Biber der Informatik 2020
Aufgaben und Lösungen

OESTERREICHISCHE COMPUTER GESELLSCHAFT ®
AUSTRIAN COMPUTER SOCIETY

Ein Wettbewerb der OCG

ocg.at/biber

Der österreichischen Biber der Informatik Team 2020

Wilfried Baumann, Österreichische Computer Gesellschaft
Anoki Eischer, Österreichische Computer Gesellschaft
Gerald Futschek, Technische Universität Wien
Katharina Resch-Schobel, Österreichische Computer Gesellschaft
Florentina Voboril, Studentin Technische Universität Wien

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

Wilfried Baumann, Österreichische Computer Gesellschaft
Michael Barot, Kantonsschule Schaffhausen
Wilfried Baumann, Österreichische Computer Gesellschaft
Robert Czechowski, BWINF
Christian Datzko, Wirtschaftsgymnasium und Wirtschaftsmittelschule Basel
Susanne Datzko, freischaffende Graphikerin, ETH Zürich
Anoki Eischer, Österreichische Computer Gesellschaft
Nora Escherle, SVIA *
Fabian Frei, ETH Zürich, ABZ **
Gerald Futschek, Technische Universität Wien
Jens Gallenbacher, ETH Zürich, ABZ
Martin Guggisberg, Pädagogische Hochschule FHNW / SVIA
Juraj Hromkovic, ETH Zürich, ABZ
Dennis Komm, ETH Zürich, ABZ
Regula Lacher, ETH Zürich, ABZ
Lucio Negrini, Scuola universitaria professionale della Svizzera italiana / SVIA
Gabriel Parriaux, Haute École Pédagogique Vaud / SVIA
Jean-Philippe Pellet, Haute École Pédagogique Vaud / SVIA
Katharina Resch-Schobel, Österreichische Computer Gesellschaft
Peter Rossmanith, RWTH Aachen
Lars Teuber, BWINF
Florentina Voboril, Technische Universität Wien

* Schweiz. Verein für Informatik in der Ausbildung
** Ausbildungs- und Beratungszentrum für Informatikunterricht

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

Einleitung

Der Biber der Informatik ist ein internationaler Online-Bewerb mit Aufgaben zur Informatik, der in Österreich von der OCG durchgeführt wird. Er erfordert Informatisches Denken, aber keine Vorkenntnisse. Der Biber der Informatik will aufzeigen, wie vielfältig und spannend Fragestellungens der Informatik sein können.

Der Biber der Informatik findet jährlich im November statt. An der 14. Austragung im Jahr 2020 beteiligten sich 2.356 Schulen und andere Bildungseinrichtungen mit 19.741 Schülerinnen und Schülern, trotz aller Widrigkeiten der Corona-Pandemie.
Die Möglichkeit, auch in Zweierteams zu arbeiten, wurde gern genutzt.

Die Teilnahme am Biber der Informatik 2020 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 Informatik-Biber 2020 wurde in fünf Altersgruppen durchgeführt. In den Klassenstufen 3 bis 4 waren innerhalb von 30 Minuten 9 Aufgaben zu lösen, jeweils drei in den Schwierigkeits-stufen leicht, mittel und schwer. In den Klassenstufen 5 bis 6 waren innerhalb von 35 Minuten 12 Aufgaben zu lösen, jeweils vier in den Schwierigkeitsstufen leicht, mittel und schwer. In den Klassenstufen 7 bis 8, 9 bis 10 und 11 bis 13 waren innerhalb von 40 Minuten 15 Aufgaben zu lösen, jeweils fünf in den Schwierigkeitsstufen leicht, mittel und schwer.

Die 36 Aufgaben des Informatik-Biber 2020 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.

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

Wir laden die Schülerinnen und Schüler ein, auch 2021 wieder beim Informatik-Biber mitzumachen, und zwar in der Zeit vom 8. bis 19. November. Weitere Informationen werden über die Website ocg.at/biber und per E-Mail an die Koordinatorinnen und Koordinatoren bekannt gegeben.

Bebras: International Challenge on Informatics and Computational Thinking

Der pakistanische Biber

Der österreichische Biber der Informatik ist Partner der internationalen Initiative Bebras. 2004 fand in Litauen der erste Bebras Challenge statt. 2006 traten Estland, die Niederlande und Polen der Initiative bei, und und auch Deutschland veranstaltete einen ersten Biber-Testlauf. 2007 war Österreichs erster Biber der Informatik. Seitdem kamen viele Bebras-Länder hinzu. Zum Drucktermin sind es weltweit 61, und weitere Länderteilnahmen sind in Planung. Insgesamt hatte der Bebras Challenge 2020 weltweit etwa 2,5 Millionen Teilnehmerinnen und Teilnehmer – wegen der Corona-Pandemie deutlich weniger als im Vorjahr.

Die Bebras-Community erarbeitet jedes Jahr auf einem internationalen Workshop anhand von Vorschlägen der Länder eine größere Auswahl möglicher Aufgabenideen. Die Ideen zu den 36 Aufgaben des Informatik-Biber 2020 stammen aus 22 Ländern: Belgien, Deutschland, Indien, Island, Japan, Kanada, Litauen, Neuseeland, Niederlande, Nordmakedonien, Österreich, Pakistan, Philippinen, Portugal, Russland, Schweiz, Slowakei, Südkorea, Tschechien, Türkei, Ungarn und Vietnam.

Der italienische Biber

Der estnische Biber

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

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

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.

OESTERREICHISCHE COMPUTER GESELLSCHAFT ®
AUSTRIAN COMPUTER SOCIETY

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.

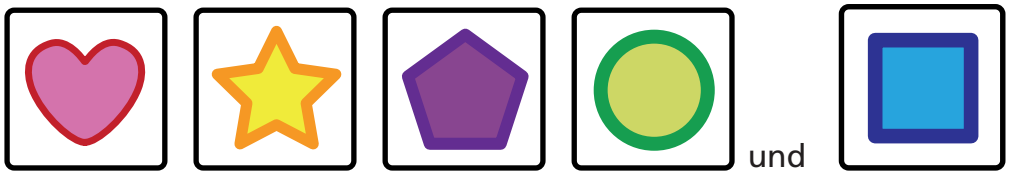
ioi
INTERNATIONAL OLYMPIAD IN INFORMATICS

Aufgabenliste

Das sind die 36 Aufgaben des Informatik-Biber 2020, grob geordnet nach steigender Schwierigkeit und gelistet mit einer Klassifikation ihres informatischen Inhalts.

Am schwersten

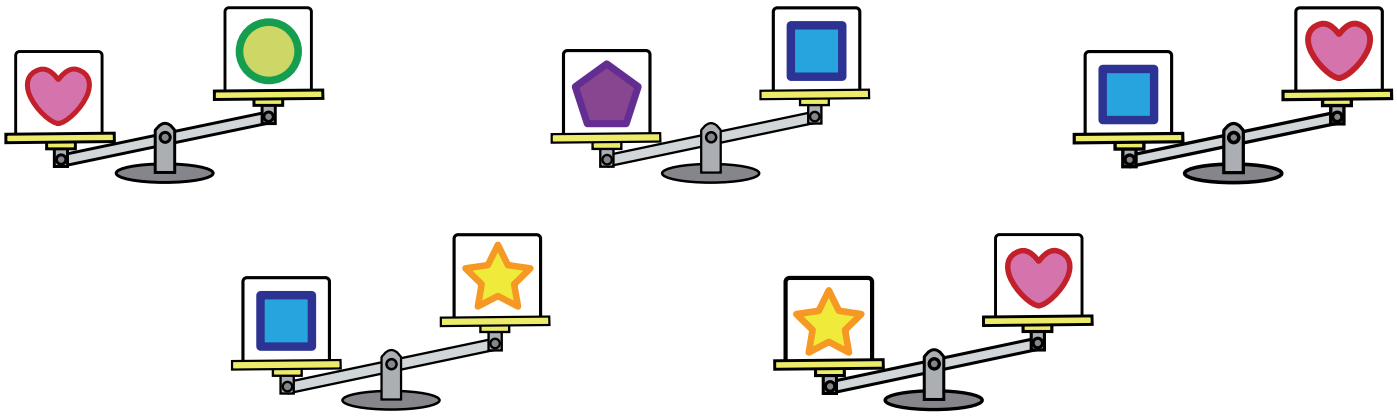
Fünf Kisten sind mit fünf verschiedenen Bildern markiert:

 und

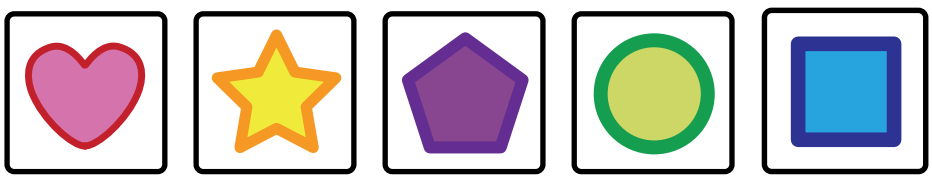
Mit einer Waage werden jeweils zwei Kisten verglichen.

Zum Beispiel ist  schwerer als 

Insgesamt wird fünf Mal verglichen:



Welche Kiste ist am schwersten?



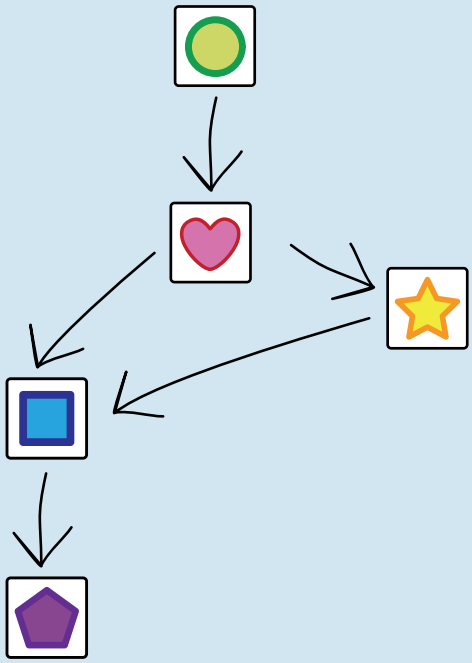
Die Kiste mit dem Fünfeck ist am schwersten.

Das ergeben die Vergleiche: Die Kiste mit dem Fünfeck ist schwerer als die Kiste mit dem Quadrat . Die Kiste mit dem Quadrat ist schwerer als die Kiste mit dem Stern . Die Kiste mit dem Stern ist schwerer als die Kiste mit dem Herz . Und die Kiste mit dem Herz ist schwerer als die Kiste mit dem Kreis . Indirekt ist die Kiste mit dem Fünfeck also schwerer als alle anderen Kisten. Wer clever ist, kann bei dieser Aufgabe abkürzen und Folgendes überlegen: Die schwerste Kiste kann nicht leichter als eine andere Kiste sein und deshalb beim Vergleich mit der Waage nur unten und niemals oben landen. Das gilt für genau eine Kiste, und zwar für die mit dem Fünfeck.

Das ist Informatik!

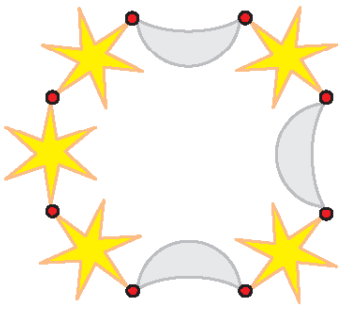
Aus einer Menge von Dingen kann man das schwerste (oder größte oder längste oder …) bestimmen, indem man die Dinge dem Gewicht (oder der Größe oder der Länge oder …) nach sortiert und dann an das Ende der Sortierung schaut. Auch die Frage in dieser Biberaufgabe kann also durch Sortieren beantwortet werden.

Die Ergebnisse von fünf Vergleichen sind bekannt. Man kann sie so aufzeichnen wie im Bild: Von einer Kiste zu einer anderen zeigt ein Pfeil, wenn die beiden Kisten verglichen wurden und die eine Kiste leichter ist als die andere. Daraus bildet man die Sortierung; sie ist zu Beginn leer, enthält also noch keine Kiste. Jetzt nimmt man nach und nach eine Kiste weg (und die Pfeile, die von ihr ausgehen), auf die kein Pfeil zeigt. Wir stellen diese Kiste an das Ende der Sortierung – womit die erste Kiste gleichzeitig auch den Anfang der Sortierung bildet). Unter den danach noch vorhandenen Kisten gibt es nämlich keine, die leichter als die weggenommene Kiste ist und deshalb in der Sortierung vorher kommen müsste. Wenn man auf diese Weise alle Kisten weggenommen hat, erhält man eine Sortierung. An deren Ende steht die schwerste Kiste: die mit dem Fünfeck.



Dieses Verfahren, aus einer Menge einzelner Vergleiche eine Sortierung zu bilden, ist in der Informatik als *topologische Sortierung* bekannt. Aber Achtung, es funktioniert nicht immer. Wenn im Bild einige Kisten und Pfeile einen Kreis bilden, gibt es in diesem Kreis keine Kiste, auf die kein Pfeil zeigt, und man muss die Sortierung abbrechen. Außerdem muss jede Kiste an mindestens einem Vergleich beteiligt sein, sonst weiß man nichts über sie und kann sie nicht einsortieren.

Armband

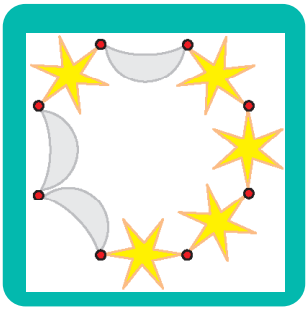


Marie wünscht sich ein Armband aus Monden und Sternen, so wie rechts.
Sie kann aber nicht gut zeichnen und gibt lieber Anweisungen, wie das Armband gemacht werden soll:

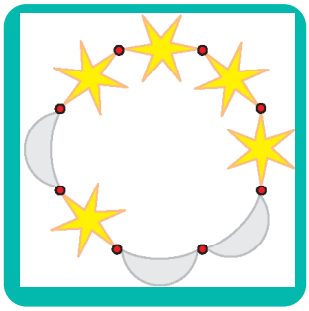
1. Verbinde einen Stern und einen Mond  zu einem Paar.
2. Mache den ersten Schritt noch zweimal, so dass du am Ende drei Paare hast.
3. Verbinde die drei Paare zu einer Kette.
4. Füge weitere zwei Sterne an einem Ende der Kette an.
5. Verbinde die Enden zu einem Armband.

Welches Armband kann NICHT nach Maries Anweisungen gemacht werden?

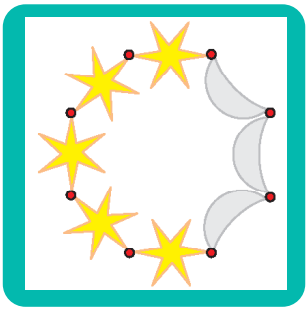
A)



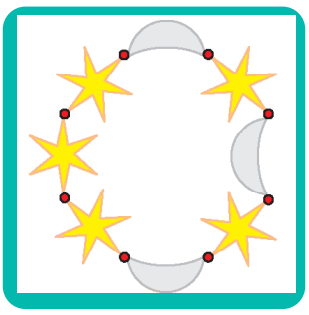
B)



C)

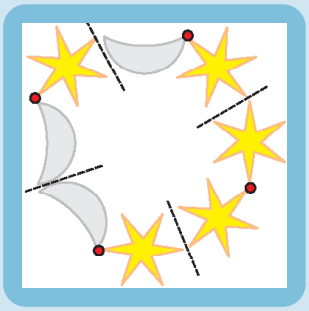


D)

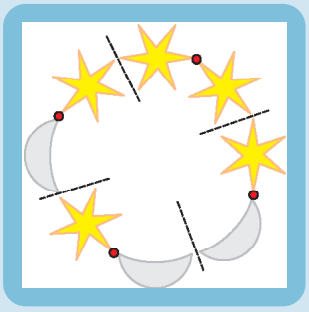


Antwort C ist richtig:
Die Bilder zeigen, dass jedes andere Armband in drei Stern-Mond-Paare und ein an diese drei Paare angefügtes Stern-Stern-Paar zerteilt werden und deshalb nach den Anweisungen gemacht werden kann.

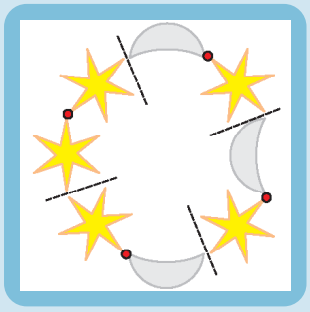
A)



B)



D)



Nach Anweisung 1 muss jeder Mond im Armband neben einem Stern sein. In einem Armband, das nach den Anweisungen gemacht wurde, können deshalb nicht drei Monde hintereinander vorkommen – so wie in Armband C.

Das ist Informatik!
Maries Anweisungen in dieser Biberaufgabe haben also nicht ein einzelnes Armband beschrieben, sondern viele Armbänder: nämlich die Menge aller Armbänder, die nach den Anweisungen gemacht werden können. Hätte Marie sich in Anweisung 3 nicht auf drei Paare beschränkt, sondern die Anzahl der Paare offen gelassen, wäre diese Menge sogar unendlich groß.

Computer sind nicht unendlich groß; insbesondere ist ihr Speicher begrenzt. Damit sie trotzdem mit potenziell unendlich großen Mengen umgehen können, verwendet die Informatik Methoden, Mengen auf endliche Art und Weise zu beschreiben – wie Maries Anweisungen in dieser Biberaufgabe.

Eine solche Informatik-Methode ist, die Struktur der Elemente einer Menge durch Regeln zu beschreiben. Versuchen wir das einmal mit Maries Armbändern:

- Ein Armband A besteht aus einer Kette K und zwei Sternen S: A $\longrightarrow$ KSS
- Eine Kette K besteht aus drei Paaren P: K $\longrightarrow$ PPP
- Ein Paar P besteht aus einem Stern S und einem Mond M (oder umgekehrt): P $\longrightarrow$ SM, P $\longrightarrow$ MS

Maries Wunscharmband entsteht, wenn man die Regeln so anwendet
($\longrightarrow$ steht für die Anwendung einer Regel):

A $\longrightarrow$ KSS $\longrightarrow$ PPPSS $\longrightarrow$ MSPPSS $\longrightarrow$ MSMSPSS $\longrightarrow$ MSMSMSSS

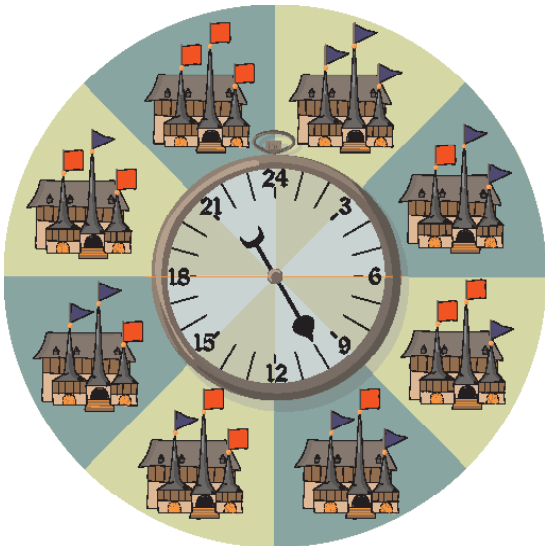
Die Regeln beschreiben, wie ein Startsymbol (hier das A für Armband) nach und nach durch andere Symbole ersetzt werden kann, bis nur noch bestimmte Endsymbole (hier S und M für Stern und Mond) vorkommen. Alle Kombinationen von Endsymbolen, die durch Anwendung der Ersetzungsregeln auf das Startsymbol entstehen können, gehören zu der durch die Regeln beschriebenen Menge. In der Informatik heißt eine solche Menge von Ersetzungsregeln mit Start- und Endsymbolen *Grammatik,* und die durch die Regeln beschriebene Menge heißt *Sprache.* Solche Regelsysteme erinnern nämlich an natürliche Sprachen, bei denen eine Grammatik beschreibt, wie die Sätze der Sprache aufgebaut werden müssen.

Bequeme Biber

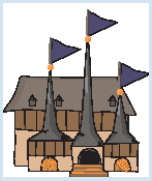
In einem Dorf leben sehr bequeme Biber.
Sie teilen den Tag in nur 8 Zeitabschnitte zu je 3 Stunden ein.
Am Rathaus zeigen drei Flaggen den aktuellen Zeitabschnitt an.
Es gibt zwei Arten von Flaggen: rotes Quadrat und blaues Dreieck.

Die Flaggenanzeige ist so aber noch nicht bequem genug.
Die Biber wünschen sich: Immer wenn ein neuer Zeitabschnitt beginnt, soll nur eine Flagge gewechselt werden.

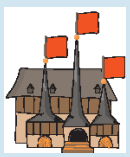
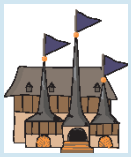
Ändere die Flaggenanzeige so, wie die Biber wünschen.



So ist es richtig:
Eine Lösung kann man durch geschicktes Ausprobieren finden. Die einzige „Problemstelle“ in der Flaggenanzeige ist der Übergang

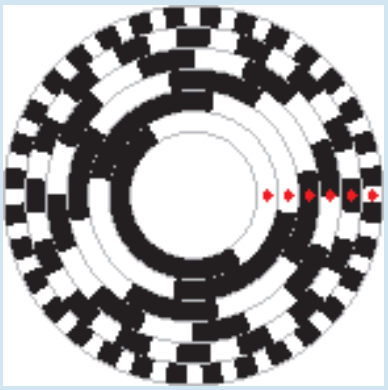
von  zu  um Mitternacht. Wenn man  mit  (15-18 Uhr) tauscht, ist der Wunsch der Biber erfüllt:

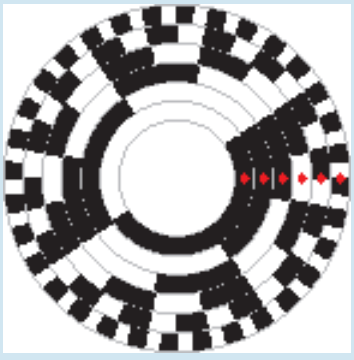
Systematisch findet man eine Lösung mit der folgenden Methode:
Die acht Flaggenbilder kann man mit dreistelligen Binärzahlen beschreiben; 0 steht für ein rotes Quadrat und 1 für ein blaues Dreieck:

000	001	010	011	100	101	110	111
							

Wir müssen also für die Binärzahlen 000, 001, 010, 011, 100, 101 110, 111 eine Anordnung finden, die diese Bedingung erfüllt: Zwei beliebige benachbarte Zahlen sowie die erste und die letzte Zahl unterscheiden sich jeweils nur in einer Stelle.
Wir betrachten zuerst nur die 000. Die schreiben wir zweimal hintereinander, ändern beim zweiten Mal aber die erste Stelle (von rechts) von 0 zu 1. So erhalten wir die Anordnung 000, 001; sie erfüllt die Bedingung.
Jetzt schreiben wir dahinter diese beiden Zahlen nochmals, aber in umgekehrter Reihenfolge (also 001, 000) und ändern nun die zweite Stelle von 0 zu 1 (also 011, 010). So erhalten wir die Zahlenfolge 000, 001, 011, 010, die ebenfalls die Bedingung erfüllt.
Diese neue Zahlenfolge schreiben wir jetzt gleich nochmals und wieder umgekehrt dahinter und ändern nun überall die dritte Stelle von 0 zu 1. So erhalten wir diese Anordnung der acht dreistelligen Binärzahlen: 000, 001, 011, 010, 110, 111, 101, 100. Sie erfüllt die Bedingung und entspricht genau der oben „von Hand“ gefundenen Flaggenanzeige.
Diese Methode (Spiegeln der bestehenden Zahlenfolge und Ändern der nächsthöheren Stelle von 0 zu 1) kann man beliebig lange fortsetzen, um solche Anordnungen für beliebig viele statt nur drei Flaggen bzw. Bits zu erhalten. Man kann sich überlegen, weshalb diese Methode immer funktioniert und dass immer alle möglichen Muster verwendet werden.

Das ist Informatik!
Ein Anordnung von Binärzahlen, welche die „Bequemlichkeits-Bedingung“ dieser Biberaufgabe erfüllt, heißt *Gray-Code,* benannt nach dem Erfinder Frank Gray. Generell kann ein Gray-Code dazu eingesetzt werden, die Übertragung oder Messung binär kodierter Information abzusichern: Wenn benachbarte Werte sich nur in einem Bit unterscheiden, führen kleine Fehler oder Ungenauigkeiten auch nur zu einem leicht falschen Wert.
Ein gutes Beispiel ist das Messen der Winkelposition einer Drehscheibe. Mit Hilfe eines sechsstelligen Binärcodes lassen sich 64 Winkelpositionen unterscheiden. Wir zeichnen den Code der Zahlen 0 bis 63 so auf die Scheibe wie im Bild links unten gezeigt: Weiß steht für 0 und Schwarz für 1; und die Farbwerte eines Codes liegen auf einer geraden Linie von der Mitte nach außen. Die roten Punkte im Bild sind Sensoren, die auf einer Linie angebracht sind und zwischen Schwarz und Weiß unterscheiden können. Mit den Sensoren lässt sich also der aktuelle Drehwinkel der Scheibe ablesen. Im Bild links wird ein Gray-Code verwendet, im Bild rechts die übliche binäre Zahlcodierung.





Was passiert nun bei Ungenauigkeiten? Links befindet sich der vierte Sensor von links genau auf der Grenze zwischen Schwarz und Weiß. Der Sensor liest also entweder 001010 oder 001110. In einem Gray-Code sind dies benachbarte Zahlen, so dass die abgelesene Zahl sich nur leicht vom wirklichen Winkel der Scheibe unterscheidet.
Rechts, beim normalen Binärcode folgen z.B. die Code-Wörter 111010 und 111001 aufeinander, die sich in zwei Bits unterscheiden. Wenn die Sensoren genau dazwischen stehen, sind vier verschiedene Werte möglich: 111000, 111001, 111010 und 111011. Im schlimmsten Fall befänden sich die Sensoren an der Grenze zwischen dem komplett weißen Code-Wort 000000 und dem komplett schwarzen Code-Wort 111111. Dann wären 64 verschiedene Werte möglich, und die Winkelmessung wird unbrauchbar.

Bergsteiger

Greta klettert am liebsten in den Gipfeln des Bibergebirges.

