielfiguren sind Steine .

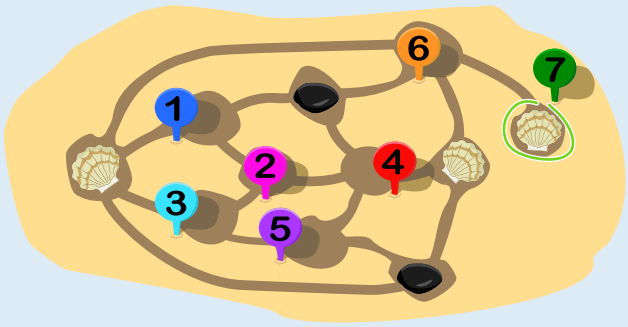
Abwechselnd setzen sie eine ihrer Spielfiguren in eine freie Mulde. Verloren hat, wer als erstes zwei eigene Figuren in zwei direkt verbundene Mulden gesetzt hat.

Unten siehst du den Spielstand nach einigen Zügen. Ann ist an der Reihe.

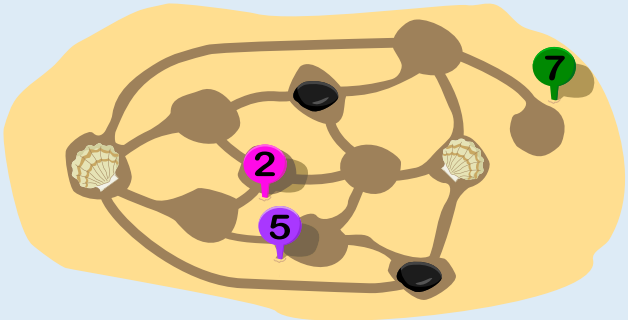
In welche Mulde muss Ann ihre nächste Muschel setzen, um sich den Sieg zu sichern?



So ist es richtig:

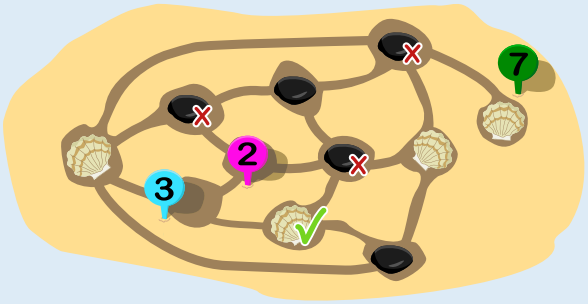


Ann ist an der Reihe. Für sie kommen die Mulden 1, 3, 4 und 6 nicht in Frage, es bleiben also 2, 5 und 7.



Sie sieht, dass für Bob die Mulden 1, 4, 5 und 6 nicht in Frage kommen. Für ihn bleiben also 2, 3 und 7.

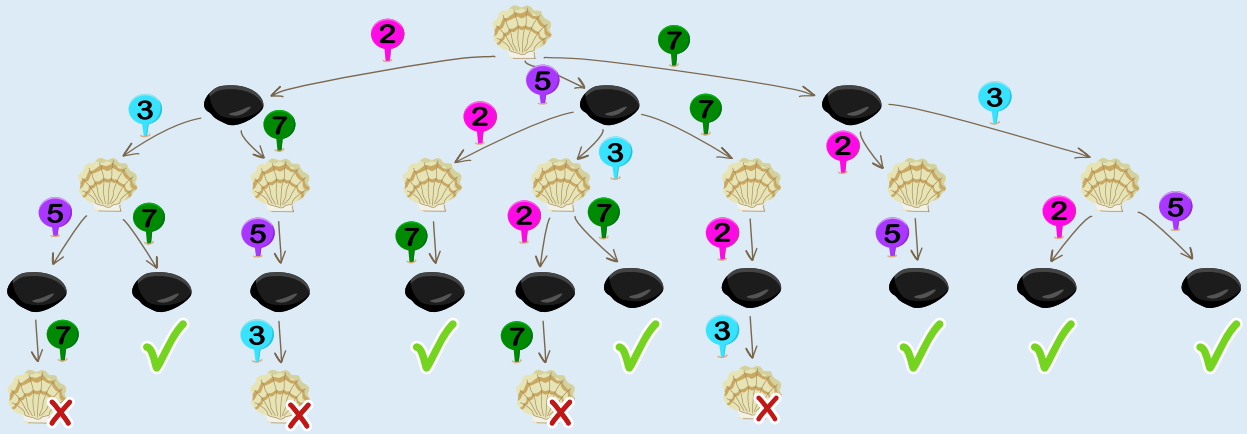
Wenn Ann die 7 spielt, kann Bob entweder 2 oder 3 spielen; in beiden Fällen kann Ann noch die 5 spielen und Bob verliert.



Wenn Ann beim vorgegebenen Spielstand die 2 spielen würde, könnte Bob als nächstes die 7 spielen. Danach müsste Ann die 5 spielen, Bob die 3, und dann hätte Ann verloren.
Würde Ann die 5 spielen, könnte Bob die 7 spielen, Ann müsste die 2 spielen, Bob die 3, und wieder hätte Ann verloren.
Übrigens könnte Bob auch nicht gewinnen, wenn er beim vorgegebenen Spielstand an der Reihe wäre und Ann alles richtig macht.

Das ist Informatik!

Um die möglichen Spielzüge von Ann und Bob systematisch darzustellen, bietet sich ein sogenannter Spielbaum an:



In diesem Spielbaum lässt sich ablesen, mit welchem Zug Ann sich den Sieg sichern kann: Im rechten Ast, der damit beginnt, dass Ann die 7 spielt, sind nur Situationen erreichbar, in denen sie gewinnt. Die Informatik beschäftigt sich mit Algorithmen zur Auswertung solcher Spielbäume, u.a. in der kombinatorischen Spieltheorie. Computer mit ausreichender Rechenleistung können in Spielen wie Schach gegen Menschen antreten und gewinnen. Die Spieltheorie bietet im allgemeinen aber auch für die Psychologie, die Wirtschaftswissenschaften und andere Fächer Modelle für komplexe Systeme, in denen „Spieler“ interagieren, etwa für das Kaufverhalten von Kunden bei Preisänderungen oder für die Routenwahl im Straßenverkehr.

Bei dem Spiel von Ann und Bob in dieser Biberaufgabe handelt es sich um eine Instanz von “COL”. Dieses Spiel für zwei Spieler wurde von Colin Vout eingeführt und spielt im bekannten Buch “On Numbers and Games” (Erstpublikation 1976) des Mathematikers John Horton Conway (1937-2020) eine Rolle. Dahinter steckt ein Färbeproblem, so wie bei der Aufgabe „Ausmalbild“.

Stickmuster

Lana besitzt eine programmierbare Stickmaschine.

Die Maschine kann diese zwei Zeichen sticken: + oder X .
Außerdem kann sie aus diesen Zeichen ein drittes Zeichen zusammensetzen: ✳

Dazu muss der Stoff zwischen X und + (oder umgekehrt)
um ein Zeichen zurückgeschoben werden.
Lana programmiert die Stickmaschine mit diesen drei Tasten:

⊕	Die Stickmaschine stickt + .
⊗	Die Stickmaschine stickt X .
⊖	Der Stoff wird um ein Zeichen zurückgeschoben.

Die Stickmaschine führt ein Programm so oft hintereinander aus, wie Lana will.
So hat Lana dieses Muster

+✳X+✳X+✳X+✳X+✳X+✳X+✳X

mit diesem Programm erstellt:

⊕ ⊕ ⊖ ⊗ ⊗

Mit welchem Programm hat Lana nun dieses Muster erstellt?

XX✳X✳XX✳X✳XX✳X✳XX✳X

A ⊗ ⊗ ⊗ ⊖ ⊕ ⊗ ⊕ ⊖ ⊗ ⊗ ⊗

B ⊗ ⊗ ⊗ ⊖ ⊕ ⊗ ⊗

C ⊗ ⊗ ⊗ ⊖ ⊕ ⊗ ⊕ ⊖ ⊗

D ⊕ ⊖ ⊗ ⊕ ⊖ ⊗ ⊗ ⊕ ⊖ ⊗ ⊗

Antwort C ist richtig:

Das Muster wird durch eine (unbekannte) Anzahl von Ausführungen des gesuchten Programms erstellt. Es enthält also mehrfach hintereinander ein immer gleiches Teil-Muster, das durch einmalige Ausführung des Programms entsteht. Zunächst müssen wir also dieses sich wiederholende Teil-Muster erkennen; das ist dieses hier:

.

Welches Programm erzeugt nun dieses Muster bei einmaliger Ausführung? Wir können zum einen versuchen, aus dem Muster auf das Programm zu schließen. Wir wissen, dass das Zeichen X durch die Taste ⊗ gestickt wird. Das Zeichen ✳ wiederum kann durch zwei verschiedene Tastenfolgen gestickt werden, nämlich ⊗ ← ⊕ oder ⊕ ← ⊗. Das gesuchte Programm hat also die Form ⊗ ⊗ ? ⊗ ?, wobei an der Stelle der Fragezeichen je eine der beiden Tastenfolgen für ✳ stehen kann. Nur das Programm in Antwort C hat diese Form.

Zum anderen können wir für jedes Programm feststellen, welches Muster es bei einmaliger Ausführung erzeugt. Hier ist eine Tabelle; das obige Teil-Muster entsteht durch Programm C.

A	⊗ ⊗ ⊗ ← ⊕ ⊗ ⊕ ← ⊗ ⊗ ⊗	X X ✳ X ✳ X X
B	⊗ ⊗ ⊗ ← ⊕ ⊗ ⊗	X X ✳ X X
C	⊗ ⊗ ⊗ ← ⊕ ⊗ ⊕ ← ⊗	X X ✳ X ✳
D	⊕ ← ⊗ ⊕ ← ⊗ ⊗ ⊕ ← ⊗ ⊗	✳ ✳ X ✳ X

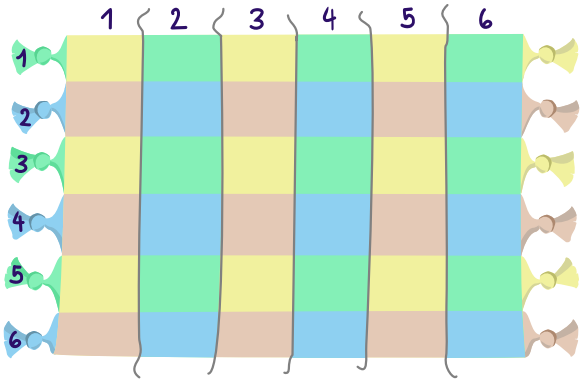
Das ist Informatik!

Die Stickmaschine kennt drei verschiedene Anweisungen. Sie kann eine beliebige, aber endliche Folge dieser Anweisungen ausführen und so ihre Muster erzeugen. Außerdem kann sie – auf Anstoß des Benutzers oder der Benutzerin – die Ausführung der Anweisungsfolge wiederholen. Die Anweisungsfolge wird als *Programm* bezeichnet.

Damit funktioniert die Stickmaschine ganz ähnlich wie ein Computer. Auch Computer kennen eine Reihe von Anweisungen, meist aber deutlich mehr als drei. Auch Computerprogramme bestehen aus Folgen dieser Anweisungen. Allerdings ist es für Menschen recht schwierig, mit Hilfe der Anweisungen, die ein Computer direkt ausführen kann (in der Informatik auch *Maschinenanweisungen* genannt), ein Programm zu schreiben. Deshalb gibt es Programmiersprachen mit anderen Anweisungen, die Menschen besser verstehen können. Damit ein Computer ein Programm ausführen kann, das in einer solchen höheren Programmiersprache geschrieben ist, müssen die Anweisungen der Programmiersprache in Maschinenanweisungen übersetzt werden.