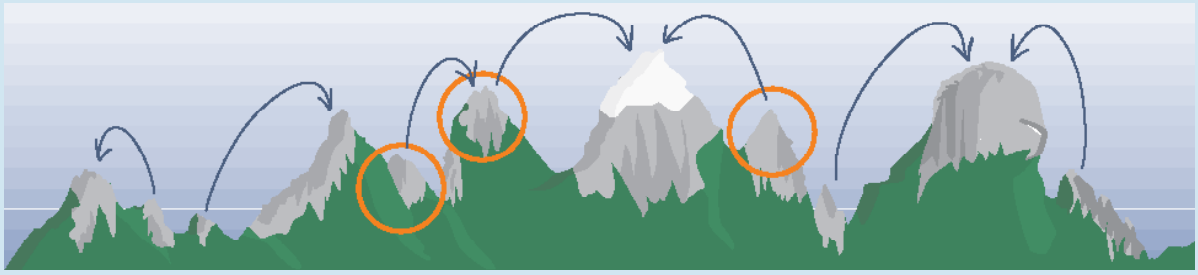
Sobald sie einen Gipfel erreicht hat, schaut sie sich die beiden Nachbar-Gipfel an:

- Ist nur ein Nachbar-Gipfel höher, klettert sie auf diesen Gipfel.
- Sind beide Nachbar-Gipfel höher, klettert sie auf den höheren der beiden.

Das wiederholt sie so lange, bis sie einen Gipfel erreicht, der keinen höheren Nachbarn hat.

Von welchen Gipfeln aus erreicht Greta so den höchsten Gipfel?

So ist es richtig:



Die Pfeile zeigen, ob Greta von einem Gipfel aus weiter klettert und wohin. Von den drei eingekreisten Gipfeln aus erreicht Greta also den höchsten Gipfel, entlang der Pfeile.

Das ist Informatik!
Greta möchte hoch hinaus! Dabei verfolgt sie dieses Prinzip: In jeder Situation (also auf jedem Gipfel) macht sie den nächsten Schritt, der sie möglichst weit nach oben bringt. Wie diese Biberaufgabe zeigt, kommt sie dabei aber nicht immer auf den insgesamt höchsten Gipfel, sondern bleibt auf Gipfeln stecken, die nur in ihrer Umgebung, also lokal, die höchsten sind.

Gretas Prinzip ist in der Informatik als „greedy“ (deutsch: gierig) bekannt. Auf der Suche nach einem optimalen Ergebnis machen gierige Algorithmen immer den nächstbesten Schritt und laufen deshalb Gefahr, ein nur lokales Optimum zu finden und ein insgesamt optimales Ergebnis zu verpassen. Ein Beispiel ist der so genannte Bergsteigeralgorithmus, englisch auch „hill climbing“ genannt, der ganz offensichtlich die Idee zu dieser Biberaufgabe geliefert hat. Algorithmen nach dem Prinzip „greedy“ sind aber einfach zu realisieren und deshalb beliebt. Man kann sie gut verwenden, wenn man ausschließen kann, dass es lokale Optima gibt, oder wenn man mit vielleicht recht guten aber nicht optimalen Ergebnissen zufrieden ist. In der Informatik ist gut bekannt, unter welchen Bedingungen gierige Algorithmen wie gut funktionieren.

Biber-Parcours

Die Informatik-Lehrerin hat für den Unterrichtsbeginn ein Bewegungsspiel eingeführt: den Biber-Parcours. Das Spiel geht so:

Auf dem Tischplan des Klassenraums sind sieben Tische mit Zahlen markiert. Auf jedem dieser Tische liegt ein kleiner Gegenstand.

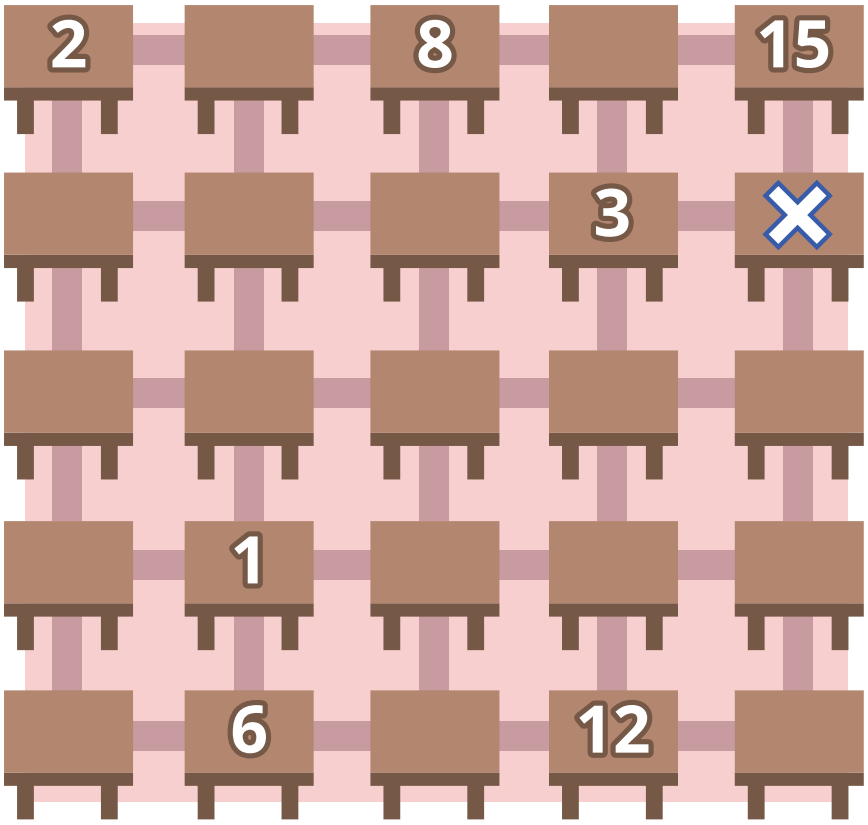
Du sollst innerhalb von 16 Sekunden möglichst viele Gegenstände zum Lehrertisch bringen.

Du kannst an einem beliebigen Tisch starten und dich dann entlang der Linien von Tisch zu Tisch bewegen. Du darfst jede Linie nur einmal entlang gehen. Du brauchst genau 1 Sekunde, um von einem Tisch zum nächsten zu gehen und, falls vorhanden, einen Gegenstand zu nehmen.

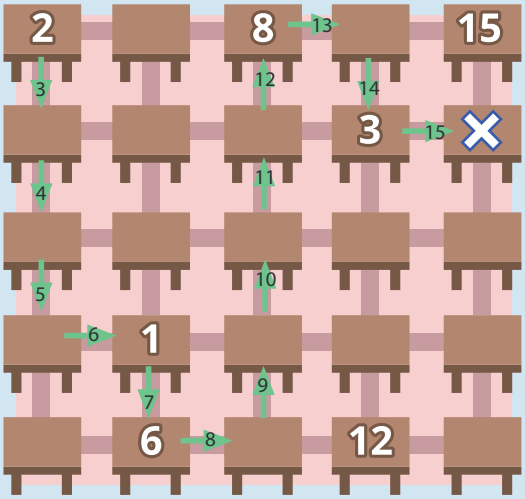
Die Zahl auf einem Tisch sagt dir, nach wie vielen Sekunden du frühestens bei diesem Tisch sein darfst.

Hier ist der Tischplan. Du kannst auf die Linien zwischen den Tischen klicken, um Wege durch den Klassenraum auszuprobieren.

Wie viele Gegenstände kannst du innerhalb von 16 Sekunden höchstens zum Lehrertisch bringen?



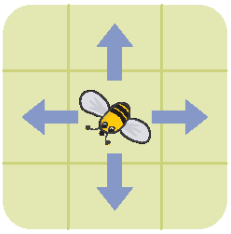
5 ist die richtige Antwort:
Das Bild zeigt einen Weg durch den Klassenraum, auf dem innerhalb von 15 Sekunden (und damit auch innerhalb von 16 Sekunden) fünf Gegenstände zum Lehrertisch gebracht werden können. Ein Pfeil steht für den Gang von einem Tisch zum nächsten. Die Zahl in der Pfeilspitze sagt, nach wie vielen Sekunden man frühestens beim nächsten Tisch sein kann. Damit man bei einem Tisch nicht warten muss, muss die Zahl in der Pfeilspitze größer oder gleich der Zahl auf dem Tisch sein. Es gibt noch einige andere Wege durch den Klassenraum, auf denen man auch fünf Gegenstände nehmen und zum Lehrertisch bringen kann.



Im weiteren heißen die Gegenstände einfach so wie die Zahl auf ihrem Tisch. Es ist nicht möglich, alle sieben Gegenstände innerhalb von 16 Sekunden zum Lehrertisch zu bringen. Frühestens nach 12 Sekunden darf man bei Gegenstand 12 ankommen. Von dort aus kann man frühestens nach weiteren 5 Sekunden bei Gegenstand 15 sein – zu spät, um auch diesen Gegenstand mitbringen zu können. Ist man zuerst (nach 15 Sekunden) bei Gegenstand 15, hat man erst recht keine Zeit, um auch Gegenstand 12 mitzubringen.
Die Gegenstände 12 und 15 schließen sich also aus. Kann man denn sechs Gegenstände rechtzeitig zum Lehrertisch bringen? Dazu müsste man entweder auf 12 oder auf 15 verzichten. Wenn man auf 15 verzichtet, müsste man 8 und 12 mitbringen Das geht aber nicht. Von 8 aus ist man frühestens nach 13 Sekunden bei 12 und schafft es dann nicht mehr rechtzeitig bis zum Lehrertisch. Also muss man auf 12 verzichten. Jetzt kann man sich überlegen, dass man oben rechts die Gegenstände 3, 8 und 15 nur dann rechtzeitig mitbringen kann, wenn man nach 11 Sekunden bei 8 ist. Auf insgesamt sechs Gegenstände kommt man dann wiederum nur, wenn man vorher schon 1, 2 und 6 genommen hat. Man kann durchaus nach 11 Sekunden bei 8 sein, nämlich wenn man nach 6 Sekunden bei 6 ist – und vorher mindestens Gegenstand 2 genommen hat. Denn von 6 aus kann man entweder auf direktem Wege oder über Gegenstand 1 in 5 Sekunden bei 8 sein. Aber wenn man zuerst Gegenstand 2 nimmt, ist man erst nach 7 Sekunden bei 6 und nach 12 Sekunden bei 8. Nimmt man also zuerst 1, 2 und 6 und danach 8, 3 und 15, dauert das eine Sekunde zu lange. Andere Wege mit sechs Gegenständen sind sogar noch länger. Man kann also nur fünf Gegenstände innerhalb von 16 Sekunden zum Lehrertisch bringen.

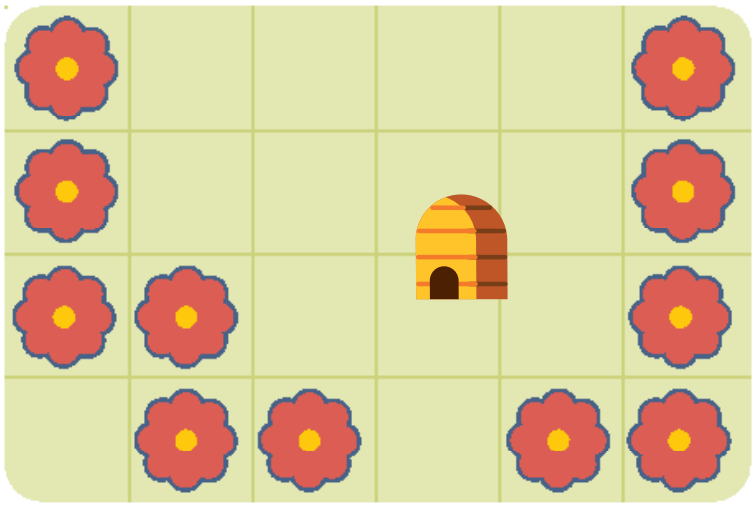
Das ist Informatik!
Wenn man eine Aufgabe auf eine bestmögliche Weise lösen soll, hat man es mit einem Optimierungsproblem zu tun. In dieser Biberaufgabe gibt es viele Lösungen der Aufgabe, Gegenstände unter den gegebenen Bedingungen zum Lehrertisch zu bringen. Optimal ist darunter, laut Fragestellung, eine Lösung mit möglichst vielen Gegenständen. Die Anzahl der Gegenstände bestimmt also, wie gut eine Lösung ist, und liefert damit die Optimierungsfunktion (siehe auch die Aufgabe „Blumenkasten“ in diesem Biberheft). Während hier nur nach dem optimalen Wert dieser Funktion gefragt wurde, ist es häufig auch wichtig, mindestens eine Lösung anzugeben, die diesen Funktionswert hat – eine optimale Lösung also.
Optimierungsprobleme stellen sich in vielen Lebensbereichen: Wie kommt man am schnellsten von der Wohnung zur Schule? Welche Route soll der Paketwagen nehmen, damit er in der vorgesehenen Zeit möglichst viele Pakete ausliefern kann? Es gibt viele Optimierungsprobleme, bei denen es schwierig ist, in allen Fällen die beste(n) Lösung(en) zu finden. Wenn sich die Lösung aus einzelnen Schritten zusammensetzt (wie hier aus den Wegen zwischen den einzelnen Gegenständen), kann man es mit der in der Informatik als „backtracking“ bekannten Methode versuchen: An Stellen, an denen es mehrere Möglichkeiten gibt, den Lösungsversuch um einen Schritt zu ergänzen, wählt man eine davon aus. Das macht man so lange, bis man entweder eine Lösung hat (hier: innerhalb von 16 Sekunden am Lehrertisch angekommen ist) oder sicher ist, dass man mit den bisher gewählten Schritten keine Lösung finden kann oder nur eine, die schlechter ist als die bisher beste. Dann geht man auf der „Spur“ (engl.: track) des aktuellen Lösungsversuchs so lange zurück (engl.: back), bis man an einer Auswahlstelle eine andere Möglichkeit wählen kann.

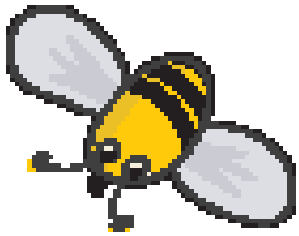
Bienenflug



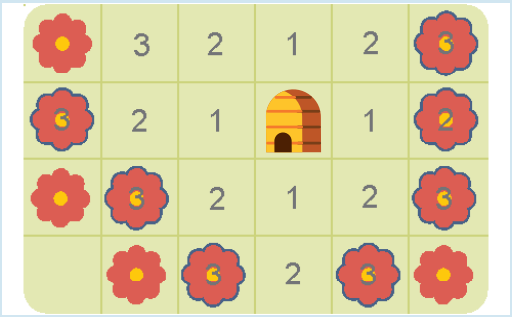
Vom Bienenstock aus können die Bienen drei Felder weit fliegen.
Von einem Feld zum nächsten fliegen sie nach links, rechts, oben oder unten.

Zu welchen Blumen können die Bienen vom Bienenstock aus fliegen?





So ist es richtig:
Das Bild zeigt, zu welchen Feldern die Bienen fliegen können, wenn sie 1 Feld, 2 Felder oder 3 Felder weit fliegen können. Die Blumen mit einer Zahl können sie dabei also erreichen. Diese Blumen sind für die Bienen höchstens 3 Felder weit vom Bienenstock entfernt.
Um zu den anderen Blumen fliegen zu können, müssten die Bienen 4 Felder weit fliegen können. Das können sie aber nicht.



Das ist Informatik!
Diese Biberaufgabe kann gelöst werden, indem vom Bienenstock aus der Abstand zu allen Feldern bestimmt wird. Dann kennt man am Ende auch den Abstand zu allen Blumen. Um bei der Bestimmung des Abstands nicht immer wieder neu beim Bienenstock anfangen zu müssen, kann man schrittweise vorgehen: Erst bestimmt man alle Felder mit Abstand 1 zum Bienenstock. Die Felder mit Abstand 2 zum Bienenstock sind dann alle Felder, die zu den „Abstand-1-Feldern“ den Abstand 1 haben – und nicht schon selbst zu den „Abstand-1-Feldern“ des Bienenstocks gehören. Die Felder mit Abstand 3 werden danach mit Hilfe der Felder mit Abstand 2 gefunden, und so könnte man weiter machen, wenn die Bienen weiter fliegen könnten.
Ein Verfahren, das Ergebnisse mit Hilfe vorher gefundener Ergebnisse berechnet, folgt einer Idee, die in der Informatik als „Dynamische Programmierung“ bekannt ist. Sie wird häufig verwendet, unter anderem bei der Bestimmung von kürzesten Wegen zwischen zwei Orten.

Blumenkasten

Peter liebt Blumen.

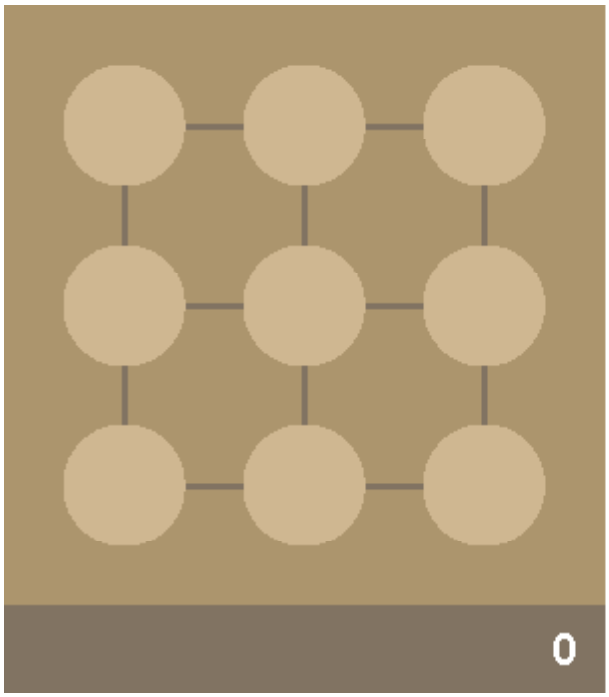
Er hat rote , gelbe und orange Blumen.
Peter hat einen neuen Blumenkasten, mit Platz für 3x3 Blumen.
Dafür sucht er die perfekte Bepflanzung.

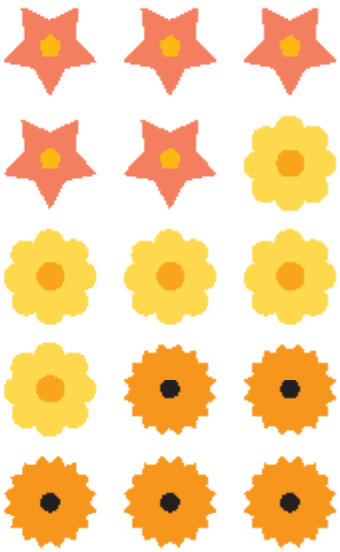
Um eine Bepflanzung zu bewerten, schaut Peter,
welche Blumenfarben direkt nebeneinander sind (gerade, nicht schräg).
Dabei vergibt er Punkte:

- Rot neben Gelb gibt 3 Punkte.
- Gelb neben Orange gibt 1 Punkt.
- Ansonsten gibt es keine Punkte.

Am Ende zählt Peter alle Punkte zusammen.
Die Bepflanzung ist perfekt, wenn sie so viele Punkte bekommt wie möglich.
Außerdem muss jede Blumenfarbe mindestens einmal vorkommen.

Finde die perfekte Bepflanzung!





So ist es richtig:
Ben vergibt Punkte für benachbarte Blumen-Paare. Jede Linie im Blumenkasten-Bild steht für ein solches Paar. Eine Bepflanzung bekommt möglichst viele Punkte (und ist dann perfekt), wenn so viele Paare wie möglich aus einer roten und einer gelben Blume bestehen. Es muss aber auch mindestens eine orange Blume vorkommen. Um auch mit einer orangen Blumen Punkte zu machen, muss sie mit gelben Blumen Paare bilden. Es sollte aber nur so wenige orange-gelbe Paare geben wie möglich, damit so viele Linien bzw. Paare für die Kombination rot-gelb übrig bleiben. Das kann man erreichen, wenn man eine einzige orange Blume in eine Ecke platziert – egal in welche. Dann kann sie nur an zwei Paaren beteiligt sein; weniger geht nicht. Neben die orange Blume werden dann gelbe Blumen gesetzt und dann abwechselnd rote und gelbe, wie im Bild. Diese Bepflanzung ist perfekt, denn mehr Punkte kann eine Bepflanzung nicht bekommen.

Das ist Informatik!
Das Bestreben nach *Optimierung* hat Menschen schon immer ausgezeichnet: der Wunsch, für ein Problem nicht nur irgendeine Lösung zu finden, sondern die beste. Optimierung ist alltäglich: Wer einen bestimmten Computer kaufen will, sucht nach dem Angebot mit dem niedrigsten Preis. Wer von A nach B reisen will, möchte nicht irgendeine Route nehmen, sondern die günstigste. Aber was bedeutet „günstig“ in diesem Fall: Soll die Route möglichst geringe Kosten verursachen, eine möglichst geringe Länge in km haben oder möglichst wenig Zeit in Anspruch nehmen?

Wenn man ein Optimierungsproblem lösen will, benötigt man also nicht nur die Kenntnis, wie eine Lösung des Problems überhaupt aussieht. Man braucht euch eine Funktion, die den Wert einer Lösung berechnen kann. Eine optimale Lösung (oder auch: ein Optimum) ist dann eine Lösung mit dem höchsten Wert – oder mit dem niedrigsten Wert, je nach Funktion. In dieser Biberaufgabe ist Bens Methode zur Berechnung eines Punktwerts für ein Blumenarrangement eine solche Optimierungsfunktion.

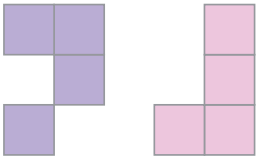
In der Mathematik wird Optimierung seit Jahrhunderten untersucht. In der Informatik wurde viel Sorgfalt darauf verwandt, mathematische Optimierungsmethoden als Algorithmen zu implementieren: Computer können für die Optimierung bei groß angelegten Problemen eingesetzt werden, bei denen Menschen keinen Erfolg hätten. Es ist bekannt, dass viele Optimierungsprobleme sehr schwer zu lösen sind. Informatikerinnen und Informatiker haben deshalb für solche Probleme Methoden (sogenannte Heuristiken) erfunden und untersucht, die nicht immer das Optimum, aber Lösungen nahe am Optimum finden.

Bominos

Bominos sind 2D-Figuren.
Ein Bomino besteht aus farbigen Quadraten, die in einem Raster liegen.
Die Quadrate müssen folgende Bedingungen erfüllen:

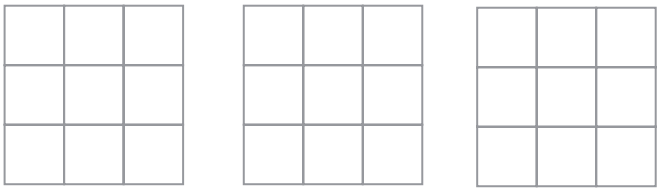
1. Jedes Quadrat berührt mindestens ein anderes Quadrat (oben, unten, links, rechts oder diagonal).
2. Für mindestens ein Paar von Quadraten des Bominos gilt:
   - Die beiden Quadrate berühren sich diagonal.
   - Kein weiteres Quadrat des Bominos berührt gleichzeitig diese beiden Quadrate.

Ein Bomino mit n Quadraten heißt auch n-Bomino. Einige Beispiele:

	Dies ist das einzig mögliche Bomino aus 2 Quadraten (also: 2-Bomino).
	Die linke Figur ist ein 4-Bomino. Die rechte Figur ist kein Bomino; Bedingung 2 ist nicht erfüllt.

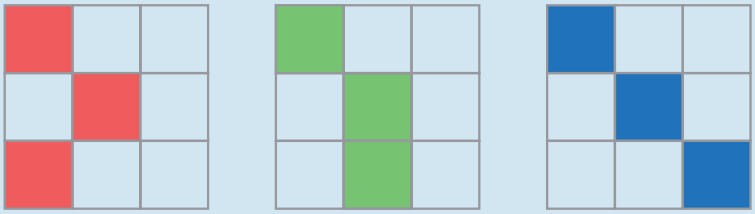
Zwei Bominos gelten als gleich, wenn das eine durch Drehen oder Spiegeln (oder beides) aus dem anderen entsteht.

Es gibt drei verschiedene 3-Bominos. Zeichne sie in die Raster.
Jedes 3-Bomino passt in ein Raster.



So ist es richtig:

Dies sind die drei verschiedenen 3-Bominos:



So sind sie entstanden: Jedes Bomino muss das oben gezeigte 2-Bomino enthalten, also ein Paar von Quadraten, die sich diagonal berühren, aber von keinem weiteren Quadrat gleichzeitig berührt werden. Deshalb kann zunächst das 2-Bomino wie in der Aufgabe gezeigt in alle drei Raster gezeichnet werden, zum Beispiel oben links. Das dritte Quadrat darf nur eines der ersten beiden berühren. Wir wählen das untere, in der Mitte des Rasters, und fügen das dritte Quadrat daran jeweils an einer anderen Position an, in der unteren Zeile: links, mitte und rechts.
Es ist egal, ob man das 2-Bomino anders oder an anderer Position ins Raster malt oder ob man das dritte Quadrat an das andere Quadrat des 2-Bomino oder auf andere Weise anfügt (z.B. in der rechten Spalte): Alle resultierenden Figuren entstehen durch Drehung oder Spiegelung aus einer der oben gezeigten Figuren, sind also nicht von ihnen verschieden.

Das ist Informatik!
Die Bominos in dieser Biberaufgabe werden durch eine genaue Beschreibung ihrer Eigenschaften eingeführt. In der Mathematik nennt man das eine *Definition,* in der Informatik oft auch eine *deklarative* Beschreibung. Die Definition einer Art von Objekten (hier: der Bominos) anhand ihrer grundlegenden Eigenschaften ist Voraussetzung dafür, diese Eigenschaften genauer zu untersuchen und möglicherweise weitere Eigenschaften der Objekte abzuleiten.

Wenn man aber Objekte einer bestimmten Art konstruieren will oder prüfen will, ob ein gegebenes Objekt zu dieser Art gehört, genügt eine deklarative Beschreibung nicht. Erforderlich ist dann eine *operationale* Beschreibung oder auch eine *Operationalisierung* der Beschreibung, z.B. als Computerprogramm, das die oben genannte Konstruktions- oder Prüfungsaufgabe erledigt.