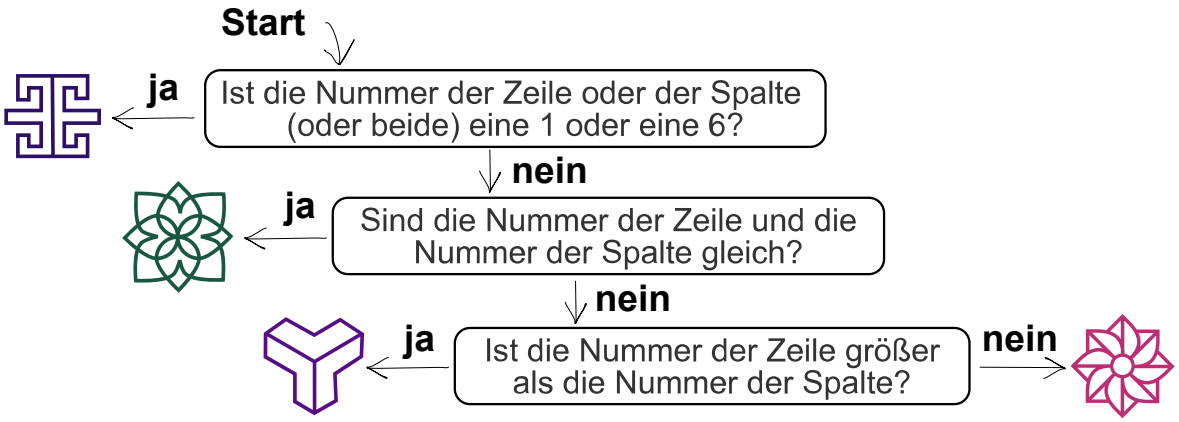
Im Gegensatz zu den Programmen der Stickmaschine gibt es in allen höheren Programmiersprachen für Computer nicht nur einfache Anweisungen und Anweisungsfolgen, sondern insbesondere *bedingte Anweisungen* und *Wiederholungsanweisungen.* Sie sorgen dafür, dass Teile des Programms abhängig von Bedingungen ausgeführt bzw. wiederholt ausgeführt werden. Erst diese besonderen Anweisungen verhelfen Programmiersprachen und damit auch Computern zu ihrer besonderen Leistungsfähigkeit.

Teppichmuster



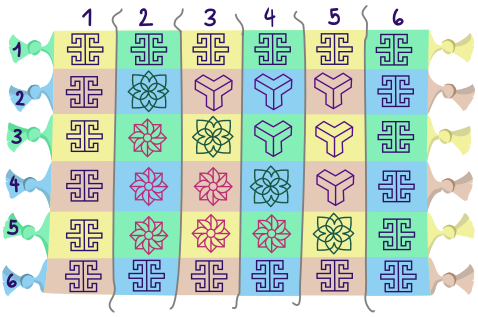
Hale ist eine türkische Künstlerin. Sie gestaltet ein Teppichmuster. Der Teppich hat Felder, in sechs Zeilen und sechs Spalten. Für jedes Feld gibt es die Nummer der Zeile und die der Spalte.

Hales Angestellte sollen in jedes Feld ein Symbol setzen. Hale hat ihnen dazu diese Anleitung gegeben:

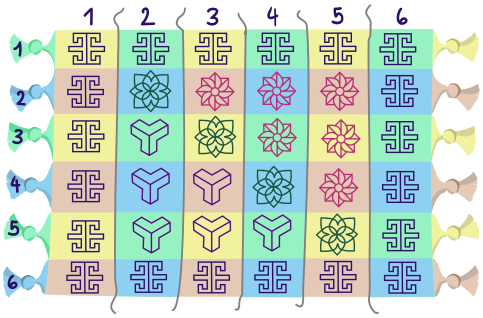


Wie wird der Teppich aussehen?

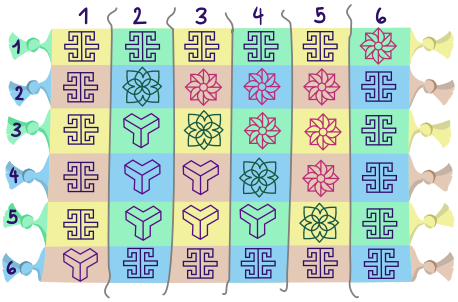
A



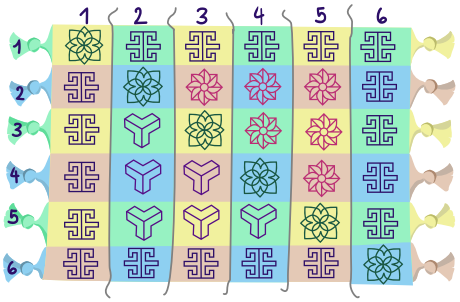
B



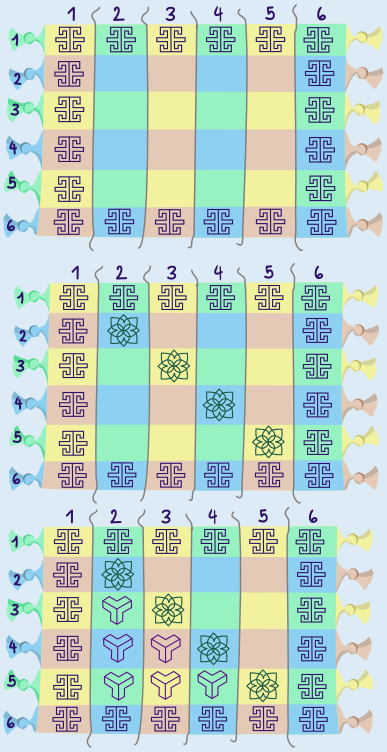
C



D



Antwort B ist richtig:



Die erste Frage der Anleitung wird genau für alle Felder am Rand des Teppiches mit "ja" beantwortet. Denn jedes Randfeld liegt in Zeile 1 oder Zeile 6 oder in Spalte 1 oder Spalte 6. In diese Felder setzen die Angestellten also das Symbol  :

Für alle anderen Felder wird als nächstes die zweite Frage gestellt. Sie wird für alle Felder auf der Diagonalen mit "ja" beantwortet; bei denen sind die Nummern der Zeilen und Spalten gleich. In diese Felder setzen die Angestellten also das Symbol  :

Für die verbleibenden Felder wird die dritte Frage gestellt. Bei den Feldern links unterhalb der Diagonale ist die Nummer der Zeile größer als die Nummer der Spalte. Für sie wird die Frage also mit „ja" beantwortet, und die Angestellten setzen dorthin das Symbol  . Für die Felder rechts oberhalb der Diagonalen wird die Frage mit „nein" beantwortet, und die Angestellten setzen dorthin das Symbol  . Der Teppich sieht also am Ende so aus – wie in Antwort B:

Das ist Informatik!

Die Anleitung, die Hale ihren Angestellten gegeben hat, kann man so zusammenfassen: Bei jedem Teppich-Feld werden nach und nach bis zu drei Bedingungen überprüft. Das Ergebnis der Überprüfung kann jeweils nur „ja" oder „nein" lauten. Die Entscheidung, welches Symbol in das Feld gesetzt wird, ergibt sich eindeutig aus den Ergebnissen der Überprüfungen. Damit würde jede Person, die diese Anleitung verwendet, genau das gleiche Teppichmuster herstellen. Im Prinzip könnte auch eine Maschine das Teppichmuster produzieren, sofern sie die Anleitung lesen und verstehen und Symbole in die Felder des Teppichs setzen kann.

In der Informatik nennt man eine solche eindeutige Anleitung einen *Algorithmus.* Wenn ein Algorithmus in einer Programmiersprache beschrieben ist und diese Beschreibung von einem Computer ausgeführt werden kann, spricht man von einem *Computerprogramm.* In praktisch allen Programmiersprachen gibt es Anweisungen, mit denen sich Bedingungen überprüfen lassen, wobei das Ergebnis der Überprüfung nur „ja" oder „nein" lauten kann. In der Informatik spricht man von *bedingten Anweisungen* und nennt solche Ja-Nein-Bedingungen *boolesche Ausdrücke.* Hales Teppichmuster-Algorithmus könnte also relativ leicht in einer Programmiersprache beschrieben werden.

Verstecke

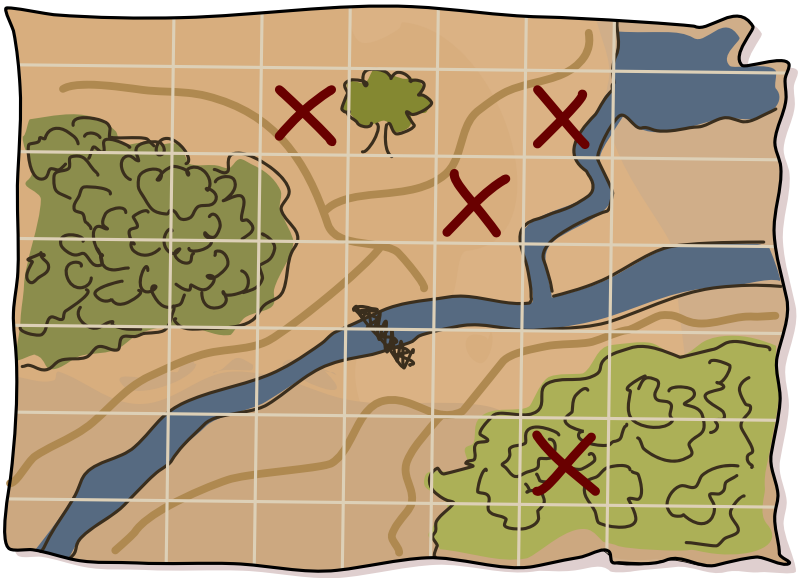
Biber Bilbo hat zwei gute Verstecke für sein Futter.

Auf einer Karte markiert er die beiden Felder, in denen die Verstecke liegen, mit ✕.
Aber was ist, wenn andere Biber die Karte und damit die Verstecke finden?

Zur Verwirrung markiert Bilbo weitere Felder mit ✕.
Das macht er so, dass in jeder Zeile und Spalte der Karte
eine gerade Anzahl an Feldern markiert ist (oder gar keines).

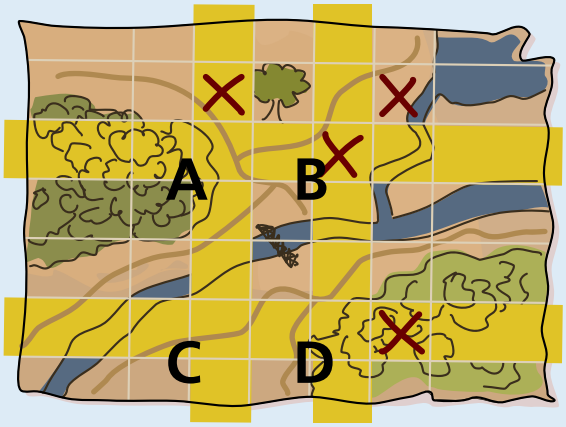
Danach entfernt er die beiden ✕ von den Feldern mit seinen Verstecken.
Unten siehst du das Ergebnis.

In welchen Feldern liegen Bilbos Verstecke?



So ist es richtig:

Um die Felder zu finden, in denen die Verstecke liegen, schauen wir uns die Karte genau an. Die Markierungen der Verstecke wurden ja entfernt. Wir wissen, dass vorher in allen Zeilen und Spalten eine gerade Anzahl Markierungen vorhanden war (oder gar keine). Wir stellen fest, dass es nun zwei Zeilen und zwei Spalten mit Markierungen gibt, in denen die Anzahl der Markierungen ✕ nicht gerade ist:



Diese Zeilen und Spalten überschneiden sich und haben vier gemeinsame Felder (A, B, C und D). Diese „Schnittfelder" sind besonders interessant. Nur wenn in einem Schnittfeld eine Markierung hinzukommt, wird die Zahl der Markierungen in Zeile und Spalte des Feldes anschließend gerade sein. Daher muss Bilbos Futter in den Schnittfeldern liegen.