Ein Beispiel: Eine formale Grammatik beschreibt deklarativ, wie gültige Programme einer Programmiersprache aufgebaut sein müssen. Ein Compiler für diese Programmiersprache ist eine Operationalisierung dieser Beschreibung und prüft (unter anderem), ob ein gegebener Quelltext ein gültiges Programm dieser Sprache ist.

Es ist häufig schwierig, eine deklarative Beschreibung zu operationalisieren. In der Informatik wurden deshalb verschiedene Methoden entwickelt, mit denen deklarative Beschreibungen automatisch in operationale Beschreibungen übersetzt werden können.

Übrigens: Bominos sind eine besondere Biber-Variante der unter anderem aus dem Spiel Tetris bekannten Polyominos. Grob gesagt sind *n*-Bominos alle *n*-Pseudo-Polyominos ohne alle *n*-Polyominos. Interessierte können in der englischen Wikipedia mehr erfahren:

https://en.wikipedia.org/wiki/Polyomino
https://en.wikipedia.org/wiki/Pseudo-polyomino

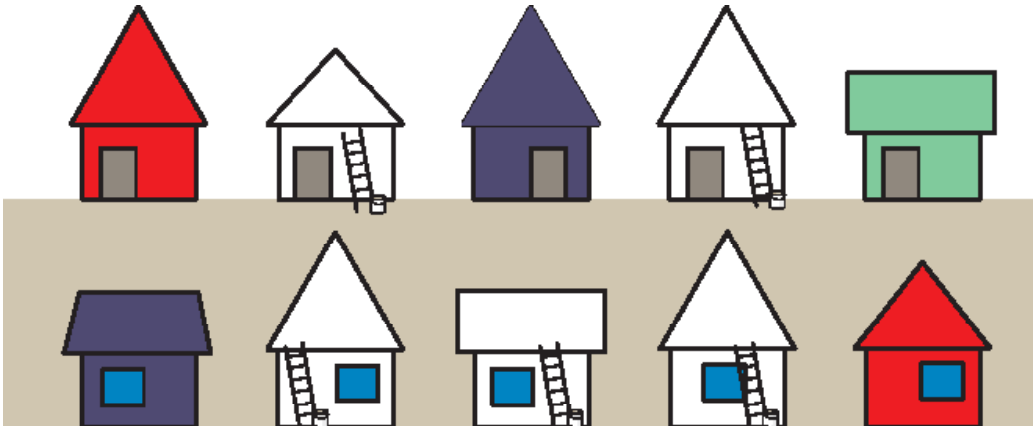
Bunte Straße

An einer Straße sollen alle Häuser bunt angestrichen werden: in Rot, Blau, oder Grün. Damit es nicht langweilig aussieht, gibt es diese Regeln:

- Zwei Häuser, die auf einer Straßenseite direkt nebeneinander stehen, dürfen nicht dieselbe Farbe haben.
- Zwei Häuser, die sich auf zwei Straßenseiten direkt gegenüber stehen, dürfen nicht dieselbe Farbe haben.

Einige Häuser sind schon fertig.

Streiche auch die weißen Häuser nach den Regeln an!

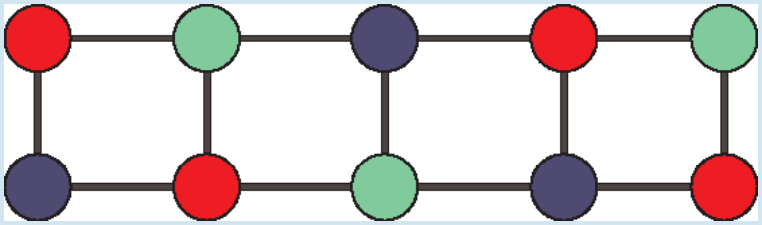


So ist es richtig:
Die Farben der weißen Häuser lassen sich schrittweise bestimmen. Beide Häuser auf der oberen Straßenseite stehen direkt neben Häusern mit verschiedenen Farben. Für den Anstrich der Häuser bleibt also jeweils nur die dritte Farbe übrig, einmal Grün und einmal Rot:

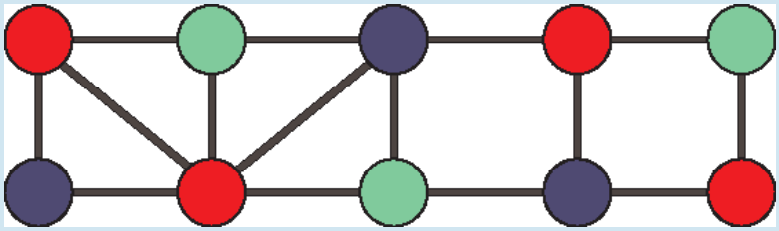
Auf der unteren Straßenseite steht nun das linke weiße Haus neben einem blauen Haus und einem grünen Haus gegenüber. Es muss also rot angestrichen werden.

Das weiße Haus rechts daneben steht damit nun neben einem roten Haus und einem blauen Haus gegenüber. Es muss also grün angestrichen werden. Danach steht auch die Farbe des letzten weißen Hauses fest: Blau.

Das ist Informatik!
Die Häuser dieser Aufgabe und deren direkte Nachbarschaften (sowohl nach links und nach rechts als auch quer über die Strasse hinweg) können als ein *Graph* modelliert werden. Dabei ist jedes Haus ein Knoten (gezeichnet als Kreis), und jede direkte Nachbarschaft ist eine *Kante* (gezeichnet als Linie). Jeder Knoten ist so gefärbt, wie das zugehörige Haus am Ende angemalt ist.



Die Regeln in dieser Biberaufgabe bewirken, dass in diesem „Häuser-Graph" zwei Knoten, die durch eine Kante verbunden sind, nicht gleich gefärbt sind. Würde man zwei Kanten hinzufügen, wäre dies nicht mehr möglich, wir bräuchten eine vierte Farbe:



Wie viele Farben braucht man dann höchstens, um in beliebigen Graphen die Knoten so einzufärben, dass zwei benachbarte (also durch eine Kante verbundene) Knoten nicht dieselbe Farbe haben?
In der Informatik ist bekannt: Vier Farben genügen, wenn der Graph planar ist, das heißt, wenn er keine Kanten hat, die sich kreuzen. Dieser Satz ist als Vier-Farben-Satz oder auch Vier-Farben-Theorem bekannt.

Es ist nicht einfach einzusehen, dass vier Farben bei allen planaren Graphen ausreichen. Erst 1976 konnten die Mathematiker Kenneth Appel und Wolfgang Haken das beweisen. Dazu haben sie Computer verwendet, um eine Vielzahl von Ausnahmen und Gegenbeispielen zu überprüfen. Andere Menschen können das ohne Computer nicht nachvollziehen. Deshalb gibt es andere Mathematiker, die den Computereinsatz bei diesem Beweis oder allgemein beim Beweisen kritisch sehen.

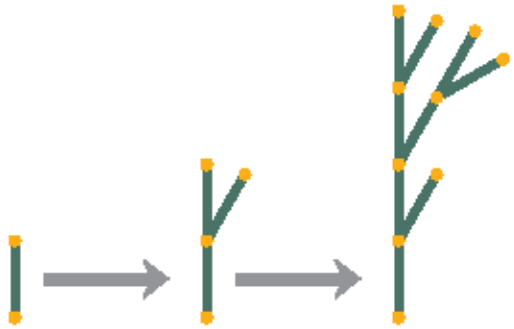
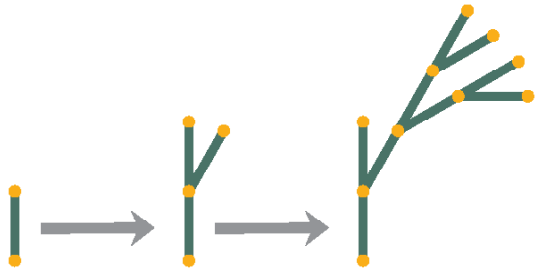
Der Vier-Farben-Satz wird in vielen Computerprogrammen angewendet: beispielsweise um Flugpläne zu erstellen, bei denen Flugzeuge Korridoren zugeteilt werden, damit sie immer genügend Abstand haben, oder auch um Frequenzbereiche von Sendemasten zuzuteilen, so dass diese sich nicht stören und der Empfang trotz vieler Sendemasten nicht schlechter wird.

Digitale Bäume

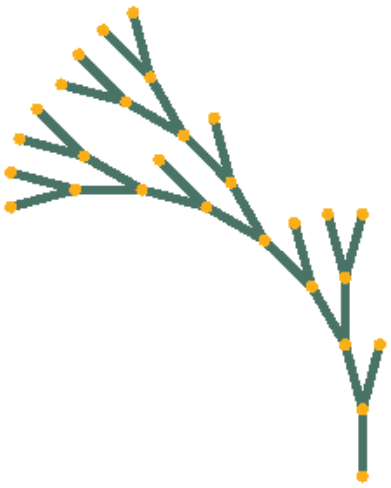
Digitale Bäume wachsen schrittweise, nach vorgege en Regeln.

Ein digitaler Baum besteht zuerst aus einem Stück: 

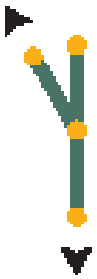
Eine Wachstumsregel gibt an, wie ein Baumstück du eine Struktur von Stücken ersetzt wird. In jedem Wachstumsschritt wird die Re gleichzeitig auf jedes Baumstück angewendet. Zwei Beispiele:

Regel	Die ersten zwei Schritte	Regel	Die ersten zwei Schritte
			

Die Pfeilspitzen geben an, wo und in welche Richtung dabei die Strukturen aneinander gesetzt werden.
Hier ist ein digitaler Baum. Er ist in 4 Schritten gewachsen.
Unten siehst du verschiedene Wachstumsregeln.

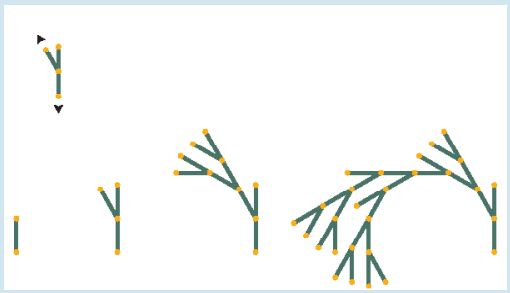
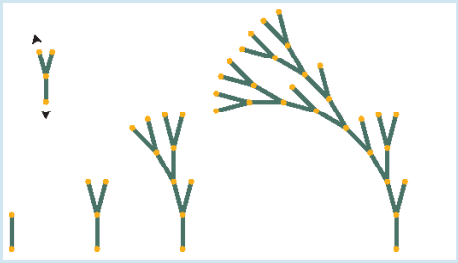
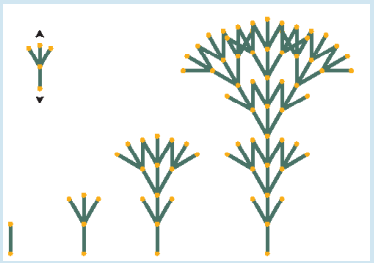
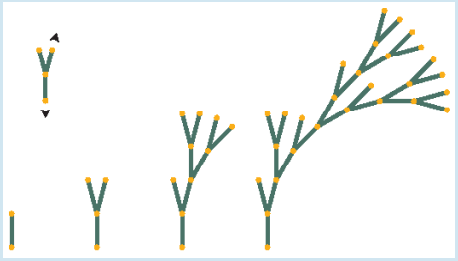


Nach welcher Regel ist der Baum gewachsen?

		C	D
			

Antwort B ist richtig:
Regel A: Der Rest des Baumes wird stets am nach links zeigenden Zweig angefügt. Der Baum krümmt sich deshalb stark nach links.
Regel B: Der Rest des Baumes wird stets am linken Abzweig angefügt; dieser neigt sich aber weniger stark nach links als der Zweig in Regel A. Der Baum neigt sich deshalb nach links.
Regel C: Der Rest des Baumes wird stets in der Mitte angefügt, in gerader Richtung. Durch die beiden Abzweige links und rechts bildet sich insgesamt eine gleichmäßige, symmetrische Struktur.
Regel D ist die horizontale Spiegelung von Regel B. Der Rest des Baumes wird stets am rechten Abzweig angefügt. Der Baum neigt sich deshalb nach rechts.

Hier sieht man, wie die digitalen Bäume nach den verschiedenen Regeln wachsen.

A	B
	
C	**D**
	

Das ist Informatik!
An den digitalen Bäumen in dieser Biberaufgabe kann man sehen, wie durch die wiederholte Anwendung einer einfachen Wachstumsregel in wenigen Schritten komplexe Figuren entstehen können. Diese Idee findet sich in den so genannten “Lindenmayer-Systemen” wieder, die kurz auch L-Systeme genannt werden. Der Biologe Aristid Lindenmayer hat sie erfunden, um das Wachstum von Pflanzen zu beschreiben.

L-Systeme können relativ einfach in Computerprogramme umgesetzt werden. Deshalb sind sie auch in der Informatik gut bekannt und werden zum Beispiel im Bereich der Computergrafik genutzt, um für Bilder oder auch Filme sehr wirklichkeitsnahe Darstellungen von Pflanzen zu erzeugen. L-Systeme verwenden das Prinzip der Rekursion, um mit einfachen Beschreibungen komplexe Strukturen entstehen zu lassen (siehe auch die Aufgaben „Sierpinski-Dreieck“ und „Zahlenmaschine“ in diesem Biberheft).

Durch den Tunnel

Anna und Benno machen mit ihren Eltern eine Wanderung. Auf ihrer Strecke liegt ein Tunnel. Aus Erfahrung wissen sie, dass jeder von ihnen unterschiedlich viel Zeit für die Tunnelpassage benötigt: Anna benötigt 10 Minuten, Benno 5 Minuten, die Mutter 20 Minuten und der Vater 25 Minuten.

Den dunklen und engen Tunnel kann man nur alleine oder zu zweit passieren. Sie müssen also mehrere Passagen machen. Zu zweit benötigt man so viel Zeit wie die langsamere der beiden Personen. Im Tunnel muss man auf jeden Fall eine Lampe benutzen.

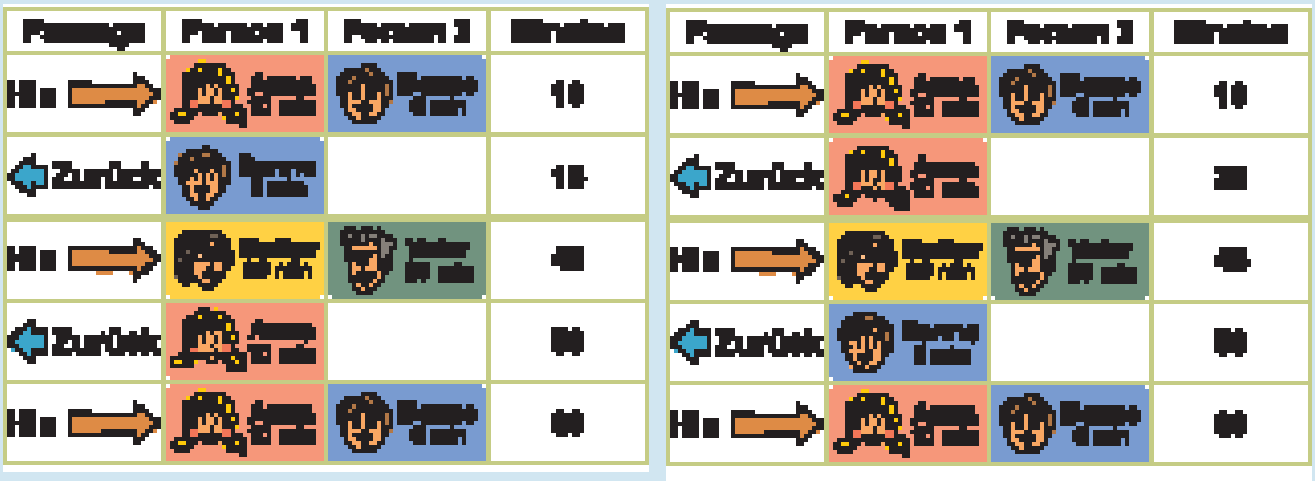
Als sie an den Eingang des Tunnels kommen, stellen sie fest: Der Akku ihrer einzigen Lampe reicht nur noch für 60 Minuten. Können sie innerhalb dieser 60 Minuten alle durch den Tunnel kommen?

Anna hat einen Plan: „Ja, können wir, und zwar mit fünf Passagen!“

Ziehe die Namen so in die passenden Felder, das Annas Plan umgesetzt wird.

Hin →		
← Zurück		
Hin →		
← Zurück		
Hin →		

So ist es richtig:
Dass alle innerhalb von 60 Minuten am Ausgang sind, ist nur möglich, wenn Mutter und Vater (die beiden langsamsten) nur einmal den Tunnel passieren. Das wiederum ist nur möglich, wenn vorher Anna und Benno den Tunnel passieren und einer von ihnen zurück geht. Der eine kann dann die Lampe Vater und Mutter geben, und der andere bringt sie durch den Tunnel zurück, damit dann beide rechtzeitig zusammen zum Ausgang gehen können. Dabei ist es egal, ob in Passage 2 Anna zurückgeht oder Benno: In Passage 4 ist der jeweils andere dran, so dass diese beiden Passagen in beiden Fällen zusammen 15 Minuten dauern.



Das ist Informatik!
Anna hat die Lösung des Tunnelproblems mit einem Plan beschrieben. Ihr Plan erfüllt die Bedingung, dass bei jeder Passage die Lampe dabei ist. Die Ausführung des Plans kommt mit den verfügbaren Mitteln aus: der 60-Minuten Akku-Reserve der Lampe.

Die Informatik entwickelt immer komplexere Systeme, die in der Lage sind, eigenes Handeln selbständig bzw. autonom zu planen, dabei die vorgegebenen Bedingungen zu erfüllen und mit den verfügbaren Mitteln auszukommen; so wie Anna. Die Konsequenzen autonomen Planens sind weitreichender, als in unserer gut überschaubaren Biberaufgabe. Zum Beispiel bei selbstfahrenden Autos: Die müssen einen sinnvollen Weg zum Ziel planen, dabei alle Verkehrsregeln einhalten und berücksichtigen, dass sie während der Ausführung immer genug Energie haben.

Souveräne und sichere Informatik-Planung wird auch ethisch und juristisch vertretbar sein müssen, wenn das Handeln autonomer Systeme Auswirkungen für Menschen, andere Lebewesen und die Umwelt hat. Ein selbstfahrendes Auto muss stets die zentrale Bedingung erfüllen, dass durch sein Handeln kein Mensch zu Schaden kommt – wenn irgend möglich. Solche Überlegungen hat Isaac Asimov bereits im Jahr 1942 angestellt, als gerade die ersten Computer gebaut wurden. Er hat sich Gesetze für Roboter überlegt. Sein erstes Robotergesetz lautet: „Ein Roboter darf keinen Menschen verletzen oder durch Untätigkeit zu Schaden kommen lassen." Und ein selbstfahrendes Auto ist ein Roboter.

Flaggenbilder

Computerbilder bestehen aus Zeilen von Bildpunkten, genannt Pixel. Wenn Computerbilder als Dateien gespeichert werden, wird im einfachsten Fall die Farbe jedes Pixels einzeln beschrieben.

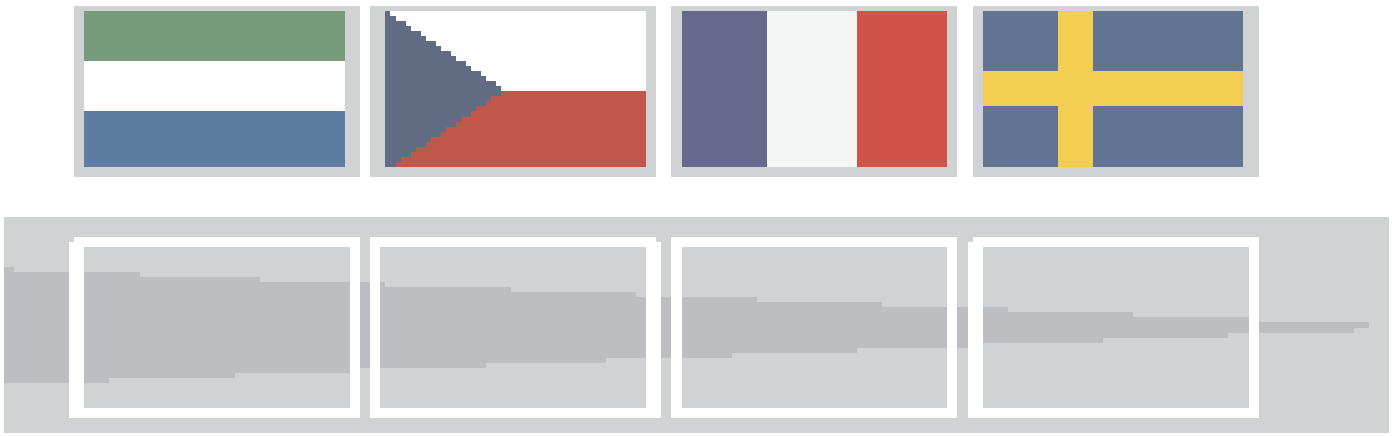
Mit dem (erfundenen) Dateiformat GIW werden Computerbilder komprimiert, also mit geringerer Dateigröße gespeichert. Das funktioniert so:

- Jede Pixelzeile wird einzeln beschrieben.
- Jede Farbe wird durch ein Kürzel aus drei Buchstaben beschrieben.
- Eine Folge gleichfarbiger Pixel wird durch ein Klammerpaar beschrieben, das ein Farbkürzel und die Anzahl der gleichfarbigen Pixel enthält.

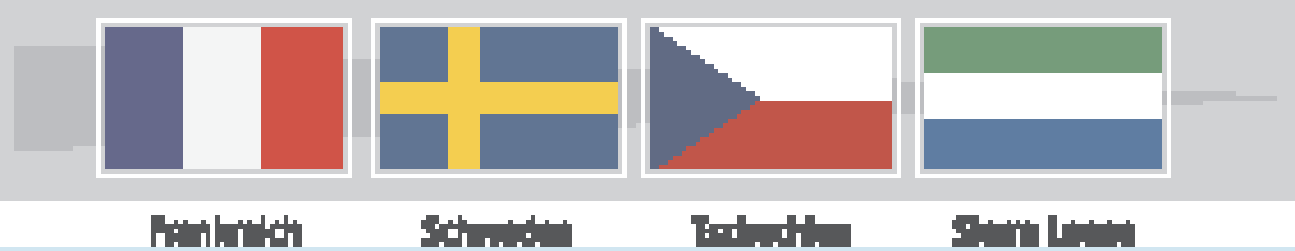
Ein Beispiel: Eine Pixelzeile, die durch die beiden Klammerpaare (grü,20)(wei,13) beschrieben wird, enthält zuerst 20 grüne und danach 13 weiße Pixel.

Unten siehst du vier Computerbilder von Flaggen. Die Bilder bestehen alle aus gleich vielen Pixelzeilen mit jeweils gleich vielen Pixeln. Sie wurden als Dateien im GIW-Format gespeichert. Die Größe einer GIW-Datei ergibt sich aus der Anzahl der darin enthaltenen Klammerpaare.

Ordne die Bilder nach der Größe ihrer GIW-Datei!



So ist es richtig:



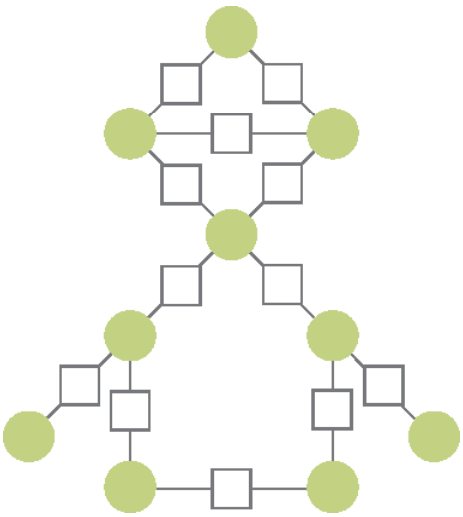
Eine Zeile, deren Pixel alle die gleiche Farbe haben, kann mit einem einzigen Klammerpaar beschrieben werden. Bei der Flagge von Sierra Leone ist das in allen Zeilen so. Die GIW-Datei enthält also ein Klammerpaar pro Zeile.

In Bildern, bei denen die Farben in einer Pixelzeile wechseln, wird nach jedem Farbwechsel ein neues Klammerpaar benötigt. Bei der Flagge von Tschechien gibt es in jeder Zeile genau einen Farbwechsel, von Blau nach Weiß bzw. Blau nach Rot. Die GIW-Datei enthält also zwei Klammerpaare pro Zeile.
Bei der schwedischen Flagge gibt es in den meisten Zeilen zwei Farbwechsel. Die GIW-Datei enthält also für diese Zeilen je drei Klammerpaare. In der Mitte sind einige Zeilen, die keinen Farbwechsel enthalten; für jede dieser Zeilen enthält die Datei nur ein Klammerpaar. Es gibt aber viel mehr Zeilen mit zwei Farbwechseln als Zeilen ohne Farbwechsel. Deshalb enthält die GIW-Datei im Mittel mehr als zwei, aber weniger als drei Klammerpaare pro Zeile.

Bei der französischen Flagge gibt es in jeder Zeile zwei Farbwechsel. Die GIW-Datei enthält also drei Klammerpaare pro Zeile.

Das ist Informatik!
Datenkompression ist ein wichtiger Bereich der Informatik. Je geringer die Datenmenge ist, mit der digitale Daten (Bilder, Audiodaten, Videos, …) beschrieben werden können, desto geringer ist der zur Speicherung der Daten benötigte Platz, und desto schneller lassen sich die Daten durch ein Netzwerk transportieren. Kompressionsverfahren können den Transferaufwand innerhalb eines Netzwerks also deutlich verringern. Würde beispielsweise die Musik eines Webradios ohne Kompression versendet, so würde die zehnfache Datenmenge anfallen als bei heutzutage typischer Kompression. Da die Menge der Daten, die durch Netzwerke transportiert werden sollen, immer weiter steigt (zum Beispiel Sprachnachrichten in Messenger-Anwendungen), forscht die Informatik weiter im Bereich der Datenkompression. Das in dieser Aufgabe vorgestellte Kompressionsverfahren arbeitet nach dem Prinzip der Lauflängenkodierung. Weitere Informationen kann das folgende Video liefern: https://www.youtube.com/watch?v=ypdNscvym_E

Geschäfte

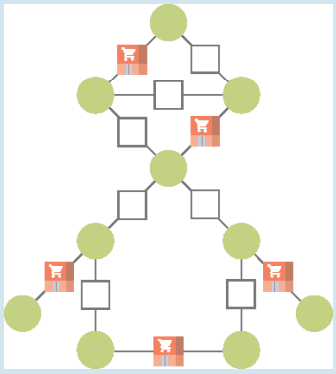


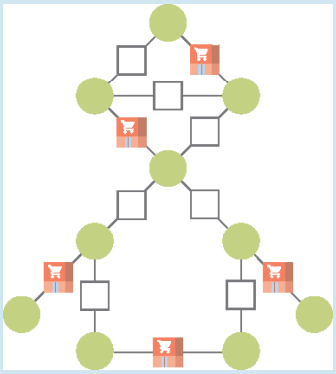
Eine Landgemeinde will die Versorgung ihrer Dörfer verbessern und dazu Geschäfte bauen.