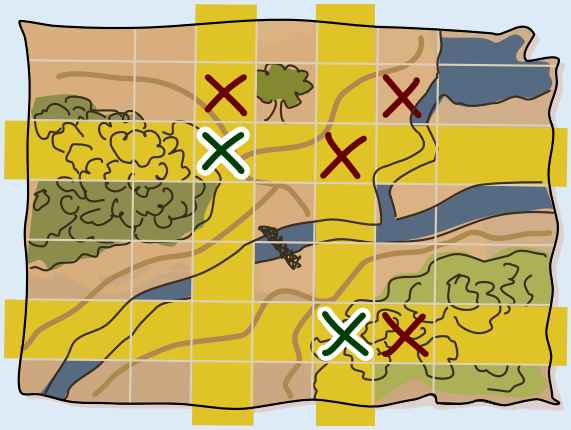
Das *Schnittfeld B* ist markiert. Dort kann kein Versteck sein, da Bilbo die Markierungen der Verstecke entfernt hat.

Das *Schnittfeld A* muss vorher markiert gewesen sein; nur so konnte die Zeile von A eine gerade Anzahl von Markierungen haben. In A liegt also ein Versteck.

In *Schnittfeld C* kann kein Versteck sein: Sonst hätte die Spalte von A und C vorher drei Markierungen enthalten, also eine ungerade Zahl.

Das zweite Versteck muss also in *Schnittfeld D* liegen.

So sah also die Karte aus, bevor Bilbo die Markierungen der Verstecke entfernt hat, mit einer geraden Anzahl von Markierungen ✕ in jeder Zeile und Spalte:



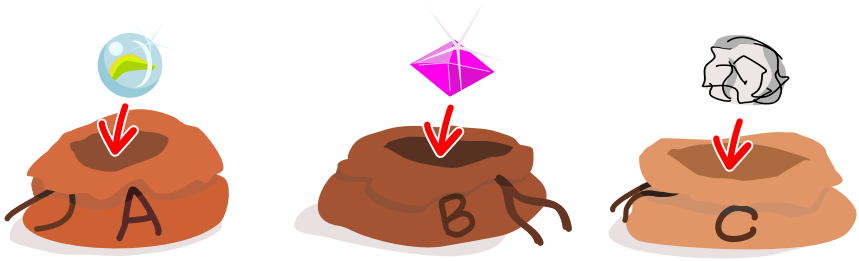
Das ist Informatik!

Bilbo verwendet in dieser Biberaufgabe eine Technik, die in der Informatik gut bekannt ist, nämlich Paritätsbits. Diese Technik ist dem Gebiet der fehlererkennenden bzw. -korrigierenden Codes zuzuordnen. In diesem Gebiet befasst sich die Informatik mit dem Problem, dass beim Speichern oder Übertragen von Daten Fehler auftreten können. Sind die Daten Folgen von Bits (0 oder 1), kann z.B. ein Bit „gedreht" werden: dann wird aus einer 0 eine 1 oder umgekehrt. Um solche Fehler erkennen und evtl. sogar korrigieren zu können, kann man den Daten zusätzliche Bits hinzufügen. Diese Bits nennt man *Paritätsbits,* wenn ihr Wert so gesetzt wird, dass die Anzahl der 1en in der gesamten Bitfolge (Datenbits plus Paritätsbits) eine bestimmte Parität hat, also entweder „gerade" oder „ungerade" wird.

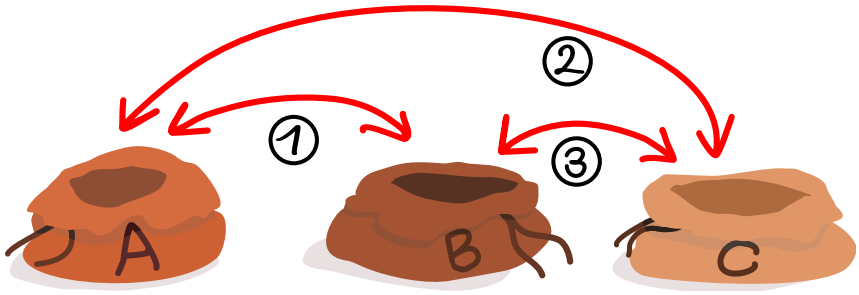
Ein Beispiel: Wenn wir, wie Bilbo, ein Paritätsbit hinzufügen und mit der Parität „gerade" arbeiten, würden wir der Bitfolge 0110101 eine 0 hinzufügen, damit die Anzahl der 1en gerade bleibt. Wird bei der Übertragung der Ergebnisfolge 01101010 zum Beispiel das zweite Bit gedreht, erhält der Empfänger die Folge 00101010. Deren Parität ist „ungerade", so dass der Empfänger den Übertragungsfehler erkennen kann. Korrigieren kann er ihn leider nicht, da er nicht erkennen kann, welches Bit gedreht wurde. Sollten bei der Übertragung gleich zwei Bits gedreht worden sein, kann der Empfänger auch das nicht erkennen, denn dann ist die Parität der empfangenen Bitfolge weiterhin „gerade".

Vertauschen

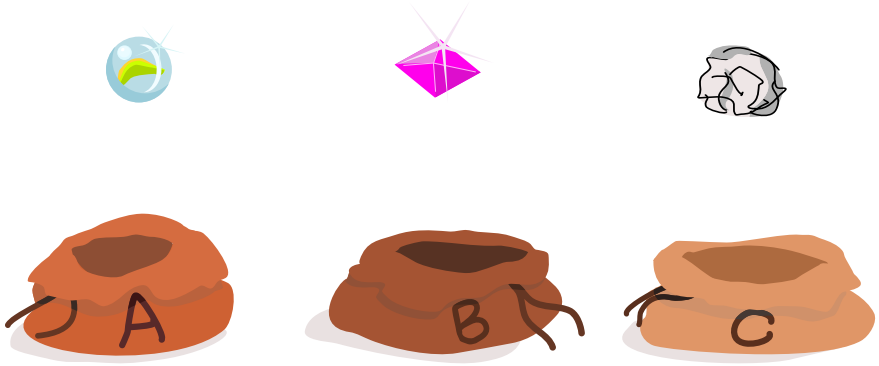
Lila legt eine Murmel in Beutel A, einen Edelstein in Beutel B und ein Stück Papier in Beutel C.



Dann vertauscht sie die Inhalte:
zuerst die von Beutel A und Beutel B, danach die von A und C,
und zuletzt die von B und C.



Wo sind die drei Dinge dann?