Die Karte rechts zeigt die Dörfer als grüne Punkte. An den Straßen zwischen den Dörfern (Linien) liegen Bauplätze für Geschäfte (Quadrate). Von jedem Dorf aus soll man ein Geschäft erreichen können, ohne durch ein anderes Dorf zu fahren. Diese Bedingung muss mit so wenig Geschäften wie möglich erfüllt werden.

Auf welchen Bauplätzen müssen Geschäfte gebaut werden?

So ist es richtig:





Wir sagen, dass ein Bauplatz bzw. ein darauf gebautes Geschäft ein Dorf versorgt, wenn der Bauplatz vom Dorf aus direkt erreicht werden kann, ohne durch ein anderes Dorf zu fahren. An den Straßen ganz links und rechts müssen Geschäfte sein, um die Dörfer ganz außen zu versorgen. Um die beiden Dörfer unten zu versorgen, reicht es, wenn dazwischen ein Geschäft gebaut wird. Damit haben wir drei Bauplätze ausgewählt, um die unteren sechs Dörfer zu versorgen. Weil ein Geschäft höchstens zwei Dörfer versorgen kann, werden mindestens zwei weitere Geschäfte benötigt, um auch die oberen vier Dörfer zu versorgen. Die Bilder zeigen die beiden einzigen Möglichkeiten, die oberen Dörfer mit nur zwei Geschäften zu versorgen. Lässt man nur eines der fünf Geschäfte weg, sind nicht mehr alle Dörfer versorgt. Mit weniger Geschäften geht es also nicht.

Das ist Informatik!

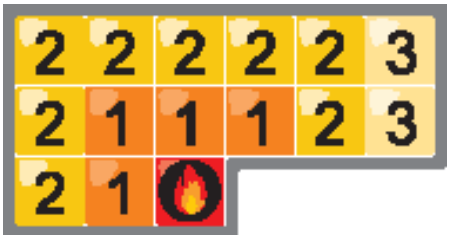
Die Karte ist ein vereinfachtes Modell der Wirklichkeit und konzentriert sich auf die Information, die für die Auswahl der Bauplätze wichtig ist: Sind zwei Dörfer durch eine Straße verbunden oder nicht? Um Beziehungen zwischen Objekten zu modellieren, werden in der Informatik häufig verwendet. Ein Graph besteht aus Knoten und Kanten. Die Knoten stehen für Objekte, und zwischen zwei Knoten gibt es eine Kante, wenn die beiden durch die Knoten modellierten Objekte miteinander in einer bestimmten Beziehung stehen. Die Karte der Dörfer und Straßen ist also die Zeichnung eines Graphen; dessen Knoten (gezeichnet als grüne Punkte) für die Dörfer stehen und dessen Kanten (gezeichnet als Linien) dafür stehen, dass zwei Dörfer durch eine Straße miteinander verbunden sind. Die Modellierung mit Graphen ist in der Informatik sehr verbreitet. Die Informatik kennt nämlich viele Verfahren, um Fragen, die man in Bezug auf Graphen stellen kann, zu beantworten. Die in dieser Biberaufgabe gestellte Frage nach den Bauplätzen kennt die Informatik als “Problem der minimalen Kantenabdeckung” (englisch: minimum edge cover): Die Herausforderung ist, aus den Kanten eines Graphen möglichst wenige Kanten so auszuwählen, dass von jedem Knoten eine der gewählten Kanten ausgeht. Eine minimale Kantenabdeckung lässt sich auch für große Graphen schnell bestimmen.

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

Hotspot-Heizung

Luis mag es warm im Bad. In sein neues Haus lässt er eine Bodenheizung mit Hotspots einbauen.

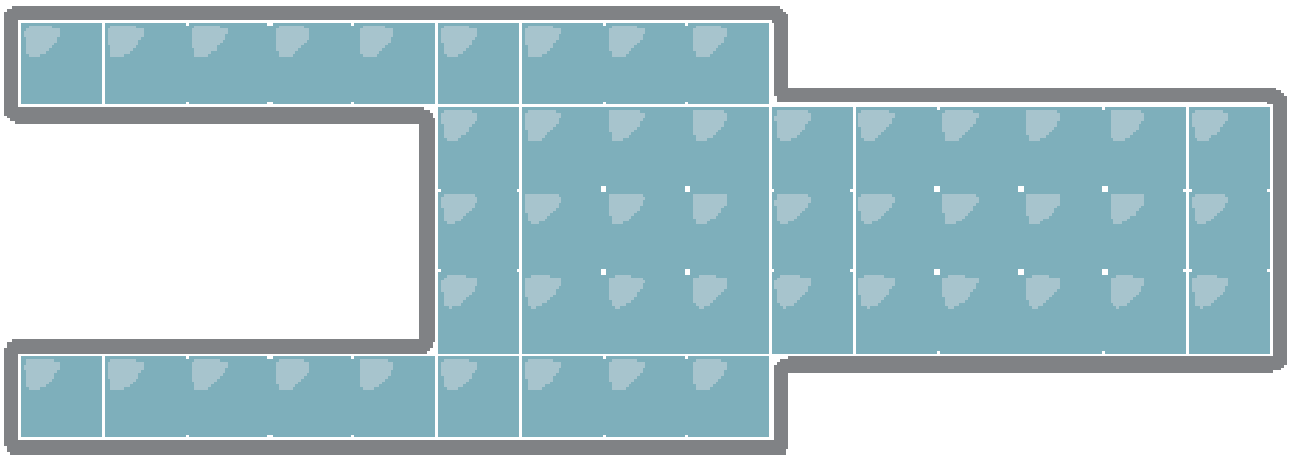
Ein Hotspot wird direkt unter einer Fliese montiert.
Schaltet man den Hotspot ein, wird diese Fliese sofort warm. Von einer warmen Fliese breitet sich die Wärme in einer Minute auf alle benachbarten Fliesen aus – seitlich und über Eck.



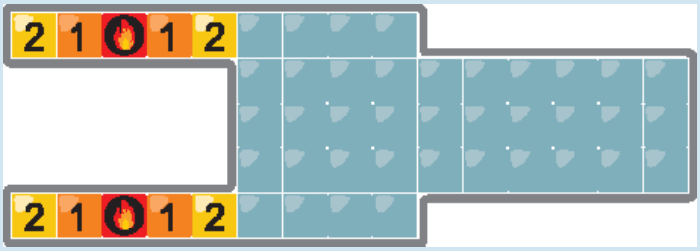
Hier ist ein Beispiel. Die Zahlen sagen für jede Fliese, nach wie vielen Minuten sie warm ist.

Luis kann sich 4 Hotspots leisten.
Luis wünscht, dass nach möglichst wenigen Minuten alle Fliesen im Bad warm sind.
Die Hotspots werden gleichzeitig angeschaltet.

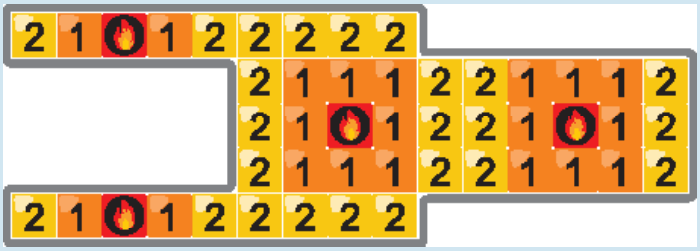
Hier ist das neue Bad.
Montiere die 4 Hotspots so, dass Luis' Wunsch erfüllt wird.



So ist es richtig:
Man kann die 4 Hotspots so montieren, dass alle Fliesen des Badezimmers nach 2 Minuten warm sind. Jeder Hotspot kann in der ersten Minute höchstens 9 Fliesen erwärmen, nach 2 Minuten höchstens 25 Fliesen. Vier Hotspots erwärmen in der ersten Minute höchstens 36 Fliesen, in 2 Minuten höchstens 100 Fliesen. Das Badezimmer hat 48 Fliesen. Es ist also unmöglich, mit vier Hotspots alle Fliesen in nur einer Minute zu erwärmen, aber 2 Minuten könnten ausreichen.
Nun müssen die Hotspots so montiert werden, dass alle Fliesen nach 2 Minuten warm sind. Wegen der schlechten Aufteilung des Raumes bietet es sich an, zwei Hotspots so zu montieren, dass die Fliesen in den beiden Gängen nach 2 Minuten warm sind:



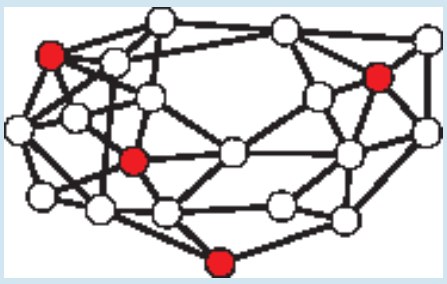
Die verbleibenden zwei Hotspots kann man dann so montieren:



Den ganz rechts montierten Hotspot kann man auch eine Fliese darüber oder darunter montieren. Auch dann sind nach 2 Minuten alle Fliesen warm.

Das ist Informatik!
Das möchte man häufig: mit möglichst wenig Aufwand möglichst viel erreichen. In dieser Biberaufgabe soll mit möglichst wenig Heizungen das Bad möglichst schnell warm werden. Will man ein großes Haus mit WLAN ausstatten, möchte man mit möglichst wenigen Hotspots eine gute Abdeckung aller Räume erzielen. Es gibt noch viele andere Beispiele.
In der Informatik ist ein ähnliches Problem bekannt, nämlich die Bestimmung eines minimalen „Dominating Set“ in einem Graphen. Ein Graph ist eine Struktur aus Knoten und Kanten (das sind Paare von Knoten). Die beiden Knoten einer Kante nennt man auch Nachbarn. Ein Dominating Set (DS) ist eine Teilmenge der Knoten, für die gilt: Jeder andere Knoten muss einen Nachbarn haben, der im Dominating Set enthalten ist. Ein DS ist minimal, wenn es keine kleinere Knoten-Teilmenge gibt, die auch ein DS ist.

Hier ist ein Beispiel-Graph: Die Knoten sind als Kreise, die Kanten als Linien gezeichnet. Jeder weiße Knoten ist Nachbar mindestens eines roten Knotens. Also sind die roten Knoten ein DS, und zwar ein minimaler: Es gibt kein DS mit weniger als vier Knoten.
Die Platzierung der Heizungen kann man als Bestimmen eines minimalen DS verstehen: Die Fliesen sind die Knoten. Zwei Fliesen sind „Heizungs-Nachbarn“ (bilden also eine Kante), wenn das Montieren einer Heizung unter einer Fliese bedeutet, dass die andere nach höchstens zwei Minuten warm ist. Fliesen, die in diesem Graphen ein minimales DS bilden, stellen eine Lösung dieser Biberaufgabe dar.



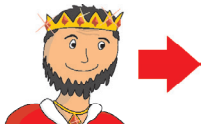
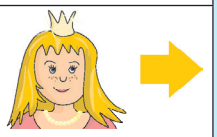
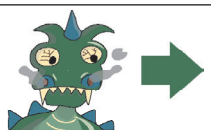
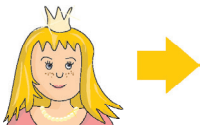
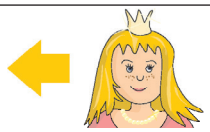
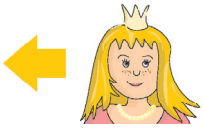
Im Theater

Im Theater spielen heute diese Figuren:

die Prinzessin , der Ritter , der König und der Drache .

Es geht los! Am Anfang ist niemand zu sehen. Dann kommen und gehen die Figuren so:

ERSTER AKT			ZWEITER AKT	
König kommt			Drache kommt	
Prinzessin kommt			Ritter kommt	
König geht			Drache geht	
Drache kommt			Prinzessin kommt	
Prinzessin geht			Ritter geht	
Drache geht			Prinzessin geht	
		Pause	Ende	

Welche Figuren sind NICHT gleichzeitig zu sehen?

A) Prinzessin und Ritter
B) König und Drache
C) König und Prinzessin
D) Ritter und Drache

Antwort B ist richtig:
König und Drache sind bei der Aufführung nicht gleichzeitig zu sehen.
Man kann sich das schrittweise überlegen. In die Tabelle ist eingetragen, wer zu sehen ist, direkt nachdem eine Figur gekommen oder gegangen ist.

Handlung	Teil 1							Pause / Teil 2						Ende
König da?		Ja	Ja	Nein	Nein	Nein	Nein		Nein	Nein	Nein	Nein	Nein	Nein	
Prinzessin da?		Nein	Ja	Ja	Ja	Nein	Nein		Nein	Nein	Nein	Ja	Ja	Nein	
Drache da?		Nein	Nein	Nein	Ja	Ja	Nein		Ja	Ja	Nein	Nein	Nein	Nein	
Ritter da?		Nein	Nein	Nein	Nein	Nein	Nein		Nein	Ja	Ja	Ja	Nein	Nein	
Hinweis			C							D		A			

Zwei Figuren sind gleichzeitig zu sehen, wenn in einer Zeile bei beiden Figuren „Ja“ steht.
Antwort A (Prinzessin und Ritter): Teil 2, Zeile 4
Antwort C (König und Prinzessin): Teil 1, Zeile 2
Antwort D (Ritter und Drache): Teil 2, Zeile 2.
Nur für Antwort B (König und Drache) gibt es keine Zeile, in der für beide Figuren „Ja“ steht.
Also sind König und Drache nicht gleichzeitig zu sehen.

Das ist informatik!
Auch wenn man sich anhand der Auftritte und Abgänge der Figuren lebhaft vorstellen kann, was im Theaterstück passiert, ist in dieser Biberaufgabe für jede Figur zu jedem Zeitpunkt nur eine Frage wichtig: befindet sie sich zu diesem Zeitpunkt auf der Bühne oder nicht? Die Antwort auf diese Frage kann nur „ja“ oder „nein“ lauten. Damit entspricht die in der Antwort enthaltene Information einem Bit, der kleinsten Informationseinheit, die in der Informatik bekannt ist. Ein Bit unterscheidet zwischen genau zwei Zuständen oder Werten: ja oder nein, wahr oder falsch, 0 oder 1 – die Namen der Werte spielen genau genommen keine Rolle.

Das Bit ist auch die kleinste Speichereinheit in Computern. Man kann es sehr einfach und günstig realisieren, etwa als ein Schaltelement, in dem eine hohe und eine niedrige Spannung – den zwei möglichen Werten entsprechend – klar unterschieden und für lange Zeit aufrecht erhalten (also gespeichert) werden können. Mit Bits kann man außerdem rechnen, ähnlich wie mit Zahlen; statt Plus und Minus gibt es dafür Operationen wie „UND“ oder „NICHT“. Die können wir gebrauchen, um in dieser Biberaufgabe die richtige Antwort zu finden: Zu keinem Zeitpunkt waren die Bits von König und Drache gleichzeitig wahr, und damit war auch die Bit-Rechnung „König UND Drache“ zu keinem Zeitpunkt wahr.

Auch für Bit-Operationen können einfache und günstige Schaltungen gebaut werden, insbesondere für die kombinierte Operation „NICHT UND“, Englisch kurz mit „NAND“ bezeichnet. Interessant ist, dass alle Berechnungen, die mit Bits überhaupt möglich sind, alleine mit NANDs realisiert werden können; andere Schaltungen braucht man nicht unbedingt. Das ist ein wichtiger Grund für die Bedeutung der Computer: dass sie aus wenigen einfachen, billigen und heutzutage mikroskopisch kleinen Elementen gebaut werden können.

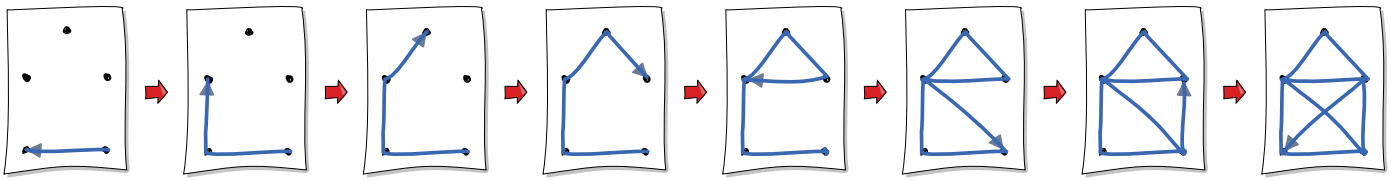
In einem Zug

Eine Strichzeichnung besteht aus Punkten und Strichen.
Jeder Strich verbindet zwei Punkte miteinander.

Manche Strichzeichnungen können in einem Zug gezeichnet werden. Das heißt:

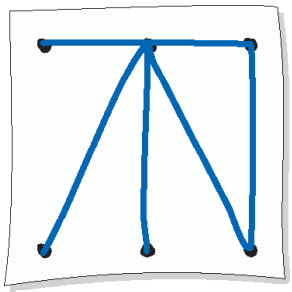
- Der Stift wird erst vom Blatt abgehoben, wenn die Zeichnung fertig ist.
- Jeder Strich wird nur einmal gezeichnet.

Ein Beispiel:

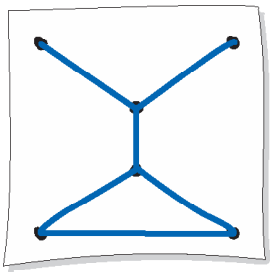


Welche dieser Strichzeichnungen kann in einem Zug gezeichnet werden?

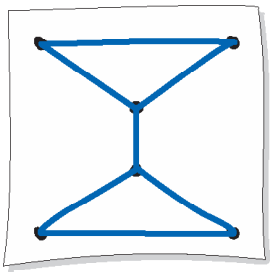
A)



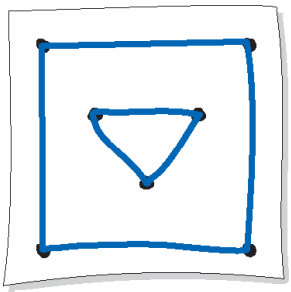
B)



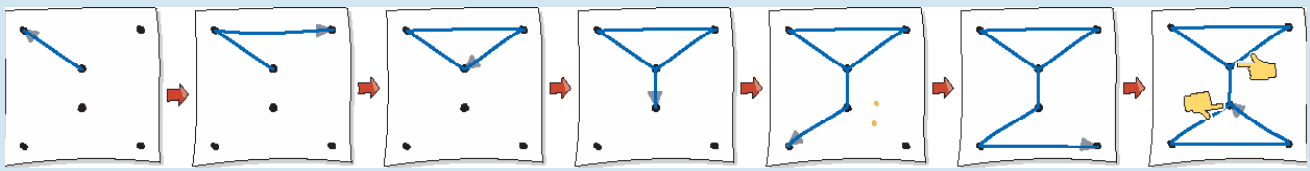
C)



D)

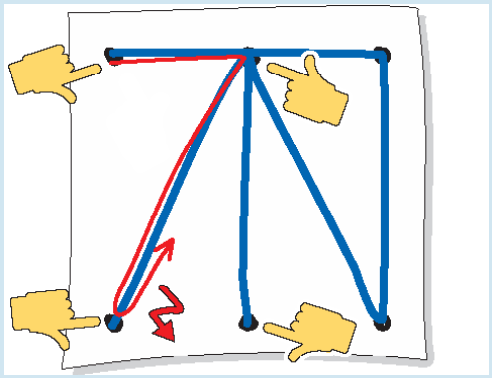


Antwort C ist richtig:
So kann man das Bild in einem Zug zeichnen:

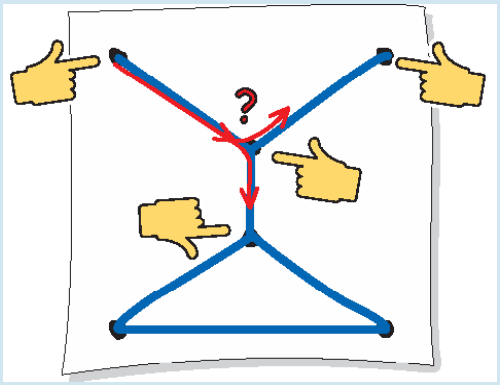


Beachte, dass in genau zwei Punkten dieser Strichzeichnung eine ungerade Zahl von Strichen zusammentreffen, nämlich drei. Diese beiden Punkte sind im Bild rechts markiert; sie sind Anfangs- bzw. Endpunkt beim Zeichnen. In allen anderen Punkten treffen zwei, also eine gerade Anzahl von Strichen zusammen. Das muss bei allen Strichzeichnungen so sein, die man in einem Zug zeichnen kann: Zu jedem Strich, mit dem man bei einem Punkt ankommt, muss es einen Strich geben, mit dem man vom gleichen Punkt aus weiter zeichnet. Kommt man evtl. noch einmal beim gleichen Punkt an, muss es einen anderen Strich geben, mit dem man weiter zeichnet – denn jeder Strich darf ja nur einmal gezeichnet werden. Die bei einem Punkt ankommenden und von ihm wegführenden Striche bilden also Paare, so dass eine gerade Anzahl von Strichen dort zusammentreffen. Die einzigen Ausnahmen sind der Anfangs- und der Endpunkt: Beim Anfangspunkt zeichnet man los, ohne vorher angekommen zu sein. Und beim Endpunkt kommt man an, ohne weiter zeichnen zu müssen.

Bei Antwort A gibt es vier Punkte, an denen eine ungerade Anzahl von Strichen zusammentreffen.



Auch bei Antwort B gibt es vier Punkte, an denen eine ungerade Anzahl von Strichen zusammentreffen. Beim Knoten in der Mitte oben sieht man gut, dass eine ungerade Zahl von zusammentreffenden Strichen das Zeichnen in einem Zug verhindert. Die Zeichnung von Antwort D besteht aus zwei nicht verbundenen Teilen, dem Dreieck innen und dem Quadrat außen. So etwas kann man nur zeichnen, wenn man den Stift mittendrin vom Blatt abhebt.



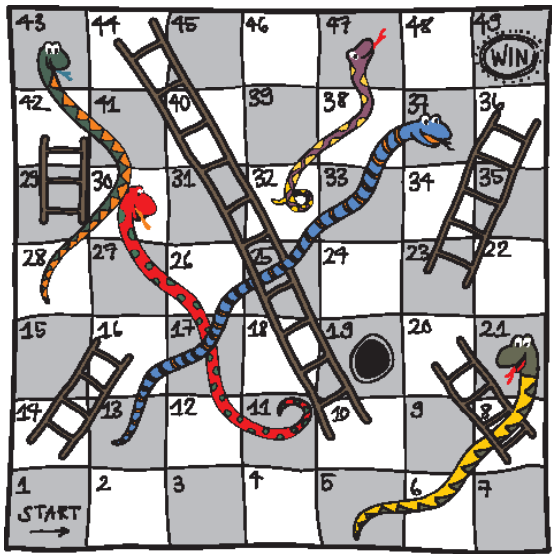
Das ist Informatik!
Die Strichzeichnungen bestehen aus Punkten sowie aus Strichen, die diese Punkte verbinden. Mit solchen Zeichnungen kann man auch einen *Graph* darstellen. Ein Graph besteht aus einer Menge von Knoten (als Punkte dargestellt) und einer Menge von Kanten (Paare von Knoten, als Striche dargestellt). Mit dieser mathematischen Struktur lassen sich Beziehungen zwischen Objekten sehr gut modellieren und weitere Informationen daraus ableiten.

Zum Beispiel stellen die Kanten die direkten Beziehungen zwischen je zwei Objekten dar. Folgen von Kanten, die von einem Startpunkt aus über einen oder mehrere Zwischenpunkte zu einem Endpunkt führen, werden als Pfade bezeichnet und liefern Informationen über indirekte Beziehungen.
Gibt es in einem Graphen einen Pfad, der über jede Kante genau einmal führt, kann man diesen Pfad in einem Zug zeichnen – wie bei Antwort C in dieser Biberaufgabe. Ein solcher Pfad wird als Eulerscher Pfad bezeichnet, benannt nach Leonhard Euler (1707-1783). Euler lebte in Königsberg und versuchte, einen Weg über die sieben Brücken in dieser Stadt zu finden, der über jede Brücke genau einmal führt. Euler stellte fest, dass es in dem Graph, den er aus den Ufern und Brücken bildete, eben keinen Eulerschen Pfad gibt. Seine Überlegungen begründeten das mathematische Gebiet der Graphentheorie.
In der Informatik sind Algorithmen bekannt, mit denen man leicht bestimmen kann, ob es in einem Graph einen Eulerschen Pfad gibt und, wenn ja, einen solchen Pfad finden kann.

Leiterspiel

Beim Leiterspiel starten alle Spieler auf Feld 1. Wer zuerst Feld 49 erreicht, gewinnt.

In jeder Runde würfelt man eine Zahl zwischen 1 und 6 und zieht mit seiner Figur entsprechende viele Felder vor. Endet man dabei auf einem Feld mit dem Kopf einer Schlange, schlittert man hinab bis zum Feld mit ihrem Schwanzende.
Endet man aber am Fuß einer Leiter, darf man diese sofort ganz hinaufklettern.

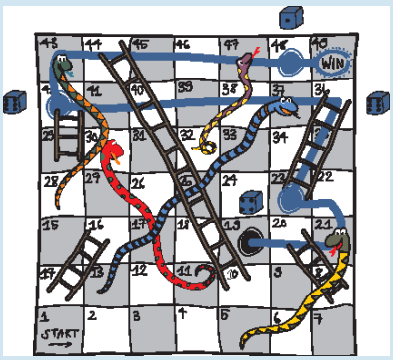


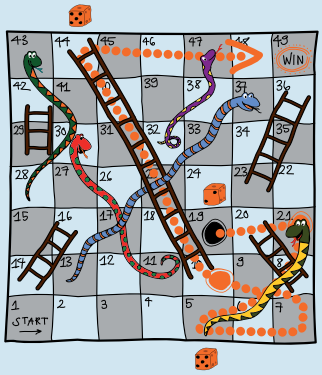
Ein Beispiel: Du stehst auf Feld 26 und würfelst eine 3. Du ziehst also auf Feld 29 und darfst über die Leiter sofort zum Feld 42 vorrücken.
In der nächsten Runde würfelst du eine 5, landest auf dem Schlangenkopf des Feldes 47 und musst sofort zurück zum Feld 32.