So ist es richtig:

Am Anfang haben wir diese Anordnung der drei Dinge in den Beuteln:

Lila vertauscht die Dinge drei Mal.
Nach der ersten Vertauschung (A-B) sehen die Inhalte der Beutel so aus:

Nach der zweiten Vertauschung (A-C) so:

Nach der dritten und letzten Vertauschung (B-C) so:

Am Ende ist also das Papier in Beutel A, der Edelstein in B und die Murmel in C. Dieses Ergebnis hätte man auch einfacher erreichen können, nämlich durch eine einzige Vertauschung der Inhalte von A und C.

Das ist Informatik!

Hier geht es um Reihenfolgen von Dingen. Eine solche Reihenfolge von Dingen wird auch als Anordnung bezeichnet. Eine andere Reihenfolge stellt eine andere Anordnung dar. Eine Vertauschung ändert die Reihenfolge der Dinge und führt daher zu einer anderen Anordnung. In unserer Aufgabe haben wir am Anfang die Anordnung Murmel-Edelstein-Papier und nach den drei Vertauschungen die Anordnung Papier-Edelstein-Murmel.

Welche verschiedenen Anordnungen können drei Dinge haben? Stellen wir zunächst nur alle Anordnungen her, bei denen ein bestimmtes Ding an erster Stelle ist. Für die beiden restlichen Dinge gibt es dann nur zwei Anordnungen. Wenn zum Beispiel die Murmel an erster Stelle ist, dann gibt es diese beiden Anordnungen:

Murmel-Edelstein-Papier; Murmel-Papier-Edelstein

Genau so gibt es auch mit den beiden anderen Dingen an erster Stelle jeweils zwei verschiedene Anordnungen. Also gibt es noch vier weitere und damit insgesamt sechs Anordnungen der drei Dinge:

Edelstein-Murmel-Papier; Edelstein-Papier-Murmel
Papier-Murmel-Edelstein; Papier-Edelstein-Murmel

Jede beliebige Anordnung kann nur mit Vertauschungen aus einer anderen Anordnung hergestellt werden. Dazu sind bei n Dingen höchstens $n - 1$ Vertauschungen notwendig.

Zahlenfolgen

Hier siehst du eine Folge von Zahlen, mit Namen X. An den Positionen 1 bis 5 in der Folge X stehen diese Zahlen: 5, 3, 2, 4, 1

	1	2	3	4	5
X	5	3	2	4	1

Die Zahl an einer bestimmten Position beschreiben wir, indem wir Namen und Position einklammern. Ein Beispiel: Die Zahl an Position 2 von Folge X beschreiben wir so: (X 2). Aktuell ist (X 2) = 3.

Eine so beschriebene Zahl in der Folge kann selbst auch eine Position sein.
Zum Beispiel ist (X (X 2)) = (X 3) = 2.

Hier sind drei andere Folgen: A, B und C.

A	3	2	4	1	5
B	5	4	1	3	2
C	2	5	4	3	1

Welche Zahl beschreiben wir so: (A (B (C 3))) ?

4 ist die richtige Antwort:

Die Beschreibung (A (B (C 3))) sagt: Die beschriebene Zahl steht in Folge A an Position (B (C 3)); die Position steht also in Folge B an Position (C 3); und diese Position steht wiederum in Folge C an Position 3. Kompliziert. Einfacher geht es, wenn wir die Beschreibung „von innen nach außen“ auswerten, wie bei einem Rechenausdruck – und wie es im Beispiel der Aufgabenstellung bereits vorgemacht wird:
(A (B (C 3))) = (A (B 4)) = (A 3) = 4

Das ist Informatik!

Es ist noch gar nicht so lange her, da hat man von „Datenverarbeitung“ gesprochen, wenn es um den Einsatz von Computern ging. Zu Recht, denn Computer verarbeiten unterschiedlichste Arten von Daten, wie Zahlen, Texte, Bilder, Töne usw. Die meisten interessanten, in Computern gespeicherten Daten sind komplexer Art und haben Struktur: Die über den Tag gemessenen Temperaturen an einer Wetterstation zum Beispiel kann man als Folge von Paaren speichern, die jeweils aus der Uhrzeit der Messung und der gemessenen Temperatur bestehen. Hier gibt es also eine Paar- und eine Reihenfolge-Struktur.

Daten können unterschiedlichste Strukturen haben, und so haben Informatikerinnen und Informatiker unterschiedlichste so genannte *Datenstrukturen* entwickelt, um Daten geschickt zu speichern und (genau so wichtig) effizient auf die Daten zugreifen zu können. Eine einfache Datenstruktur ist das Array (auf Deutsch auch: Feld), das in dieser Biberaufgabe die Hauptrolle spielt. Ein Array kann nämlich eine feste Anzahl an Daten (also auch Zahlen) an aufeinanderfolgenden Positionen speichern. Durch die Positionen haben die Daten im Array eine Reihenfolge-Struktur – ein Array wäre also gut für die oben genannten Uhrzeit/Temperatur-Paare geeignet. Wegen ihrer festen Größe gehören Arrays in der Informatik zu den *statischen* Datenstrukturen. Für Datenfolgen gibt es auch *dynamische* Datenstrukturen wie z.B. Listen, deren Größe sich je nach Bedarf ändern kann.

Ob statisch oder dynamisch: Wenn eine Folgen-Datenstruktur Zahlen enthält, können diese Zahlen auch Positionen in der gleichen oder einer anderen Folge angeben. Das wird in der Informatik häufig für die sogenannte „indirekte Adressierung“ benutzt: Die Adresse bzw. Position in einer Folge wird nicht direkt als Zahl, sondern indirekt durch einen (Zahlen-)Wert aus einer Folge angegeben, der auch selbst wieder indirekt adressiert werden kann – usw.

Zauberland

Im Zauberland gibt es vier verschiedene magische Objekte:

Zauberhüte  , Kristallkugeln  , Zauberbücher  und Zaubertränke  .

Zauberhüte und Kristallkugeln können jeweils auf zwei verschiedene Weisen verwandelt werden.