Deine Figur steht auf Feld 19.

Wie viele Runden brauchst du mindestens noch, um das Feld 49 zu erreichen?
A) 2 Runden B) 3 Runden C) 4 Runden D) 5 Runden

Antwort B ist richtig:
Wenn du nur Würfe berücksichtigst, mit denen du in Richtung Ziel kommst, brauchst du mindestens 4 Runden: Mit einer 4 kommt man von 19 zu 23 und per Leiter zum Feld 36. Von dort aus gibt es keine weiteren Leitern nach oben. Man braucht weitere 3 Würfe, zum Beispiel 6 – 6 – 1, um das Ziel, also Feld 49 zu erreichen (linkes Bild).





Wenn du aber einen Schritt zurück und damit eine scheinbare Verschlechterung in Kauf nimmst, schaffst du es in 3 Runden, mit den Würfen 2 – 5 – 5. Von der 19 zur 21 und die Schlange hinunter zu Feld 5. Dann zu 10 und über die Leiter hinauf zu 44, und dann ins Ziel (rechtes Bild).
In 2 Runden ist das Ziel nicht zu erreichen. Nur einen Wurf vom Ziel entfernt sind die Felder 48, 46, 45 und 44 (auf 47 kann man nicht bleiben). Keines dieser Felder ist von Feld 19 aus in einer Runde zu erreichen.

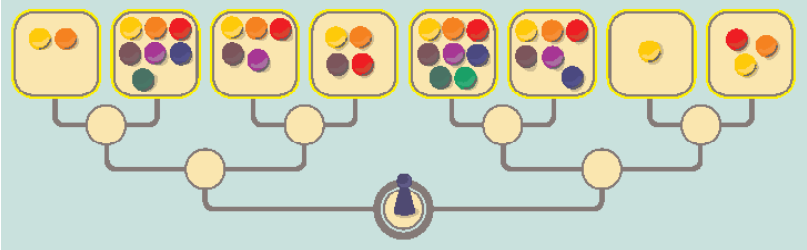
Das ist Informatik!
In dieser Biberaufgabe wurde ein kürzester Weg gesucht, von Feld 19 bis zum Ziel. Dabei war der kürzeste Weg der mit den wenigsten Runden und nicht der über die wenigsten Felder. Das kennt man auch von Navigations-Apps: Dort kann der kürzeste bzw. beste Weg die Route mit der kürzesten Wegstrecke oder die mit dem geringsten Zeitverbrauch sein, während sich ein Logistikunternehmen für die Route mit den geringsten Maut-Gebühren interessiert. Der Kürzeste-Wege-Algorithmus, den die App verwendet, muss sich deshalb nicht ändern.
In der Informatik können oft die gleichen Verfahren (Algorithmen) für ganz unterschiedliche Problemstellungen verwendet werden, wenn man diese passend modelliert.

Links-Rechts-Spiel

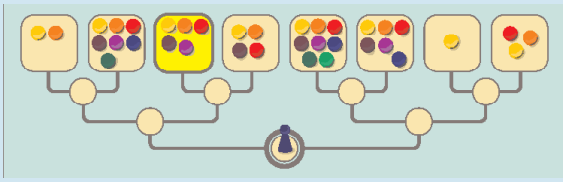
Beim Links-Rechts-Spiel gibt es acht Depots, auf die unterschiedlich viele Spiel-Chips verteilt werden. Eine Spielfigur zieht vom Startfeld aus zu einem Depot. Jedes Spielfeld hat eine Verzweigung, nach links oder rechts. Beim Startfeld entscheidet Alice, welchen Weg die Figur nimmt; bei der nächsten Verzweigung entscheidet Bob und schließlich wieder Alice. Alices Ziel ist, dass die Spielfigur ein Depot mit möglichst vielen Chips erreicht. Bobs Ziel ist hingegen, dass die Figur ein Depot mit möglichst wenigen Chips erreicht. Beide wissen voneinander, dass sie gute Spieler sind und sich immer für die Richtung entscheiden, die für ihr Ziel die beste ist.

Ein Beispiel: Wenn Bob die Figur zur Verzweigung ganz links zieht, dann weiß Bob, dass anschließend Alice die Figur auf das Depot mit 7 Spielsteinen zieht.
Ein neues Spiel beginnt, die Spiel-Chips sind verteilt.

Welches Depot wird die Spielfigur erreichen?

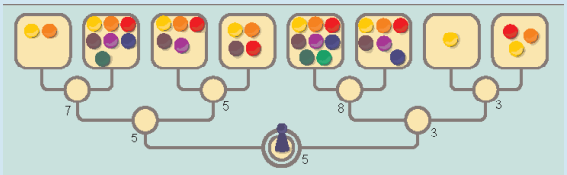


So ist es richtig:



Alice weiß: Wenn sie am Anfang die Spielfigur nach rechts ziehen würde, würde Bob anschließend sicher auch nach rechts ziehen, damit Alice höchstens 3 Chips erreichen kann. Sie zieht daher am Anfang die Spielfigur nach links; dann kann sie am Ende in jedem Fall mehr als 3 Chips erreichen. Bob weiß: Wenn er die Spielfigur dann nach links ziehen würde, würde Alice 7 Chips erreichen. Daher wählt er rechts, und Alice kann höchstens 5 Chips erreichen. Zuletzt wählt Alice natürlich links, damit sie 5 Chips erreicht und nicht nur 4 Chips.

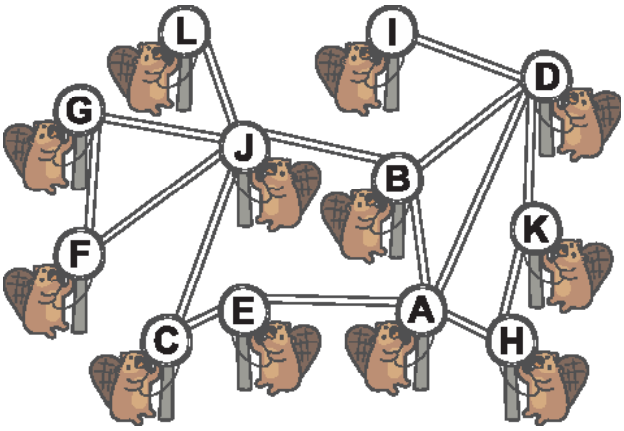
Das ist Informatik!
Die richtige Antwort in dieser Biberaufgabe kann systematisch berechnet werden, in dem man nach und nach bei den Verzweigungen aufschreibt, welches der von der Verzweigung aus erreichbaren Depots letztlich gewählt wird. An den Verzweigungen vor den Depots, bei denen Alice ihr Ergebnis maximiert, schreibt man also die größere Chip-Anzahl der beiden erreichbaren Depots auf: 7, 5, 8 und 3. Bei den Verzweigungen davor will Bob das Ergebnis minimieren; dort schreibt man also den kleineren Wert der folgenden Verzweigungen auf: 5 und 3. Beim Startfeld, wo Alice am Zug ist, wird der größere dieser beiden Werte als Endergebnis festgehalten, also 5.



Dieses Verfahren ist in der Informatik als MiniMax-Algorithmus bekannt. Damit kann man in vielen Zwei-Personen-Spielen den aktuell bestmöglichen Zug berechnen – wenn man alle Zugfolgen bis zu möglichen Endsituationen des Spiels berechnen kann. Beim Schach zum Beispiel ist das in der Regel nicht möglich, weil es zu viele solcher Zugfolgen gibt. In so einem Fall wird die Länge der Zugfolgen begrenzt und die damit erreichten Stellungen bewertet. Diese Bewertungen entsprechen den Chipzahlen der Depots in dieser Biberaufgabe. Mit dem MiniMax-Algorithmus wird dann eine Bewertung der aktuellen Stellung berechnet und der Zug, der zu dieser (besten) Bewertung führt, ausgewählt.

Nachrichten-Netz

Biber verbreiten gerne Nachrichten untereinander.

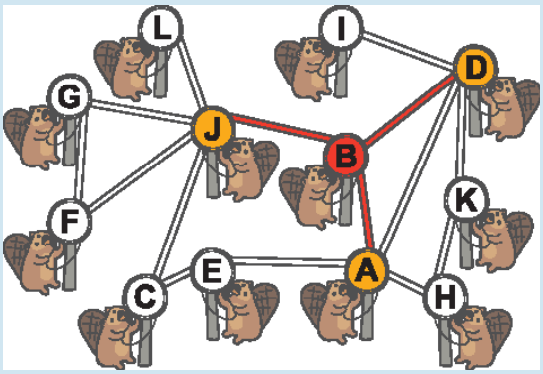


Sie haben dazu ein Nachrichten-Netz (siehe Bild rechts). Im Netz gibt es Nachbarn; sie sind durch eine Leitung miteinander verbunden.
Zum Beispiel hat Biber F die Nachbarn G und J.

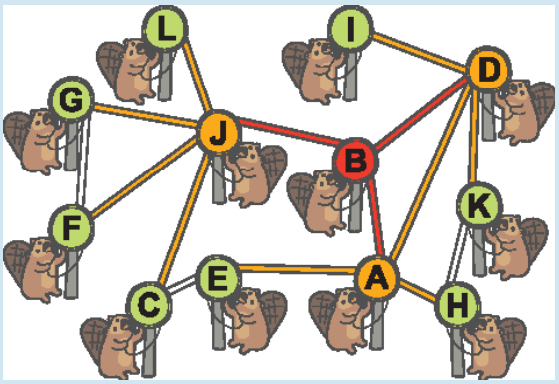
Die Nachrichten werden in Runden verbreitet:
In einer Runde leitet jeder Biber, der eine Nachricht hat, diese gleichzeitig an alle seine Netz-Nachbarn weiter.

Welcher Biber kann eine Nachricht in der kleinsten Anzahl Runden an alle anderen Biber im Netz verbreiten?

So ist es richtig:



Biber B kann in nur zwei Runden eine Nachricht an alle anderen Biber verbreiten. In der ersten Runde leite Biber B die Nachricht an alle seine Nachbarn weiter: A, D und J.
Das Bild zeigt, wer nach dieser Runde die Nachricht hat.
In der zweiten Runde leiten die Biber A, D und J die Nachricht jeweils an alle ihre Nachbarn weiter:
Biber A an die Biber E, H und D.
Biber D an die Biber A, I und K.
Biber J an die Biber C, F, G und L.
Außerdem geht die Nachricht jedes Mal wieder an Biber B zurück. Damit haben die Biber A, B und D die Nachricht nach der zweiten Runde doppelt. Für die Verbreitung spielt das aber keine Rolle. In nur einer Runde kann die Nachricht nicht verbreitet werden. Es gibt nämlich keinen Biber, der alle anderen Biber als Nachbarn hat.
Biber B ist der einzige, der die Nachricht in zwei Runden verbreiten kann: Die Biber C, E, F, G, H, J und L können Biber I nicht in zwei Runden erreichen. Und die Biber A, D, E, H, I und K können Biber L nicht in zwei Runden erreichen.



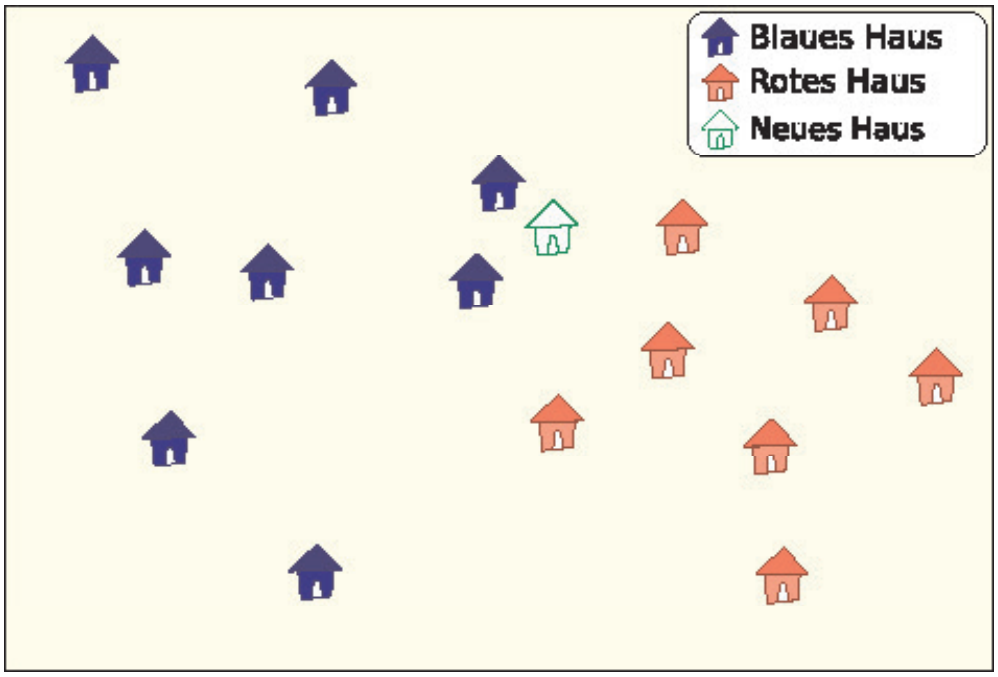
Das ist Informatik!
Die Verbreitung von Nachrichten in einem Kommunikationsnetzwerk, bei der die Nachrichten immer an alle Netz-Nachbarn weitergeleitet werden, wird in der Informatik als Broadcasting bezeichnet.
Im Netzwerk in dieser Biberaufgabe ist das Broadcasting besonders schnell abgeschlossen, wenn es bei Biber B startet. Am besten und günstigsten ist ein Netzwerk, in dem das Broadcasting von allen Netzknoten aus schnell geht und das nicht unnötig viele Verbindungen hat.
Ein Kommunikationsnetzwerk kann man mathematisch auch als Graph modellieren, mit den Stationen (hier: den Bibern) als Knoten und den Verbindungen als Kanten. In der Sprache der Graphentheorie ist Biber B bzw. seine Station ein Zentrum des Graphen, der das Netzwerk darstellt. Es gibt dann keinen Knoten, der zu allen anderen Knoten eine kleinere Entfernung hätte. In der Informatik sind Algorithmen bekannt, mit denen man das Zentrum bzw. die Zentren eines Graphen effizient bestimmen kann.

Neues Haus

In einem Dorf werden alle Häuser entweder blau oder rot angestrichen.
Um über die Farbe eines neuen Hauses zu entscheiden,
haben die Bewohner eine Zahl k und diese Regel festgelegt:

- Ein neues Haus muss die Farbe bekommen, welche die Mehrheit der k nächstgelegenen Häuser hat. Wenn es keine Mehrheit gibt, entscheidet die Mehrheit der $k + 1$ nächstgelegenen Häuser.

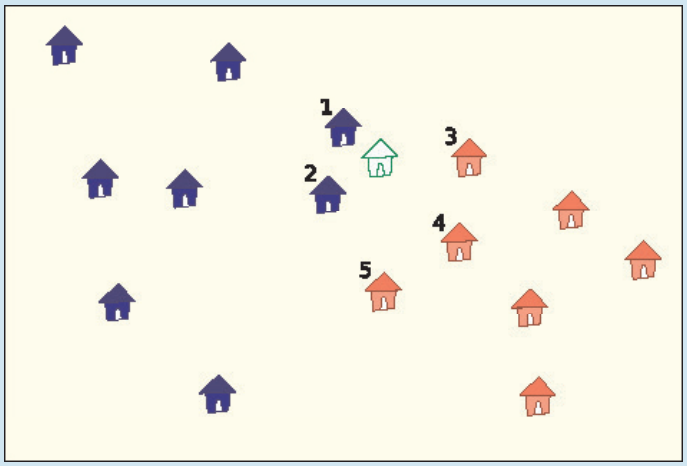
Nun wurde wieder ein neues Haus gebaut. Das Bild zeigt die Lage aller Häuser im Dorf.



Die Regel entscheidet, dass das neue Haus die Farbe Rot bekommt.

Wie lautet die kleinste Zahl k, die zu dieser Entscheidung führt?

***k* = 4 ist die richtige Antwort:**
Man kann die richtige Antwort finden, indem man für *k* ansteigende Zahlen einsetzt und jedes Mal prüft, ob das neue Haus gemäß der Regel die Farbe Rot bekommen muss.



Wir beginnen mit *k* = 1. Es wird nur das nächstgelegene Haus betrachtet; es ist blau. Gemäß der Regel muss das neue Haus die Farbe Blau bekommen.
Nun setzen wir *k* = 2. Die 2 nächstgelegenen Häuser sind blau. In diesem Fall besagt die Regel ebenfalls, dass das neue Haus die Farbe Blau bekommen muss.
Nun setzen wir *k* = 3. Von den 3 nächstgelegenen Häusern ist 1 rot und 2 blau. Die Mehrheit der 3 nächstgelegenen Häusern hat die Farbe Blau, und das neue Haus muss ebenfalls die Farbe Blau bekommen.
Nun setzen wir *k* = 4. Von den 4 nächstgelegenen Häusern sind 2 rot und 2 blau. Weil es keine Farb-Mehrheit gibt, muss die Mehrheit der *k* + 1 = 5 nächstgelegenen Häuser entscheiden. Von diesen haben 2 Häuser die Farbe Blau, und 3 Häuser haben die Farbe Rot – das ist die Mehrheit.
Für *k* = 4 bekommt das neue Haus also die Farbe Rot. Da alle kleineren Zahlen zu einer Entscheidung für Blau geführt haben, ist 4 die kleinste Zahl, die zu einer Entscheidung für die Farbe Rot führt.
Auch *k* = 5 führt zu einer Entscheidung für Rot, ist aber nicht die kleinste solche Zahl.

Das ist Informatik!
Eine typisch menschliche Vorgehensweise ist es, Dinge „in Schubladen zu stecken“, also in Kategorien oder Klassen einzuteilen. Ob man zum Beispiel ein Ding zum Sitzen benutzen kann, ergibt sich daraus dass man es der Klasse „Stuhl“ zuordnen kann. Dieses Klassifizieren vereinfacht das Leben, weil man nicht mehr über jedes Ding einzeln nachdenken muss.
Insbesondere Computersysteme, die Tätigkeiten von Menschen übernehmen, übernehmen damit auch das Klassifizieren: Ist ein Bankkunde kreditwürdig oder nicht? Zu welcher Risikogruppe gehört ein Kunde einer Versicherung? Zeigt ein digitales Bild einen Hund oder eine Katze?
Damit Computer klassifizieren können, benötigen sie geeignete Algorithmen. In der Informatik sind viele, zum Teil sehr leistungsfähige Algorithmen zur Klassifikation bekannt. Einen davon stellt diese Biberaufgabe vor: „*k* nächste Nachbarn“. Ein Ding wird derjenigen Klasse zugeordnet, zu der die *k* anderen Dinge gehören, die dem neuen Ding am nächsten oder nach irgendeinem Maßstab am ähnlichsten sind. Wenn man festlegt, dass *k* eine ungerade Zahl ist, gibt es immer eine Mehrheit und man kann sich die *k*+1-Regel aus der Aufgabe sparen.
In der Aufgabe gibt es nur zwei Klassen, nämlich die Farben Rot und Blau. „*k* nächste Nachbarn“ funktioniert aber auch mit mehreren Klassen. Allerdings hängt es von vielen Feinheiten ab, wie gut die Klassifizierung ist. Allgemein sollten Menschen die Qualität von klassifizierenden oder auch anderweitig künstlich intelligenten Computersystemen gründlich überprüfen.

https://de.wikipedia.org/wiki/N%C3%A4chste-Nachbarn-Klassifikation

Nim(m)

Susi und Hans spielen ein Spiel mit 3 schwarzen und 7 weißen Steinen.
Sie nehmen abwechselnd Steine weg. Ein Spieler darf
entweder 1 oder 2 schwarze Steine wegnehmen oder 1, 2 oder 3 weiße Steine.
Der Spieler, der den letzten Stein einer der beiden Farben wegnimmt, hat gewonnen.

Susi beginnt.
Welche Steine soll Susi nun wegnehmen, damit sie sicher das Spiel gewinnt?
A) 1 weißen Stein B) 2 schwarze Steine
C) 3 weiße Steine D) Das ist egal, Susi gewinnt auf jeden Fall.

Antwort C ist richtig:
Wir unterscheiden 2 Fälle:
Susi nimmt 1 oder 2 schwarze Steine. Dann verliert sie, da Hans dann anschließend die restlichen schwarzen Steine wegnehmen kann. Daher kann Antwort B nicht richtig sein – und Antwort D auch nicht. Susi nimmt weiße Steine. Dann kommt es darauf an, wie viele sie davon wegnimmt. Wenn sie 3 weiße Steine nimmt, dann bleiben für Hans 3 schwarze und 4 weiße Steine übrig. Egal ob dann Hans schwarze oder weiße Steine nimmt, Susi kann als Antwort darauf immer eine der beiden Farben komplett wegnehmen. Daher ist C die richtige Antwort.
Wählt Susi 1 oder 2 weiße Steine, dann kann Hans als Antwort darauf die Anzahl weißer Steine auf 4 reduzieren. Jetzt sind also für Susi 3 schwarze und 4 weiße Steine übrig, und Susi ist in der selben Verlustposition wie Hans oben. Daher kann Antwort A nicht richtig sein.

Das ist Informatik!
Das Nim-Spiel, das Susi und Hans miteinander spielen, ist ein einfaches Exemplar der Klasse der Zwei-Personen-Spiele. Das Gute an einfachen Exemplaren ist, dass man sie besonders gut studieren und die Erkenntnisse dann möglicherweise auf alle Exemplare der Klasse übertragen kann. Die Erkenntnisse über Spiele wiederum sind für die Informatik interessant, da auch reale Interaktionen, zum Beispiel bei wirtschaftlichem Handeln, wie Spiele funktionieren.
Allerdings ist Nim ein besonderes Zwei-Personen-Spiel. In komplexen Spielen wie Schach, können in den meisten Situationen noch beide Spielenden gewinnen, egal welcher Zug als nächster gemacht wird. In Nim ist aber in jeder Situation klar, wer gewinnen wird – vorausgesetzt, der nächste Zug ist der richtige. In der Situation im Nim-Spiel dieser Biberaufgabe (3 schwarze und 7 weiße Steine, Susi ist am Zug) kann Susi gewinnen, wenn sie den richtigen Zug macht. Diese Situation ist deshalb eine Gewinnsituation für Susi.
Erkenntnisse über Nim können also in der Regel nur auf Zwei-Personen-Spiele mit der gleichen Eigenschaft übertragen werden. Das gilt zum Beispiel für die Formeln zur Berechnung der besten Züge in Nim, die schon früh entdeckt wurden. Kompliziertere „Spiel-Algorithmen“ wie etwa der Minimax-Algorithmus (siehe auch die Aufgabe „Links-Rechts-Spiel“ in diesem Biberheft) sind für Nim nicht erforderlich. Weil Nim-Spiele so systematisch zu spielen und zu gewinnen sind, wurden sie schon sehr früh als Computerprogramme realisiert. Schon 1951 gab es den Computer Nimrod, der Nim erfolgreich spielen konnte.

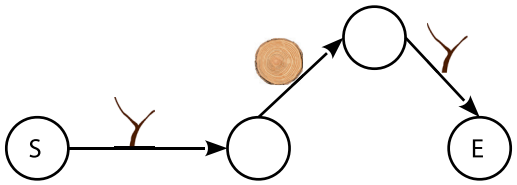
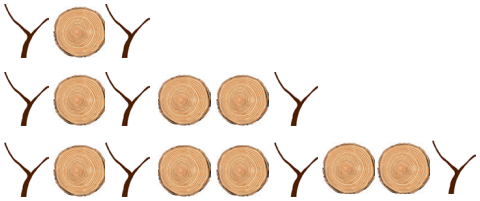
https://de.wikipedia.org/wiki/Nim-Spiel

Passwörter

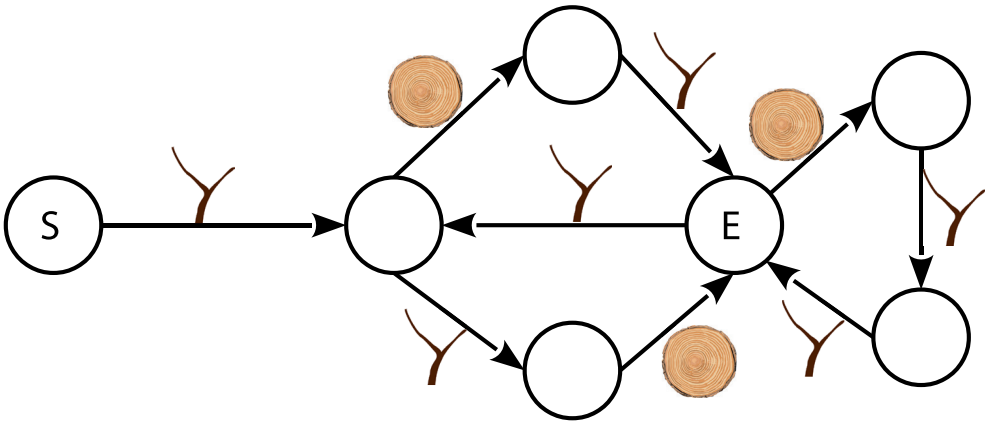
Die Biber haben eigene Regeln für Passwörter.

Die Passwörter sind aus den beiden Symbolen  und  aufgebaut.

Ein Diagramm legt die gültigen Passwörter fest. Ein Passwort ist dann gültig, wenn es im Diagramm einen Pfad vom Kreis S entlang der Pfeile zum Kreis E gibt, der der Reihe nach die Symbole des Passworts enthält. Andere Passwörter sind nicht gültig.

Ein Beispiel: Nach diesem Diagramm …	… sind unendlich viele Passwörter gültig, z.B.:
	

Die Biber erfinden ein neues Diagramm:



Welches der folgenden Passwörter ist nach dem neuen Diagramm gültig?

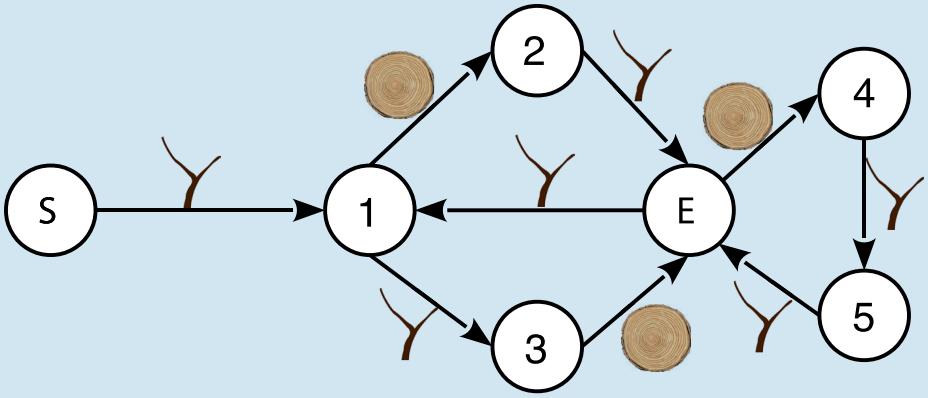
A 

B 

C 

D 

Antwort A ist richtig:
Hier ist noch einmal das neue Diagramm, mit den Nummern 1 bis 5 und den Buchstaben S und E als Bezeichnungen für die Kreise. Damit können wir Pfade im Diagramm als Folge dieser Bezeichnungen beschreiben.



Antwort A: Das Passwort ist gültig.
Der Pfad S-1-3-E-4-5-E-1-2-E enthält die Symbole des Passworts der Reihe nach.

Antwort B: Das Passwort ist nicht gültig.

Das erste Symbol ist , aber alle gültigen Passwörter beginnen mit .

Antwort C: Das Passwort ist nicht gültig.
Der einzig mögliche Pfad, auf dem die Symbole in der Reihenfolge des Passworts enthalten sind, ist S-1-2-E-4-5-E-4. Dieser Pfad endet aber nicht beim Kreis E. Außerdem kann man erkennen, dass die Länge aller gültigen Passwörter ein Vielfaches von 3 sein muss. Das Passwort aus Antwort C hat aber 7 Symbole.

Antwort D: Das Passwort ist nicht gültig.
Von S aus kann man genau einen Pfad gehen, der die ersten acht Symbole dieses Passworts der Reihe nach enthält: S-1-3-E-4-5-E-4-5. Dieser Pfad kann aber nicht weitergeführt werden, da von Kreis 5 aus kein Pfeil mit dem Symbol weiterführt.
Außerdem kann man erkennen, dass gültige Passwörter doppelt so viele wie haben. Das Passwort aus Antwort D enthält das Symbol 5-mal und das Symbol 4-mal.

Das ist Informatik!
Zur Überprüfung möglicher Passwörter verwenden die Biber Diagramme mit Kreisen und Pfeilen, welche die möglichen Übergänge zwischen den Kreisen beschreiben. Dabei geht von jedem Kreis für jedes Symbol höchstens ein Pfeil aus, es gibt also höchstens einen nächstmöglichen Kreis. Dadurch wird die Überprüfung von Passwörtern sehr vereinfacht, weil man immer nur einen einzigen Weg durch das Passwort-Diagramm verfolgen muss.
Diagramme wie die Passwort-Diagramme in dieser Biberaufgabe werden in der Informatik auch als Zustands-Übergangsdiagramme bezeichnet. Mit ihnen kann man das Verhalten sogenannter *endlicher Automaten* beschreiben. Das sind Modelle für einfache Berechnungen: Sie haben eine Menge von Zuständen, und für jeden Zustand ist festgelegt, in welchen anderen Zustand der Automat bei der Verarbeitung einer neuen Eingabe übergeht. Bei jeder neuen Berechnung befindet sich der Automat in einem besonderen Startzustand. Genau dann, wenn der Automat nach Verarbeitung aller Eingaben einen besonders gekennzeichneten Zustand erreicht, erfüllt die Eingabe die durch den Automaten realisierten Bedingungen. Ein solcher Automat wird als *deterministisch* bezeichnet, wenn von jedem Zustand aus für jedes Eingabezeichen höchstens ein Folgezustand festgelegt wird.
Jedes Passwort-Diagramm in dieser Biberaufgabe beschreibt also einen deterministischen endlichen Automaten. Der Startzustand ist mit „S“ gekennzeichnet, und es gibt jeweils genau einen Zustand „E“, bei dessen Erreichen nach Verarbeitung aller Symbole das Passwort als gültig akzeptiert wird.

Prüf-Biber

Der Biber-Boss setzt vier Biber ein, um den anderen Bibern Flaggen-Nachrichten zu senden.
Jeder dieser Nachrichten-Biber hält entweder eine rote oder eine gelbe Flagge hoch.

Es kann passieren, dass ein Biber die falsche Flagge hochhält.
Das möchte der Biber-Boss erkennen können.
Deshalb bestimmt er drei Prüf-Biber.

Jeder Prüf-Biber prüft drei Nachrichten-Biber.
Wenn diese drei eine ungerade Anzahl an roten Flaggen hochhalten sollen,
dann soll ihr Prüf-Biber auch eine rote Flagge hochhalten, sonst eine gelbe.
Wenn alle die richtigen Fahnen hochhalten, dann halten ein Prüf-Biber
und seine Nachrichten-Biber zusammen eine gerade Anzahl an roten Flaggen hoch.

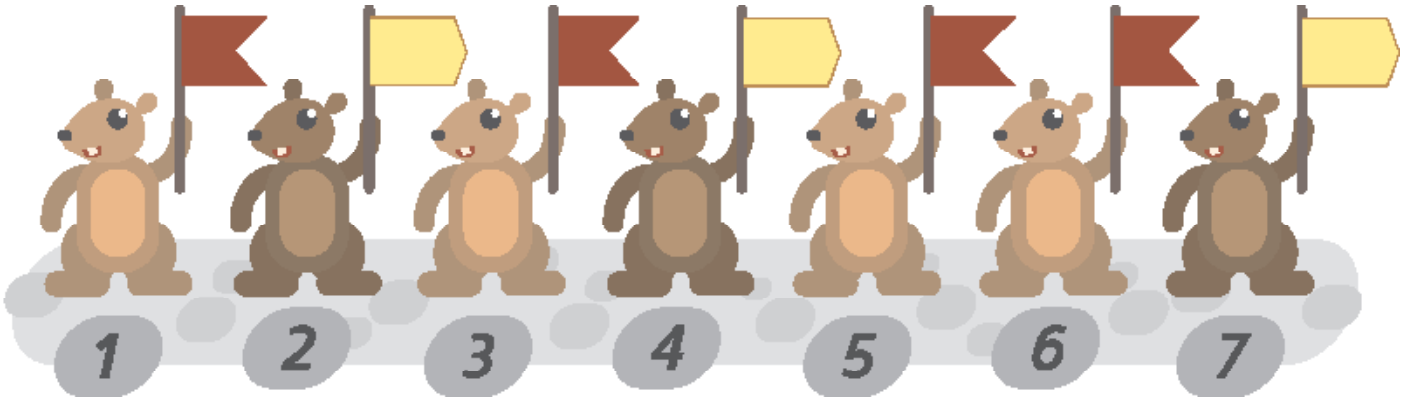
Insgesamt werden in einer Nachricht nun sieben Flaggen hochgehalten.

Der Boss gibt Nachrichten-Bibern und Prüf-Bibern Nummern und ordnet sie so einander zu:

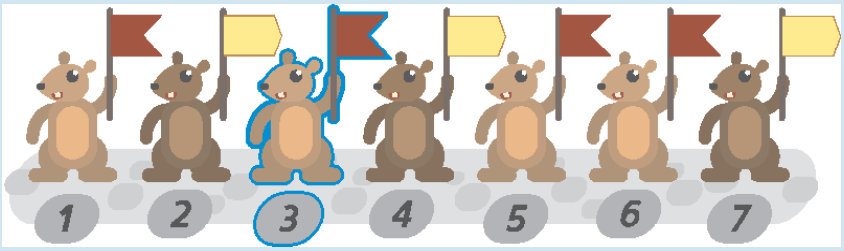
Nachrichten-Biber	Prüf-Biber
1, 2, 3	5
1, 2, 4	6
2, 3, 4	7

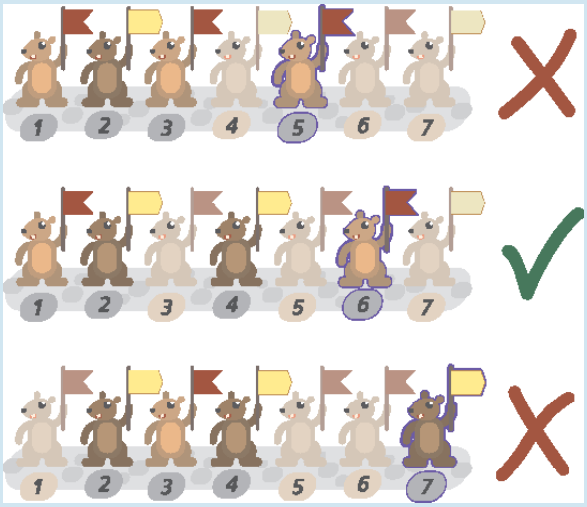
Der Biber-Boss sieht die Nachricht unten.
Er weiß sofort, dass genau einer der sieben Biber die falsche Fahne hochhält.

Welcher Biber hält die falsche Fahne hoch?



So ist es richtig:
Biber Nummer 3 hält die falsche Fahne hoch.



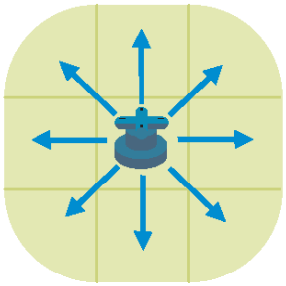


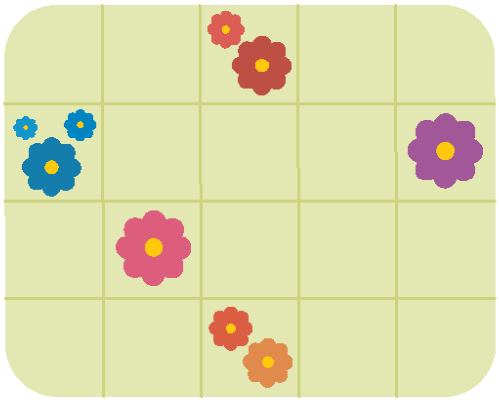
Einen Prüf-Biber und seine Nachrichten-Biber wollen wir als Gruppe bezeichnen und geben ihr die Nummer des Prüf-Bibers. Es gibt also die drei Gruppen 5, 6 und 7. Eine Gruppe soll also eine gerade Anzahl an roten Flaggen hochhalten. Hier sind die Gruppen, der Prüf-Biber der Gruppe ist markiert.
Man sieht: Bei Gruppe 5 gibt es einen Fehler, sie hält drei rote Fahnen hoch. Bei Gruppe 6 ist alles in Ordnung (zwei rote Fahnen), aber bei Gruppe 7 gibt es auch einen Fehler, sie hält nur eine rote Fahne hoch.
Die Biber aus Gruppe 6, also die Biber 1, 2, 4 und 6, machen alles richtig. In Gruppe 5 könnte also entweder Biber 3 oder Biber 5 die falsche Fahne hochhalten. In Gruppe 7 könnte entweder Biber 3 oder Biber 7 die falsche Fahne hochhalten. Da insgesamt nur genau ein Biber die falsche Fahne hochhält, muss der gleiche Biber für die Fehler in den Gruppen 5 und verantwortlich sein. Das kann nur Biber 3 sein.

Das ist Informatik!
In der digitalen Welt werden Nachrichten und auch andere Daten binär codiert, also als Folgen aus Bits (0 und 1), und so über Kommunikationskanäle übertragen, ob über Leitungen oder drahtlos. Dabei kann es zu Störungen kommen, welche die Bits vertauschen: aus 0 wird 1 oder umgekehrt. Für eine korrekte Informationsverarbeitung möchte man feststellen, ob eine Übertragung von einer Störung betroffen war, und die entstandenen Fehler korrigieren. Dazu wurden *fehlerkorrigierende Codes* entwickelt.
Eine einfache Möglichkeit wäre, jedes Bit dreimal hintereinander zu senden. Eine Störung an einem der drei Bits könnte leicht erkannt und korrigiert werden, denn die zwei anderen Bits tragen noch immer die richtige Information; die Mehrheit entscheidet dann. Dieses einfache Verfahren führt aber dazu, dass drei mal so viele Daten übertragen werden müssen. Damit die Datenleitungen nicht verstopfen, ist es wichtig, möglichst wenige zusätzliche Bits zur Fehlerkorrektur zu verwenden.
Der Biber-Boss setzt einen *Hamming-Code* ein. In einem Hamming-Code gibt es für mehrere Gruppen der eigentlichen Datenbits jeweils ein Prüfbit. Das Prüfbit wird so aus den Datenbits der passenden Gruppe berechnet, dass Datenbits und Prüfbit zusammen eine gerade Anzahl von 1en enthalten; diese Eigenschaft nennt man auch „gerade Parität“. Wenn in einer Nachricht für alle Gruppen aus Datenbits und zugehörigem Prüfbit die Parität stimmt, dann kann man annehmen, dass die Nachricht richtig übertragen wurde. Wurde nur ein Bit einer mit Hamming-Code kodierten Nachricht durch eine Störung verändert, lässt sich das gestörte Bit und die Originalnachricht korrekt bestimmen, indem man schaut, bei welchen Gruppen die Parität nicht stimmt.
Mit drei Prüfbits kontrolliert der Hamming-Code vier Datenbits, wie in dieser Biberaufgabe. Mit vier Prüfbits können schon 11 Datenbits kontrolliert werden, das Codewort ist dann 15 Bits lang. Mit k Prüfbits kann das Codewort 2^k Bits lang sein. Für lange Codewörter werden also nur wenige zusätzliche Bits benötigt. Dass man mit Hilfe des Hamming-Codes nur einen Bitfehler pro Codewort korrigieren kann, ist in vielen Fällen gut genug. In der Informatik sind Codes bekannt, mit deren Hilfe man auch mehrere Fehler korrigieren kann.

https://de.wikipedia.org/wiki/Hamming-Code

Rasensprenger



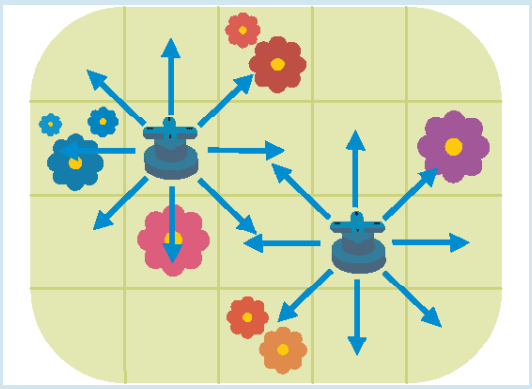


Unten siehst du Bobs Garten. Der Garten ist in Felder eingeteilt. In einigen Feldern sind Blumen.

Bob möchte die Blumen mit Rasensprengern gießen. Ein Sprenger kann nur auf ein leeres Feld gestellt werden.Er gießt dann die 8 Felder rundherum.

Stelle Sprenger im Garten auf, die alle Blumen gießen. Benutze so wenige Sprenger wie möglich!

So ist es richtig:
Man kann zwei Rasensprenger aufstellen und damit alle Blumen gießen:



Ein Rasensprenger muss rechts, links oder schräg neben einer Blume stehen, um sie zu gießen. Ein einzelner Sprenger kann nicht gleichzeitig neben allen Blumen in Bobs Garten stehen, zum Beispiel nicht neben den Blumen ganz links und der Blume ganz rechts. Deshalb muss man mindestens zwei Sprenger aufstellen, und zwar genau so wie im Bild.

Das ist Informatik!
Diese Biberaufgabe stellt uns vor ein *Optimierungsproblem:* Alle Blumenfelder sollen bewässert werden, aber die Anzahl der benötigten Rasensprenger soll möglichst klein sein. Möglichst klein ist hier „optimal“, also am besten – zum Beispiel für das Bankkonto der Person, welche die Sprenger bezahlen muss. (Siehe auch die Aufgabe „Blumenkasten“ in diesem Biberheft.)
Durch das Aufstellen der Sprenger teilen wir die Menge der Blumen auf: Die Blumen, die vom selben Sprenger gegossen werden, gehören zum selben Teil. Mit mehr Sprengern könnten wir mehr Teile bilden, aber es soll ja eine beste Aufteilung mit möglichst wenig Sprengern sein. Eine solche Aufteilung heißt in der Informatik auch Partitionierung.
Beste bzw. optimale Aufteilungen kann man in vielen Situationen gebrauchen, etwa wenn man mit möglichst wenigen Hotspots alle Räume einer Schule mit gutem WLAN versorgen möchte. Gibt es viele Räume (oder viele Blumenfelder), kann die Suche nach einer besten Aufteilung ein sehr schwieriges Problem sein. Daher ist man häufig schon zufrieden, wenn man eine Aufteilung findet, die vielleicht nicht optimal ist, aber fast. Bräuchte man für die beste Aufteilung theoretisch 100 Hotspots, ist eine Aufteilung mit 105 Hotspots ziemlich gut, wenn es zu schwierig ist, die beste zu finden.

Rückseite

Dein Freund Aristo hat Spielkarten mitgebracht.
Auf der einen Seite jeder Karte ist ein Buchstabe und auf der anderen Seite ist eine Zahl.
Aristo behauptet: Wenn auf der einen Seite einer Karte ein Vokal ist,
dann ist auf der anderen Seite eine gerade Zahl.

Aristo legt vier Karten vor dich hin.
Du weißt, dass E ein Vokal, V ein Konsonant, 2 gerade und 7 ungerade sind.
Aber weißt du auch, ob Aristo die Wahrheit gesagt hat?
Du willst seine Behauptung sicher überprüfen.

Welche Karten musst du dazu unbedingt umdrehen?

So ist es richtig:

Die E-Karte muss umgedreht werden, um zu prüfen, ob auf der Rückseite eine gerade Zahl ist. Wäre sie ungerade, hätte Aristo die Unwahrheit gesagt.
Die V-Karte muss nicht umgedreht werden. Über Konsonanten hat Aristo nichts gesagt, also keine Wahrheit und auch keine Unwahrheit. Die 2-Karte muss nicht umgedreht werden. Falls auf der Rückseite ein Konsonant wäre, hätte Aristo keine Unwahrheit gesagt. Falls dort ein Vokal wäre, hätte er die Wahrheit gesagt. Die 7-Karte muss umgedreht werden. Wäre auf der Rückseite ein Vokal, hätte Aristo die Unwahrheit gesagt.

Das ist Informatik!
Es ist gar nicht schwer, einen Computer denken zu lassen. Vor allem, wenn es um das Denken in klassisch-logischen Implikationen geht. Einige Programmiersprachen bieten dazu als Basis das Konstrukt (IF a THEN b) an. Das ist nicht die berühmte „if-Anweisung“ oder „bedingte Anweisung“, sondern ein Ausdruck mit logischem Wert, der die beiden Ausdrücke a und b zu „wenn a, dann b“ verbindet. Der Ausdruck ist genau dann „falsch“, wenn a den Wert „wahr“ und b den Wert „falsch“ hat; sonst ist der Ausdruck „wahr“.
In dieser Biberaufgabe gibt es auch ein (IF a THEN b): a ist die Aussage „auf einer Seite ist ein Vokal“, und b ist die Aussage „auf der anderen Seite ist eine gerade Zahl“. (IF a THEN b) kann also falsch sein, wenn auf der sichtbaren Seite wirklich ein Vokal ist (a ist „wahr“) oder eine ungerade Zahl ist (b ist „falsch“). Deshalb mussten die Karten mit E und 7 umgedreht werden, um die Wahrheit des gesamten Ausdrucks zu überprüfen.
In Programmiersprachen mit (IF a THEN b) kann man sogar einen weit verbreiteten menschlichen logischen Denkfehler programmieren:
(IF (IF a THEN b) THEN (IF b THEN a))
Dieser Ausdruck ist nicht immer „wahr“, sondern ist „falsch“, wenn a „falsch“ und b „wahr“ ist.

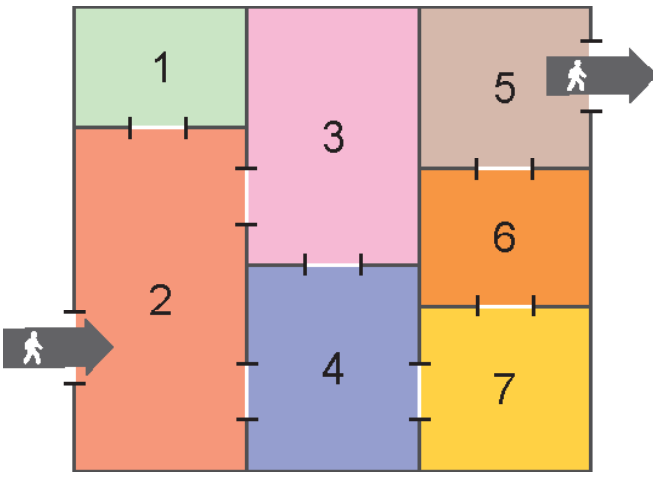
Rundgang

Ein neues Museum wird geplant. Die Besucher sollen darin einen Rundgang machen. Bei einem Rundgang geht man durch alle Räume und betritt jeden Raum nur einmal.

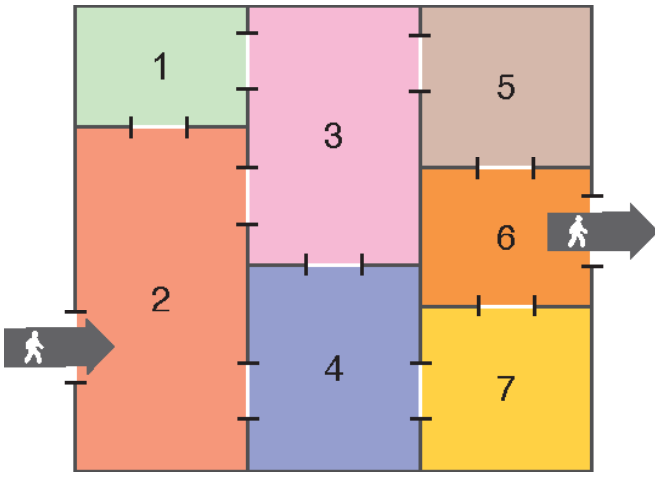
Es werden vier Pläne gemacht. Alle haben sieben Räume (1 bis 7).
Die Pfeile zeigen, wo die Besucher in das Museum hineingehen und wo sie wieder hinausgehen sollen.

Nur bei einem Plan kann man einen Rundgang machen. Bei welchem?

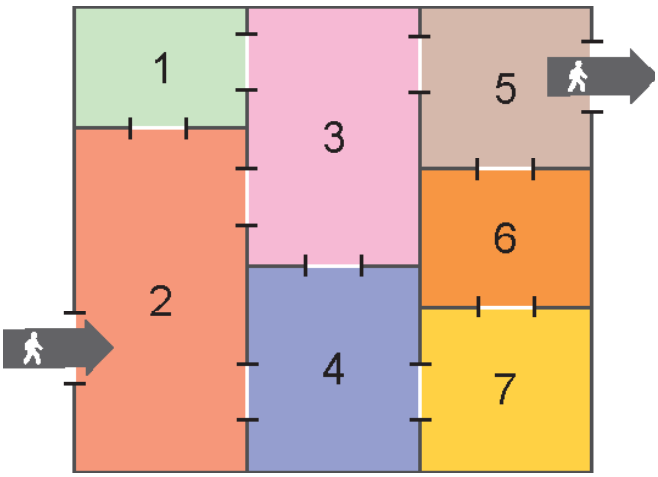
A)



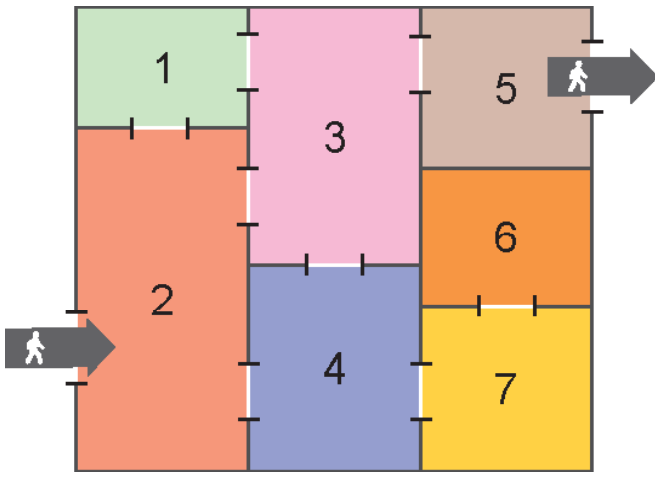
B)



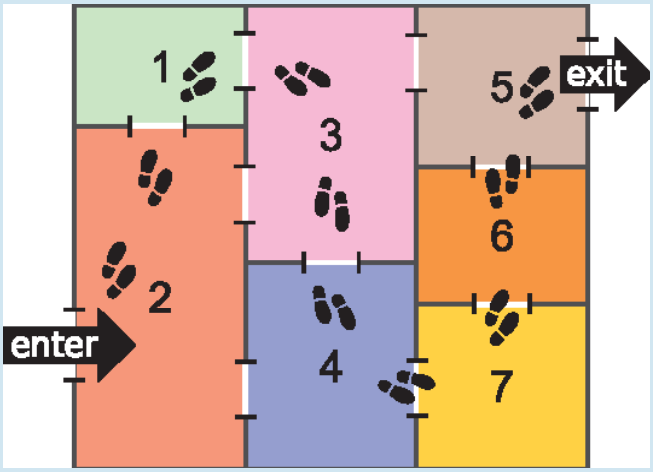
C)



D)

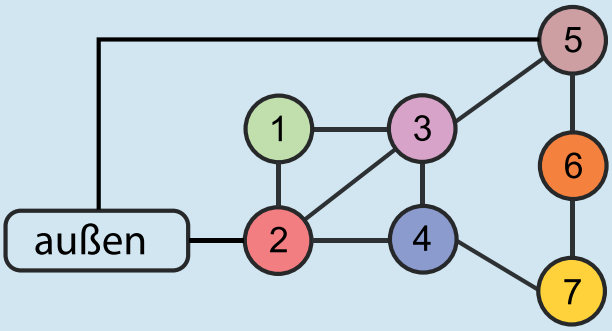


Nur der Plan C erlaubt den Besuchern, einen Rundgang zu machen, und zwar so:



Falls ein Raum nur einen Eingang hat, kann man keinen Rundgang nach den vorgegebenen Regeln machen: Wenn eine Besucherin diesen Raum betritt, kann sie ihn nur durch den einen Eingang auch wieder verlassen. So kommt sie zurück in den Raum, aus dem sie gekommen ist, und betritt diesen zum zweiten Mal. Damit ist die Rundgang-Regel verletzt.
Deshalb erlauben die Pläne A und D keinen Rundgang. In Plan A hat Raum 1 nur einen Eingang, und in Plan D hat Raum 6 nur einen Eingang.
In Plan B kann der Raum 6 von Raum 5 oder von Raum 7 erreicht werden. Falls die Besucherin von Raum 5 kommt, muss sie Raum 6 durchqueren, um den Raum 7 zu betreten (und umgekehrt). Dann kann sie den Ausgang nur noch erreichen, indem sie Raum 6 zum zweiten Mal betritt.