Die Tabelle zeigt, was dabei aus den Objekten entsteht – genau an der Stelle, wo sie vorher waren, und genau in der gezeigten Anordnung:

Aus	entsteht
	
	
	
	

Verwandlungen können beliebig oft und in beliebiger Reihenfolge passieren.
So kann aus einem einzigen magischen Objekt eine lange Anordnung von Objekten entstehen.

Welche Anordnung kann aus einem einzigen Zauberhut NICHT entstehen?

A 

B 

C 

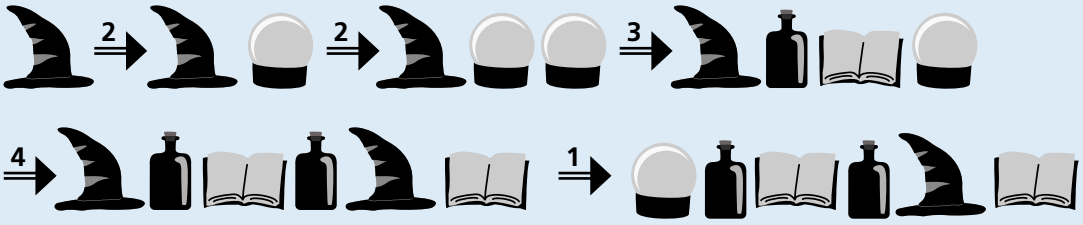
D 

Antwort B ist richtig:

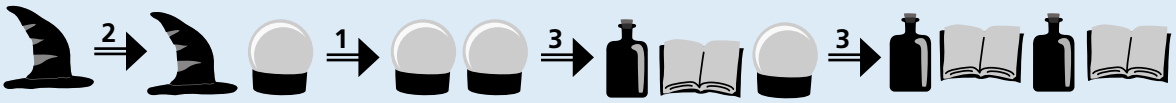
Zunächst nummerieren wir die vier Verwandlungen mit 1 bis 4, in der Tabelle von oben nach unten. Aus einem einzigen Zauberhut können die Anordnungen der Antworten A, C und D entstehen:

Antwort A:

Antwort C:



Antwort D:



Die Anordnung in Antwort B enthält unterschiedlich viele Zauberbücher (2) und Zaubertränke (3). Aber immer, wenn bei einer Verwandlung ein Zaubertrank entsteht (nämlich bei den Verwandlungen 3 und 4), entsteht auch ein Zauberbuch. In jeder Anordnung, die im Zauberland durch Verwandlungen entsteht, muss es also genau so viele Zauberbücher wie Zaubertränke geben. Die Anordnung in Antwort B kann also nicht entstehen, weder aus einem Zauberhut noch aus einer Kristallkugel.

Das ist Informatik!

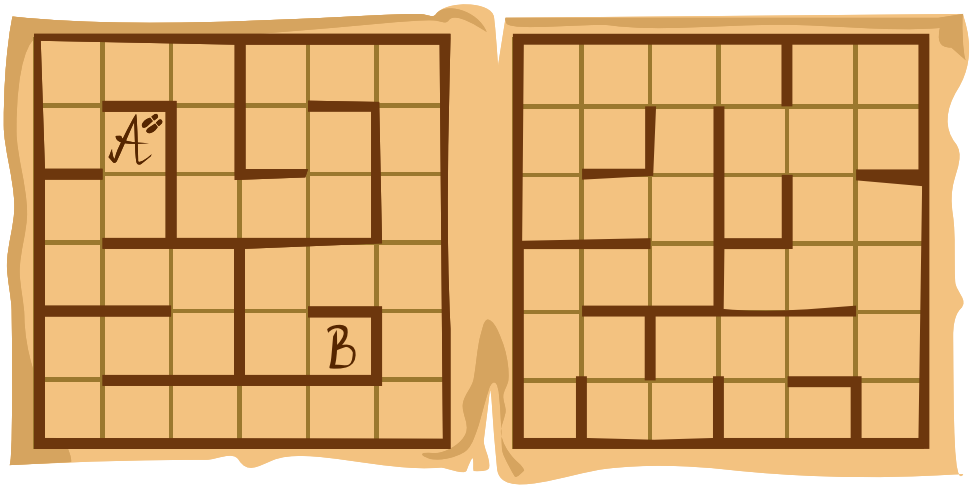
Wenn die Kristallkugeln und Zauberhüte aus dieser Biberaufgabe immer wieder verwandelt werden, entstehen immer wieder andere Anordnungen. Weil bei den Verwandlungen auch immer wieder neue Kristallkugeln und Zauberhüte entstehen, können unendlich viele Anordnungen entstehen. Es ist also nicht möglich, alle Anordnungen, die mit Hilfe der Verwandlungen entstehen können, aufzuschreiben. Aber das ist auch nicht nötig, denn die Menge der Anordnungen ist durch die Verwandlungen selbst genau festgelegt.

Noch bevor es Computer gab, ist die Idee entstanden, unendliche Mengen von Anordnungen mit Hilfe einer vergleichsweise kleinen und jedenfalls endlichen Menge von Verwandlungen zu beschreiben. In der Informatik heißen die Verwandlungen „Ersetzungsregeln“, Regelmengen heißen „Grammatiken“, und die Mengen, die sie definieren, werden konsequenterweise als „Sprache“ bezeichnet. Eine zentrale Rolle bei diesen „formalen Sprachen“ der Informatik spielt das Entscheidungsproblem: Gehört eine Anordnung von Objekten zur Sprache (also: kann es durch Anwendung der Regeln entstehen) oder nicht?

Beim Beantworten dieser Biberaufgabe musstest du dieses Problem für vier Anordnungen lösen. Zum Glück fällt die Zauberland-Grammatik in die Klasse der „kontextfreien Grammatiken“: Kristallkugeln und Zauberhüte können sich verwandeln, ohne zu beachten, welche Dinge sich um sie herum, also in ihrem Kontext, befinden. Für kontextfreie Grammatiken ist das Entscheidungsproblem generell gut lösbar, weshalb sie in der Informatik sehr beliebt sind, zum Beispiel um Programmiersprachen zu beschreiben.