Das ist Informatik!

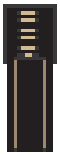


Auch wenn das Museum nicht sehr groß ist, muss man genau hinsehen, wo man lang geht, um die möglichen Wege zu prüfen. Dabei sind die Größe der Räume und deren genauen Lage für die Frage nach einem Rundgang unerheblich. Man muss nur wissen, von welchem Raum man in welche anderen Räume gelangen kann und wo die Verbindungen nach außen sind. Das können wir für Plan C so aufzeichnen:
Damit haben wir den Plan auf das Wesentliche reduziert – und ihn als Graphen modelliert (siehe auch die Aufgabe „Geschäfte“ in diesem Biberheft). Ein Graph ist eine mathematische Struktur aus Knoten und Kanten: Hier entsprechen die Räume (einschließlich des Raums „außen“) den Knoten und die Türen den Kanten. Ein Rundgang ist dann ein Weg über Kanten des Graphen, bei dem alle Knoten genau einmal besucht werden, bis man beim ersten Knoten („außen“) wieder ankommt.
In der Welt der Graphen nennt man einen solchen Weg einen Hamiltonkreis. Das Hamiltonkreis-Problem, also die Entscheidung darüber, ob es in einem Graph einen Hamiltonkreis gibt, ist eines der berühmten NP-vollständigen Probleme und damit im Allgemeinen eines der schwierigsten Probleme, die die Informatik kennt. Wenn der Graph so übersichtlich ist wie bei den Museumsplänen dieser Biberaufgabe, ist das Hamiltonkreis-Problem aber nicht so schwer zu lösen. Generell lässt sich die Informatik von einem schwierigen Problem nicht beeindrucken. Sie sucht dann unter anderem nach Spezialfällen, für die man das Problem doch gut lösen kann. Für das Hamiltonkreis-Problem sind einige bekannt.

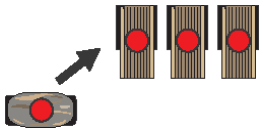
https://de.m.wikipedia.org/wiki/Hamiltonkreisproblem

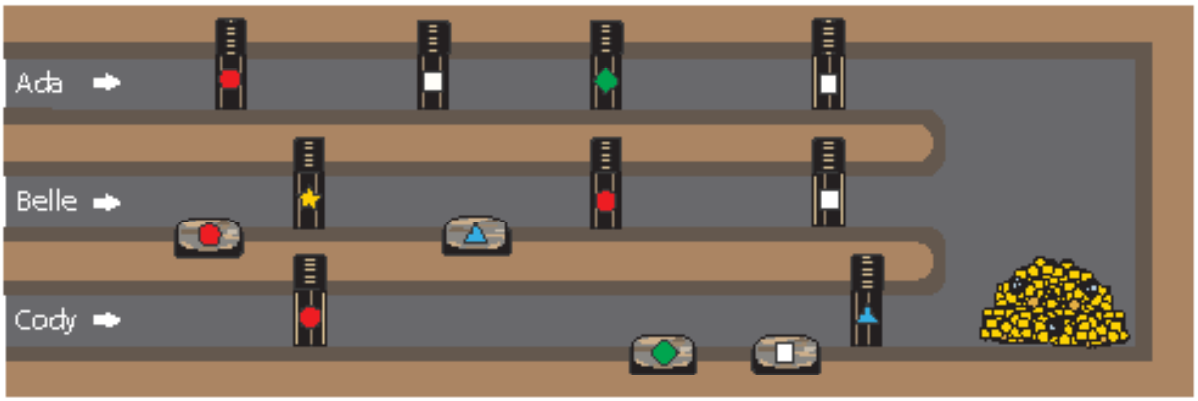
Schatzsuche

Die drei Entdeckerinnen Ada, Belle und Cody wollen einen Schatz erreichen.
Jede geht durch einen anderen Gang.

In den Gängen sind Tore  und Steine  .
Darauf sind verschiedene farbige Zeichen.
Am Anfang sind alle Tore verschlossen.

Wenn eine Entdeckerin an ein verschlossenes Tor kommt,
muss sie warten, bis das Tor geöffnet wird.
Wenn eine Entdeckerin auf einen Stein tritt,
werden alle Tore mit dem gleichen Zeichen für immer geöffnet.





Wer kann den Schatz erreichen?

A) Ada
B) Belle
C) Cody
D) Niemand kann den Schatz erreichen.

Antwort A ist richtig:
Belle öffnet zuerst alle Tore mit einem roten Kreis. Dann kann Cody weitergehen und zuerst alle Tore mit einem grünen Karo und dann alle Tore mit einem weißen Kasten öffnen. Danach sind in Adas Gang alle Tore offen, und sie kann den Schatz erreichen.
Antwort B ist falsch: Weil es keinen Stein mit einem gelben Stern gibt, kommt Belle in ihrem Gang nur bis zum ersten Tor und nicht weiter.
Antwort C ist falsch: Das letzte Tor in Codys Gang kann nur durch Belle geöffnet werden, indem sie in ihrem Gang auf den Stein mit dem blauen Dreieck tritt. Aber Belle kann diesen Stein niemals erreichen. Also kann Cody den Schatz nicht erreichen.
Antwort D ist falsch, weil Ada den Schatz erreichen kann.

Das ist Informatik!
Die drei Entdeckerinnen in der Aufgabe sind wie Prozessoren eines Computers, die parallel Daten verarbeiten und gemeinsame Sperrmechanismen verwenden.

Ein Prozessor ist das Herzstück eines Computers. Er führt die Anweisungen der auf dem Computer aktiven Programme aus. Um die Arbeitsgeschwindigkeit von Computern zu steigern, wurde in der Informatik lange Zeit daran gearbeitet, Technik und Logik von Prozessoren zu verbessern. Weil das seit einigen Jahren immer schwieriger geworden ist, werden moderne Computer dadurch beschleunigt, dass sie mehrere Prozessoren bekommen, die sich die Arbeit teilen.

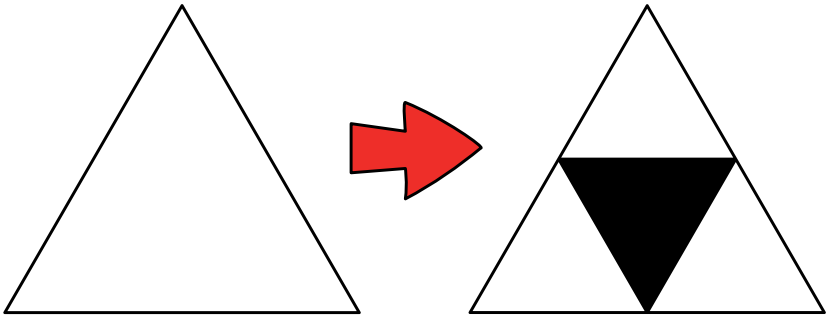
Um das auszunutzen, entwickeln Informatikerinnen und Informatiker Methoden, die Anweisungen von Programmen in Teile aufzuspalten, die möglichst unabhängig voneinander sind und deshalb zeitlich parallel – d.h. von mehreren Prozessoren gleichzeitig – ausgeführt werden können. Die parallel laufenden Teile werden in der Informatik auch als „Threads“ (englisch für Fäden) bezeichnet.

In der Regel können die Abhängigkeiten zwischen den Threads aber nicht ganz aufgelöst werden. Dann gibt es kritische Anweisungen, die nicht gleichzeitig von mehreren Threads bearbeitet werden dürfen. Wenn ein Thread zu diesen Anweisungen kommt, setzt er für alle anderen Threads eine Sperre (engl.: Lock) und öffnet sie wieder, wenn die kritischen Anweisungen abgearbeitet sind. Das Gleiche erleben die Entdeckerinnen in dieser Biberaufgabe auf ihrem gleichzeitigen Weg durch die Gänge, wenn sie an ein Tor gelangen, das den Weg versperrt – und von einer anderen Entdeckerin geöffnet wird, die in ihrem Gang auf den passenden Stein tritt.

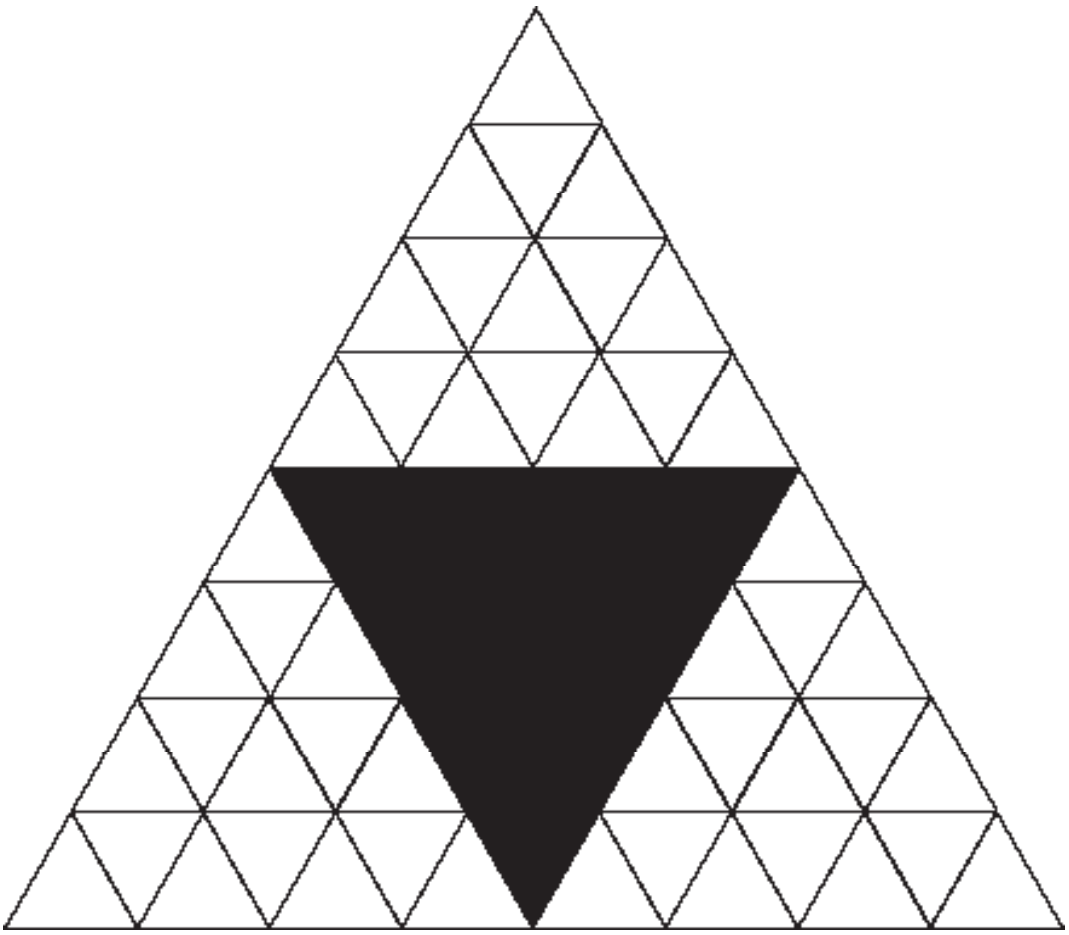
Sierpinski-Dreieck

Ein Sierpinski-Dreieck zeichnet man so:
Zuerst zeichnet man ein gleichseitiges weißes Dreieck. Dann geht es schrittweise weiter.

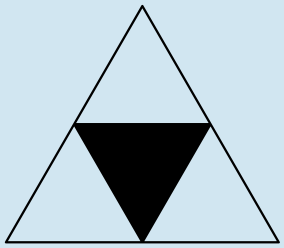
In jedem Schritt wird jedes weiße Dreieck in vier kleinere unterteilt
und das mittlere davon schwarz gefärbt:



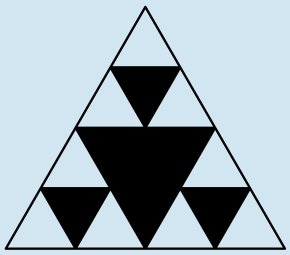
Wie sieht ein Sierpinski-Dreieck nach drei Schritten aus?
Färbe die richtigen Teildreiecke schwarz. Für den ersten Schritt ist das schon erledigt.



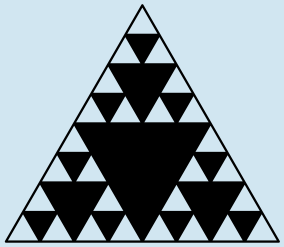
So ist es richtig:
Im ersten Schritt wird das ursprüngliche weiße Dreieck unterteilt. Das mittlere der vier bei der Unterteilung entstandenen kleineren Dreiecke wird schwarz gefärbt. Danach gibt es drei weiße Dreiecke.



Im zweiten Schritt werden die drei weißen Dreiecke unterteilt, und jeweils das mittlere (noch kleinere) Dreieck schwarz gefärbt. Jetzt gibt es neun weiße Dreiecke.



Im dritten Schritt werden nun die neun weißen Dreiecke unterteilt, mit einem schwarzen Teildreieck in der Mitte:



So sieht ein Sierpinski-Dreieck nach drei Schritten aus. Die Teildreiecke müssen also genau so gefärbt werden.

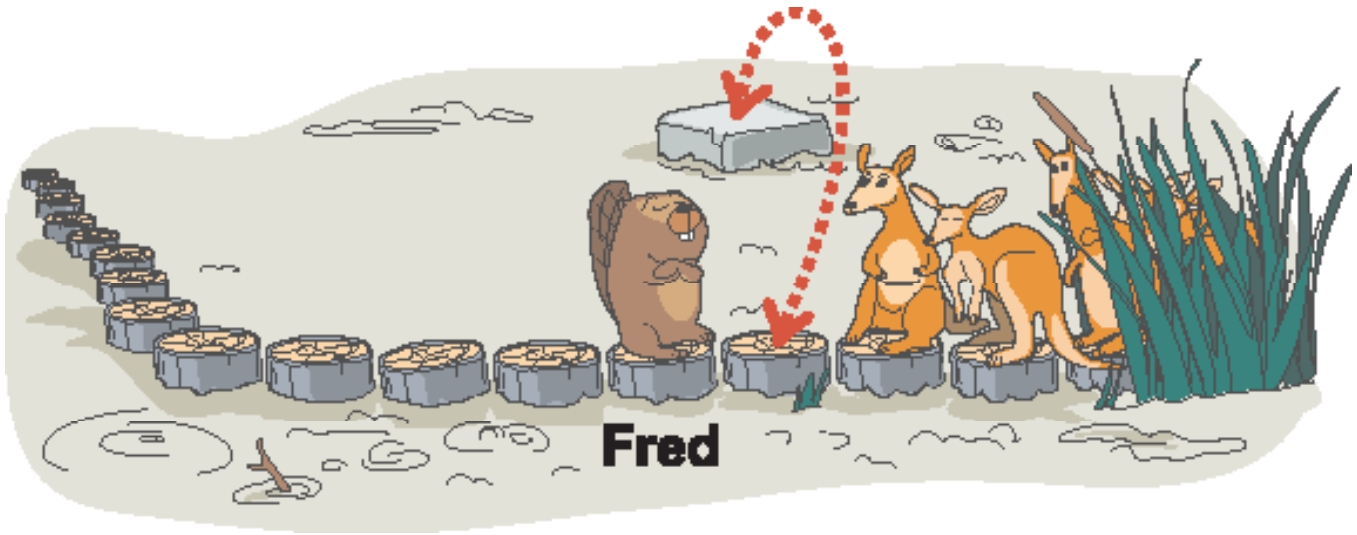
Das ist Informatik!
Das Sierpiński-Dreieck gehört zu den *Fraktalen,* einer besonderen Art geometrischer Objekte, und wurde zuerst vom polnischen Mathematiker Wacław Franciszek Sierpiński (1882-1969) im Jahr 1915 beschrieben. Die Konstruktion des Sierpiński-Dreiecks ist *rekursiv* (vom Lateinischen *recurrere:* zurücklaufen). Dies bedeutet, dass eine bestimmte Abfolge von Anweisungen (hier: das Teilen und Einfärben eines Dreiecks) mit den neu entstandenen Objekten wiederholt ausgeführt wird (der sogenannte *Rekursionsschritt*) bis eine bestimmte *Abbruchbedingung* erfüllt ist (zum Beispiel, wie oft die Dreiecke geteilt werden). Eine *rekursive Definition* erlaubt eine sehr knappe Beschreibung von recht komplexen Objekten (siehe auch die Aufgaben „Digitale Bäume“ und „Zahlenmaschine“ in diesem Biberheft).
Das Sierpiński-Dreieck und andere Fraktale wie die Koch-Schneeflocke oder die Mandelbrot-Menge wurden im 20. Jahrhundert in der Informatik intensiv untersucht, insbesondere seit es möglich wurde Computer einzusetzen, um die einzelnen Rekursionsschritte automatisch zu berechnen.

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

Sturer Fred

Über den Fluss führt ein Pfad aus Baumstümpfen.
Der Biber Fred geht über den Pfad und begegnet einer Gruppe Kängurus.
Der Biber und die Kängurus können nicht aneinander vorbei,
denn auf jeden Baumstumpf passt nur ein Tier.

Aber es gibt einen Baumstumpf, von dem aus ein Känguru
auf einen Stein und wieder zurück springen kann.
Auch auf den Stein passt nur ein Tier.



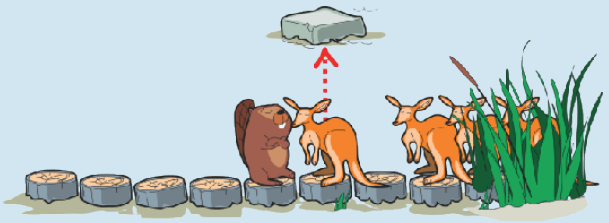
Mit Hilfe des Steins könnte Fred alle Kängurus vorbei lassen.
Aber Fred ist stur. Er will höchstens 10-mal einen Baumstumpf zurück gehen.
Vorwärts geht er beliebig oft.

Wie viele Kängurus kann Fred höchstens vorbei lassen?

A) Mehr als 10 Kängurus.
B) Genau 10 Kängurus.
C) Genau 6 Kängurus.
D) Genau 4 Kängurus.
E) Weniger als 4 Kängurus.
F) Das kann man nicht genau sagen.

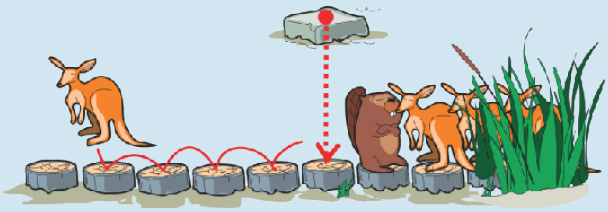
Antwort C ist richtig:
Fred kann höchstens genau 6 Kängurus vorbei lassen.
So kann Fred genau ein Känguru vorbei lassen:

Das Känguru springt auf den Stein.



Fred geht zwei Baumstümpfe vorwärts.

Das Känguru springt zurück und setzt seinen Weg fort.



Fred geht zweimal einen Baumstumpf zurück. Dann ist er wieder auf der Ausgangsposition und kann das Verfahren wiederholen, um jeweils ein weiteres Känguru vorbei zu lassen.

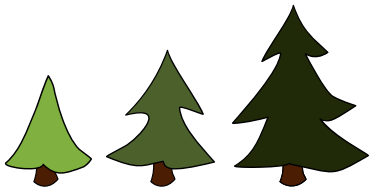
Da Fred höchstens 10-mal einen Baumstumpf zurück gehen will, kann er nach dem ersten Känguru höchstens 5 weitere Kängurus vorbei lassen, insgesamt also enau 6 Kängurus.

Das ist Informatik!
Die Lösung zu dieser Biberaufgabe haben wir oben als Algorithmus beschrieben: als Folge einfacher, klarer Anweisungen, die nach und nach ausgeführt werden, zum Beispiel „Fred geht zwei Baumstümpfe vorwärts“ oder „das Känguru springt auf den Stein“. Außerdem gibt es eine Wiederholungsanweisung, also die Anweisung, eine Folge anderer Anweisungen mehrfach auszuführen.

Mit Wiederholungsanweisungen, in der Informatik auch als *Schleifen* bezeichnet, können gleichförmige Aufgaben zuverlässig mehrfach erledigt werden. Dabei ist es in der Regel von Vorteil, bei jedem Schleifendurchlauf, also bei jeder Ausführung der zu wiederholenden Anweisungen, die gleiche Situation herzustellen – die sogenannte *Schleifeninvariante.* In dieser Biberaufgabe muss Fred jedesmal auf seine Ausgangsposition zurück, damit die die zu wiederholenden Anweisungen auch für das nächste Känguru wieder funktionieren.

Ein Algorithmus ist der Kern jedes Computersystems, denn Computer können nur nach klaren Anweisungen arbeiten. In der Informatik sind für sehr viele Probleme Algorithmen bekannt und entwickelt worden, mit deren Hilfe Computer diese Probleme lösen können.

Tannen-Sudoku

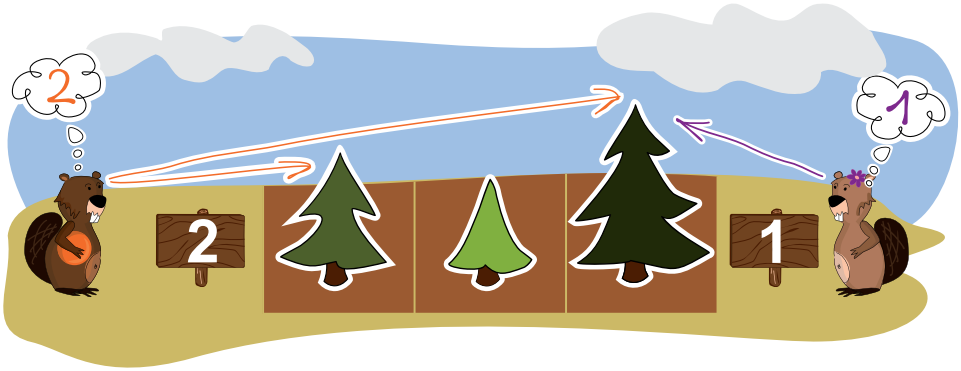


Die Biber möchten Tannen auf ein Feld setzen.
Die Tannen haben drei unterschiedliche Höhen.
Wenn eine Tanne hinter einer größeren Tanne steht, kann man sie nicht sehen.

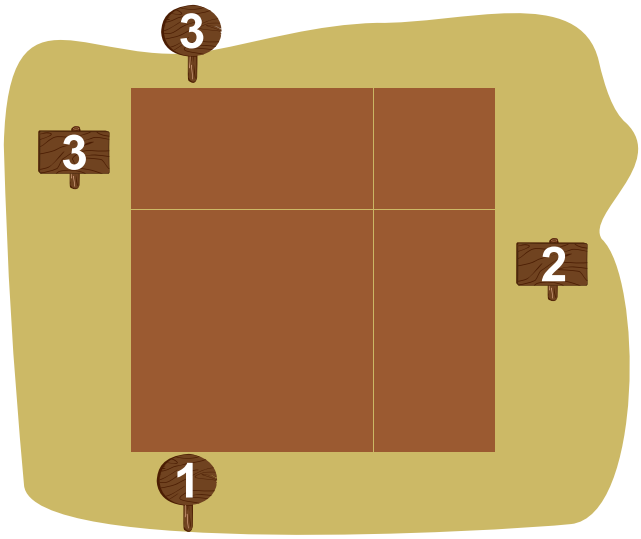
Das Feld hat Reihen mit je 3 Plätzen für Tannen:
Drei Reihen waagerecht (von links nach rechts),
drei Reihen senkrecht (von oben nach unten).

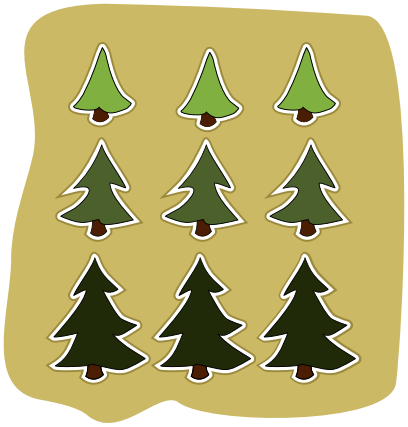
Am Ende einer Reihe steht manchmal ein Schild.
Darauf steht, wie viele Tannen in dieser Reihe man vom Schild aus sehen soll.
Die Tannen müssen so gesetzt werden, dass alle Schilder stimmen.
Und: In jeder Reihe (waagerecht und senkrecht) müssen alle Tannenhöhen vorkommen.

Hier ist ein Beispiel mit einer Reihe.
Alle Tannen stehen richtig.



Setze alle Tannen auf das Feld.
Am Ende müssen alle Tannen richtig stehen.

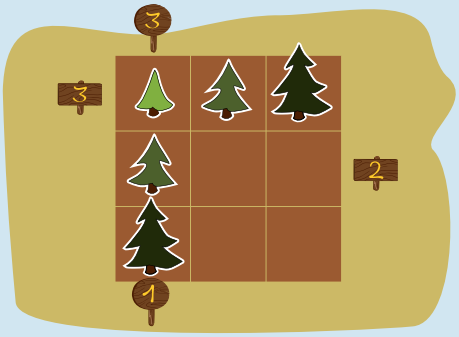




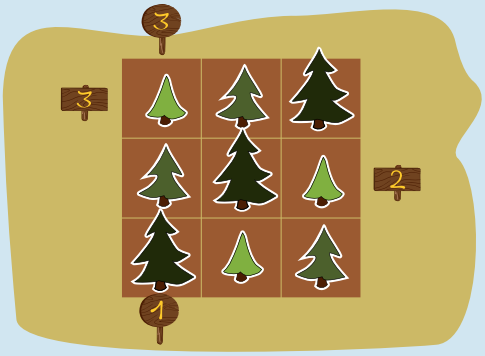
So ist es richtig:
Die beiden Schilder mit der 3 sagen, dass man drei Tannen sehen soll
- von links in der oberen waagerechten Reihe und
- von „oben“ in der linken senkrechten Reihe.