Zauberschule

Die Zauberschule hat zwei Stockwerke. Die Stockwerke liegen genau übereinander. Beide sind in Felder eingeteilt, und es gibt Wände zwischen einigen Feldern:



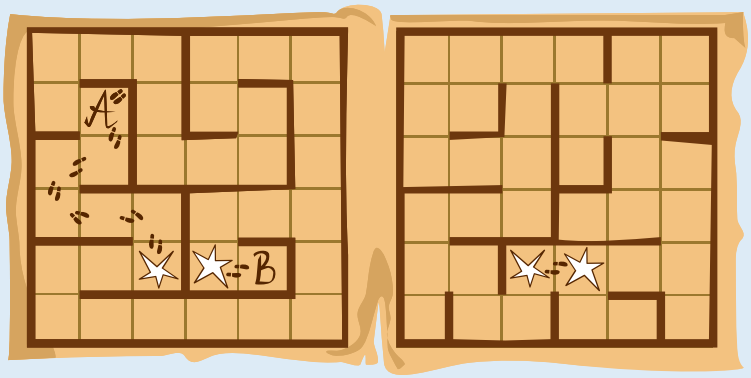
Zauberschüler Ron braucht 1 Sekunde, um auf dem gleichen Stockwerk von einem Feld zum nächsten zu gehen. Leider hat Ron vergessen, wie er durch Wände gehen kann. Er kann aber von einem Stockwerk zum entsprechenden Feld des anderen Stockwerks gelangen; dazu braucht er 5 Sekunden.

Ron möchte von Feld A zu Feld B gelangen, und zwar so schnell wie möglich.

Wie viele Sekunden braucht Ron dazu mindestens?

A 6 **B** 16 **C** 18 **D** 20

Antwort C ist richtig:

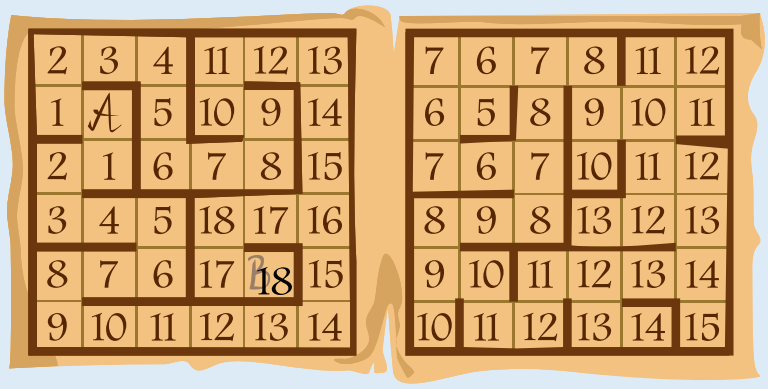


So kann Ron in 18 Sekunden von A nach B gelangen:

Aber ist das der schnellste Weg? Die „kürzesten Zeiten“ die Ron benötigt, um von Feld A aus zu irgendeinem anderen Feld zu gelangen, kann man nach und nach so berechnen:

Für Feld A beträgt die kürzeste Zeit offensichtlich 0 Sekunden. Dann geht es schrittweise so weiter: Von allen Feldern, für die bereits eine kürzeste Zeit eingetragen ist, wählt man das mit dem geringsten Wert. (Ganz am Anfang wählt man also Feld A.) Von diesem gewählten Feld aus betrachtet man alle möglichen nächsten Felder und überlegt, wie man vom gewählten Feld am schnellsten dort hin kommt; die berechneten Zeiten trägt man bei den nächsten Feldern ein. Dabei kann es passieren, das eine vorher eingetragene Zeit verbessert wird. Das gewählte Feld darf danach nicht mehr betrachtet werden; es kann also in den nächsten Schritten nicht mehr gewählt werden.

Hier sind die kürzesten Zeiten, die mit dieser Methode berechnet werden, ausgehend von Feld A:



Ron braucht also wirklich mindestens 18 Sekunden, um von Feld A zu Feld B zu gelangen. 6 Sekunden (Antwort A)wäre die Dauer des kürzesten Weges, wenn es keine Wände zwischen den Feldern gäbe. Wenn Ron dann trotzdem einmal zwischen den Stockwerken hin und her wechselte, würden 16 Sekunden (Antwort B) daraus. Gäbe es nur das Stockwerk mit den Feldern A und B, wären 20 Sekunden (Antwort D) die kürzeste Zeit für den Weg von A nach B.

Das ist Informatik!

Schnellste oder kürzeste Wege müssen recht häufig berechnet werden; ein offensichtliches Beispiel ist die Routenplanung in modernen Karten-Apps. Das Problem wird deutlich vereinfacht, wenn die Wege aus einzelnen Schritten zwischen benachbarten Punkten bestehen und für alle diese Schritte bekannt ist, wie viel sie „kosten“: Zeit, Geld, Energieverbrauch – was auch immer die für das aktuelle Problem wichtige Größe ist. In diesem Fall lassen sich Punkte, Schritte und die Kosten der Schritte zu einem Graph abstrahieren, in dem Schritte zu Wegen zusammengesetzt werden können. Für Graphen sind in der Informatik viele Algorithmen bekannt, mit denen kürzeste Wege effizient berechnet werden können. Einer davon wurde vom Informatiker Edsger Dijkstra (1930-2002) erfunden; dieser *Dijkstra-Algorithmus* kam oben bei der Erklärung der richtigen Antwort zum Einsatz.

Auch beim Entwurf von Schaltkreisen für Computer spielen kürzeste Wege eine wichtige Rolle. Dabei müssen Schaltpunkte mit möglichst geringen Kosten miteinander verdrahtet werden. Moderne Schaltkreise bestehen aus mehreren Ebenen, und eine Verdrahtung zwischen zwei Ebenen ist teurer als eine (ansonsten vergleichbare) Verdrahtung auf der gleichen Ebene – ähnlich zum Wechsel zwischen den Stockwerken in dieser Biberaufgabe, der teurer ist als ein Schritt auf dem gleichen Stockwerk.


BIBER DER INFORMATIK

Biber der Informatik 2022
Aufgaben und Lösungen

OESTERREICHISCHE COMPUTER GESELLSCHAFT®
AUSTRIAN COMPUTER SOCIETY
Ein Wettbewerb der OCG

ocg.at/biber