Alle drei Tannen einer Reihe kann man nur sehen,
wenn die Höhe der Tannen vom Schild aus gesehen ansteigt:

Die Tannen müssen also so auf die beiden Reihen gesetzt werden (und damit ist auch die Bedingung des Schildes unten links erfüllt):



Das Schild rechts mit der 2 sagt, dass von dort zwei Tannen sichtbar sind. Das geht nur, wenn die kleine Tanne auf den Platz beim Schild und die große Tanne dahinter gesetzt wird.
Die letzten beiden Plätze werden mit Hilfe der „Sudoku-Regel“ gefüllt, dass in jeder Reihe alle Tannenhöhen vorkommen müssen. So stehen alle Tannen richtig:

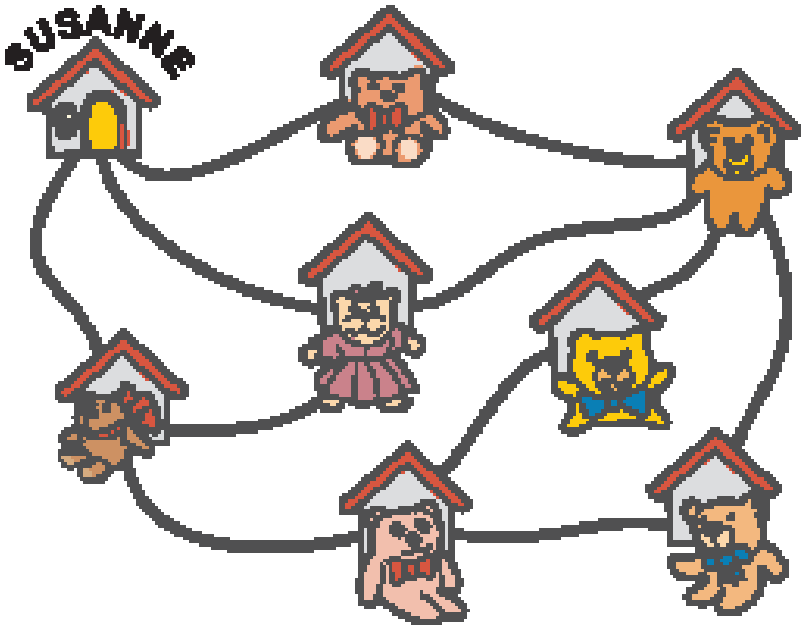


Das ist Informatik!
Wer hätte das gedacht? Vier „Sichtbarkeits-Zahlen“ und die Tannen-Sudoku-Regel in dieser Biberaufgabe genügen, um die Anordnung von neun Tannen auf dem Feld eindeutig zu bestimmen. Auch bei üblichen Sudoku-Rätseln kann die Platzierung von nur wenigen Zahlen genügen, damit ein Rätsel eindeutig gelöst werden kann. Daten (wie die Befüllung der 81 Plätze eines Sudoku-Rätsels mit Zahlen) können oft stark reduziert werden, ohne die in den Daten enthaltene Information zu verlieren. In der Informatik spricht man dann davon, dass Daten verdichtet oder komprimiert werden. Das geht meist nur, wenn man Regeln oder Gesetzmäßigkeiten ausnutzen kann, die für bestimmte Datenarten gelten – wie die Tannen-Sudoku-Regel für das Tannenfeld. Bei der Komprimierung (oder auch: Kompression) von Bilddaten kann man zum Beispiel ausnutzen, dass die meisten Bilder bestimmte Farbmuster mehrfach enthalten.

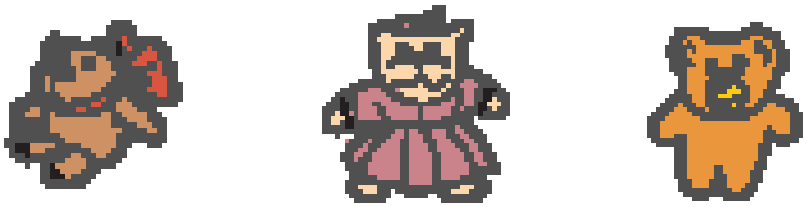
Kompression ist das tägliche Brot von Computersystemen und wird deshalb in der Informatik sehr gründlich erforscht. Ohne Kompression würden viel weniger Fotos, Songs oder Videos in den Speicher eines Smartphones oder auf die Festplatte eines Computers passen, als wir das gewohnt sind.

Teddybär

In Susannes Dorf sitzt vor jedem Haus ein Teddybär.

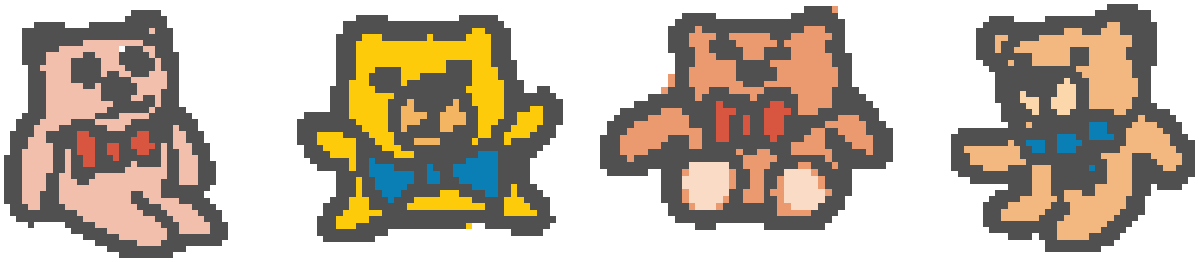


Susanne geht spazieren.
Sie geht zu Hause los, dann an vier anderen Häusern vorbei und zurück nach Hause.
Susanne hat also vier Teddys gesehen. Sie erinnert sich aber nur an diese drei:

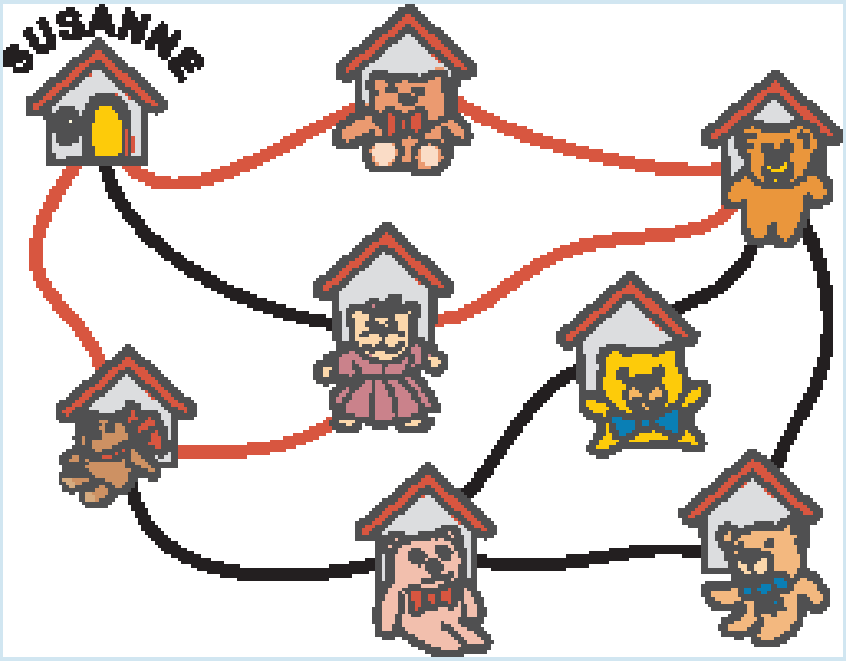


Welchen Teddybären hat Susanne noch gesehen?

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



Antwort C ist richtig:
Susanne macht einen Rundgang: Sie geht zu Hause los und kommt am Ende nach Hause zurück. Susanne kann viele verschiedene Rundgänge durch ihr Dorf machen. Der kürzeste Rundgang führt Susanne an nur zwei Häusern vorbei. Mehrere Rundgänge führen sie an mehr als vier Häusern vorbei. Aber es gibt nur einen Rundgang mit genau vier Häusern:



Auf diesem Rundgang geht es an den drei Teddybären vorbei, an die Susanne sich erinnert. Außerdem geht es beim Teddybär  vorbei. Den hat Susanne also auch noch gesehen.

Das ist Informatik!
Susanne ist ein Fehler passiert: Sie hat einen Teddy nicht „gespeichert“. Jetzt gibt es eine Lücke in der Information über ihre Erinnerung. In dieser Biberaufgabe können wir die Lücke schließen, weil wir zusätzliche Information haben: Wir wissen, wo die Teddys sitzen und dass Susanne einen Rundweg an vier Häusern entlang gemacht hat. Das ist ähnlich wie bei Lückentexten aus dem Deutschunterricht oder Mathematikaufgaben mit leeren Feldern. Weil wir die Deutsch-Regeln kennen oder wissen, wie man richtig rechnet, können wir auch dort die Lücken füllen.

Auch in der Informatik kommen Fehler und Lücken vor. Beim Übertragen oder Speichern von Daten können Fehler passieren. Aber es ist wichtig, dass die von Computern gespeicherten Daten stimmen. Deshalb wurden in der Informatik Methoden entwickelt, diese Fehler zu erkennen und zu korrigieren. Wenn der Fehler nicht zu groß ist, kann man das schaffen, indem man zusätzliche Information überträgt oder speichert – so wie in dieser Biberaufgabe.

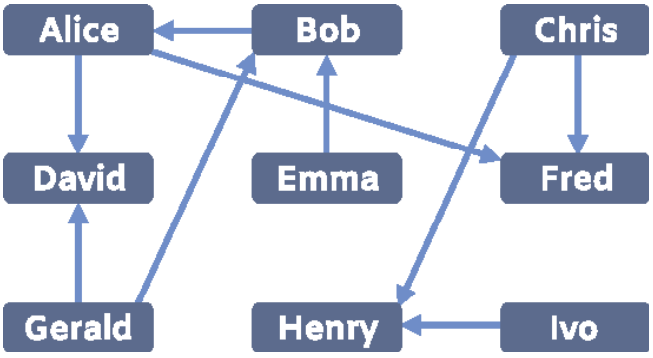
Turniersieger

In einem Freiluft-Sportturnier wurden schon einige Partien gespielt.
Im Bild unten zeigen die Pfeile, wer dabei gegen wen gewonnen hat.
Zum Beispiel hat Bob gegen Alice gewonnen und Alice gegen David.

Leider wird das Wetter schlecht.
Um die Anzahl der noch nötigen Partien zu verringern, wird eine neue Regel aufgestellt:
Es soll derjenige Turniersieger werden, der besser ist als alle anderen.

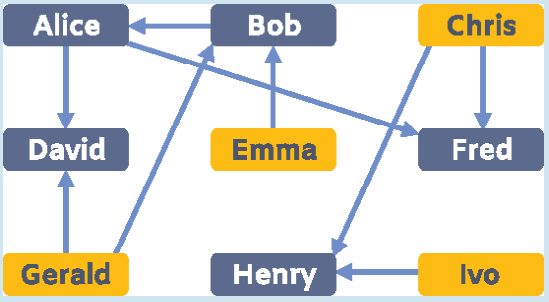
Wenn A gegen B gewonnen hat, dann gilt:
- A ist besser als B.
- Wenn B besser ist als einige andere Spieler, ist auch A besser als diese Spieler.

Jetzt sollen nur noch die Spieler weitermachen, die nach der neuen Regel noch Turniersieger werden können.



Welche Spieler können noch Turniersieger werden?

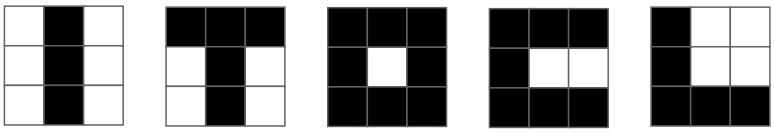
So ist es richtig:
Jeder Spieler (oder jede Spielerin), für die es keinen anderen Spieler gibt, der besser ist, kann selbst am Ende noch besser sein als alle anderen – und noch Turniersieger werden. Für alle Spieler, die schon einmal verloren haben, gibt es (mindestens) einen besseren Spieler. Deshalb können noch alle Spieler Turniersieger werden, die bisher noch kein Spiel verloren haben. Also müssen im Bild alle Namen angeklickt werden, auf die kein Pfeil zeigt.



Das ist Informatik!
In dieser Biberaufgabe stellt sich das Problem, anhand einer Reihe von Vergleichen („besser als“) eine Sortierung aufzustellen: Es sind die Spieler zu bestimmen, die nach aktuellem Stand die Besten sind – bzw. für die es keine besseren Spieler gibt. Wir begegnen diesem Problem auch bei der Aufgabe „Am schwersten“ in diesem Biberheft, und in der Informatik ist es als „topologische Sortierung“ bekannt. Wenn nicht alle zu sortierenden Dinge miteinander verglichen wurden, kann es mehrere Sortierungen geben – so wie in dieser Biberaufgabe mehrere Spieler nach aktuellem Stand die Besten sind. Beim Sortieren hilft, wenn die Vergleiche „transitiv“ sind, so wie es in dieser Aufgabe die neue Regel festlegt: Wenn A besser ist als B und B besser als C, dann ist A auch besser als C. So ergeben sich aus direkten Vergleichen weitere, indirekte Vergleiche, und die Anzahl der möglichen Sortierungen sinkt. Topologische Sortierung kann einem auch im Alltag begegnen, etwa wenn es Dinge gibt, die in ihrer Reihenfolge voneinander abhängen: Wer sich Socken und Schuhe anziehen will, sollte die Socken vor den Schuhen anziehen; das Hemd ist vor der Jacke an der Reihe; und wenn das T-Shirt vor dem Hemd drankommt, ist es transitiv auch vor der Jacke dran. Ob aber die Socken oder das T-Shirt zuerst angezogen werden, ist egal; es sind also mehrere Sortierungen möglich.

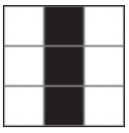
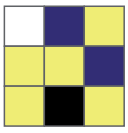
Unterscheidung

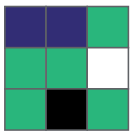
Eine Maschine erkennt genau diese Pixel-Bilder, und zwar als Buchstaben I, T, O, C und L:

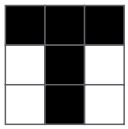


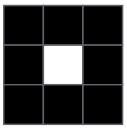
Dabei entsteht für jedes der fünf Bilder eine Unterscheidungskarte.
Die Unterscheidungskarte zeigt für jedes Bild-Pixel an dessen Position eine Farbe.
Die Farbe zeigt, wie viele der anderen Bilder an dieser Position das gleiche Pixel haben.
Je heller die Farbe, desto wichtiger ist dieses Pixel für die Unterscheidung:

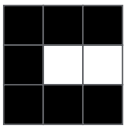
Farbe	So viele der anderen Bilder haben an dieser Position das gleiche Pixel:
	Keines (0)
	1
	2
	3
	Alle (4)

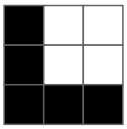
Ein Beispiel: Das Pixel-Bild  hat diese Unterscheidungskarte: 

Welches Pixel-Bild hat diese Unterscheidungskarte:  **?**

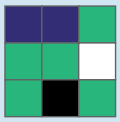
A) 

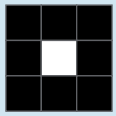
B) 

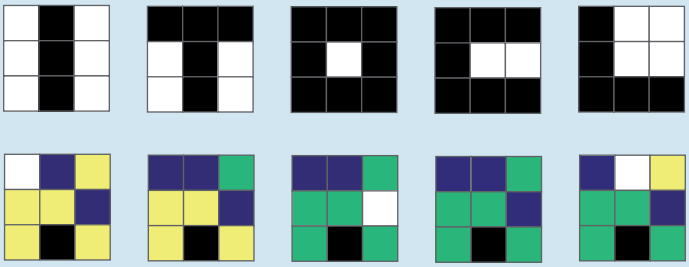
C) 

D) 

Antwort B ist richtig:

Die Unterscheidungskarte  zeigt in Reihe 2, Spalte 3 die Farbe Weiß. Das zugehörige Bild muss sich also in diesem Pixel von allen anderen Bildern unterscheiden.

Beim Bild aus Antwort B  ist dieses Pixel schwarz, bei allen anderen Bildern ist es weiß. Nur Bild B unterscheidet sich also in diesem Pixel von allen anderen Bildern.
Hier sind noch einmal alle Bilder, mit ihren Unterscheidungskarten darunter.



Das ist Informatik!
Die Unterscheidungskarten in dieser Biberaufgabe zeigen Werte als Farben an. Damit funktionieren sie so ähnlich wie die Temperaturkarten bei der Wettervorhersage: Dort wird der für einen Ort vorhergesagte Temperaturwert mit einer Farbe kodiert, häufig von Blau für niedrige Temperaturen bis Violett für (sehr) hohe Temperaturen. Mit Hilfe der Temperaturkarten (engl. „heat maps“) werden warme oder kalte Regionen gut erkennbar.

Aber auch viele andere Arten von Daten können als „heat map“, also mit Hilfe einer Farbkodierung, sehr gut veranschaulicht werden, insbesondere wenn die Daten eine räumliche Zuordnung haben.
In künstlichen neuronalen Netzen, die in der Informatik häufig als Kernmodul lernender Systeme verwendet werden, kann der Zustand der vielen aktiven Einheiten (der „Neuronen“) auch als heat map dargestellt werden. Dazu werden die Aktivierungswerte der Neuronen in einer Matrix angeordnet, bekommen also eine zweidimensionale räumliche Zuordnung. Je nach Kodierung können die Farben anzeigen, wie wichtig die einzelnen Neuronen für die Erkennung bestimmter Objekte sind – genau wie bei den Unterscheidungskarten.

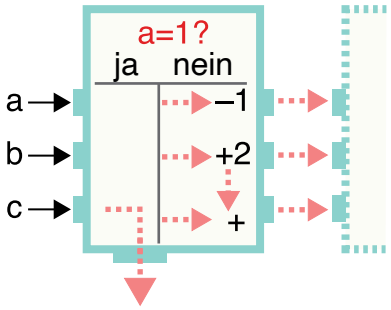
https://en.wikipedia.org/wiki/Heat_map

Zahlenmaschine

Die mysteriöse Zahlenmaschine erhält eine Zahl als Eingabe und gibt eine andere Zahl aus.

Im Inneren der Maschine arbeiten Einheiten.
Jede Einheit erhält drei Zahlen a, b, c als Eingabe und arbeitet nach diesen Anweisungen:

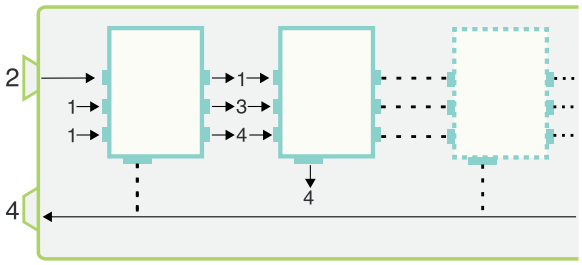
Wenn a eine 1 ist, gib c an die Zahlenmaschine als Ausgabe weiter.
Sonst mache Folgendes:
Gib a-1 als Eingabe a an die nächste Einheit weiter.
Gib B = b+2 als Eingabe b an die nächste Einheit weiter.
Gib B+c als Eingabe c an die nächste Einheit weiter.



Die Zahlenmaschine gibt ihre Eingabezahl als Eingabe a an die erste Einheit weiter.
Die Eingaben b und c der ersten Einheit sind jeweils 1.

Sobald die Zahlenmaschine von einer Einheit eine Ausgabe erhält, gibt sie diese Zahl als Ergebnis aus.

Das Bild zeigt, wie die Zahlenmaschine die Eingabezahl 2 verarbeitet. In diesem Fall verwendet sie zwei Einheiten. Als Ergebnis gibt sie die Zahl 4 aus.



Die Zahlenmaschine verarbeitet die Eingabezahl 4. Welche Zahl gibt sie als Ergebnis aus?

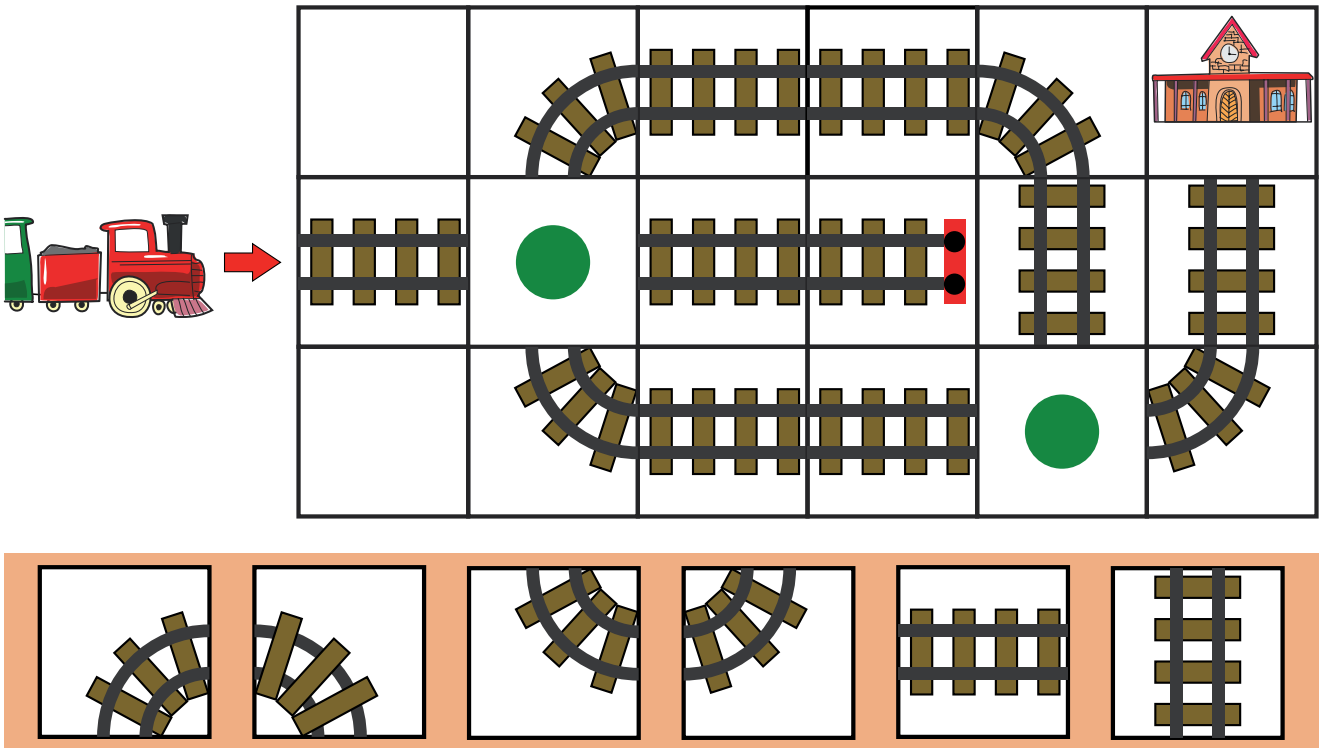
16 ist die richtige Antwort:
Die erste Einheit erhält als Eingabe (4,1,1) und gibt (3,3,4) weiter.
Die zweite Einheit erhält als Eingabe (3,3,4) und gibt (2,5,9) weiter.
Die dritte Einheit erhält als Eingabe (2,5,9) und gibt (1,7,16) weiter.
Die vierte Einheit erhält als Eingabe (1,7,16) und gibt die Zahl 16 an die Zahlenmaschine aus.
Die Zahlenmaschine gibt als Ergebnis die Zahl 16 aus.

Das ist Informatik!
In der Zahlenmaschine gibt eine Einheit ihre Ausgabe an eine andere „baugleiche“ Einheit weiter. Jede Einheit nimmt die ihr übergebenen Zahlen als Eingabe und verarbeitet sie auf exakt die gleiche Weise. Es werden so lange neue Einheiten ins Spiel gebracht, bis die erste übergebene Zahl eine 1 ist. Dies ist eine Bedingung für den Abbruch des gesamten Prozesses. Gäbe es diese Abbruchbedingung nicht, würden immer weiter neue Einheiten benötigt, und die Zahlenmaschine würde niemals anhalten. Berechnungsprozesse, in denen Recheneinheiten (wie zum Beispiel eine Berechnungsfunktion in einem Computerprogramm) eine oder mehrere weitere Kopien ihrer selbst zu weiteren Berechnung aufrufen, nennt die Informatik *rekursiv*. Rekursive Prozesse können häufig sehr einfach beschrieben werden, aber dennoch komplexe Berechnungen leisten. Bei der Realisierung als Computerprogramm kann eine rekursive Berechnung aber viel Speicherplatz beanspruchen: jede neue Einheit benötigt ihren eigenen Platz. Wenn aber jede Einheit immer nur eine Kopie rekursiv aufruft, kann der Prozess in der Regel auch als einfache Wiederholung (auch: Iteration) der immer gleichen Aktionen beschrieben und realisiert werden – wobei auch dann die Abbruchbedingung nicht vergessen werden darf. Ein iterativer Prozess verbraucht immer gleich viel Platz im Speicher, egal wie viele Wiederholungen benötigt werden.

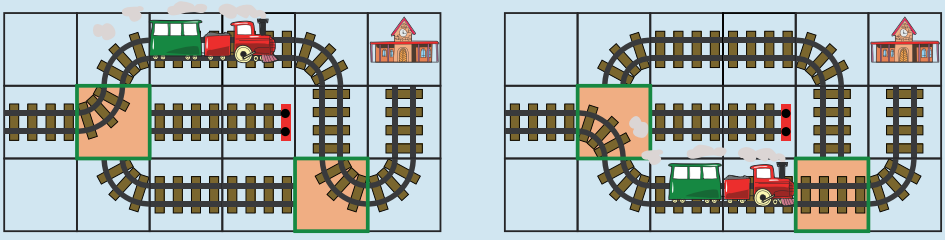
Zum Bahnhof

Ziehe Schienen auf die grünen Punkte,

so dass der Zug  zum Bahnhof  fahren kann.



So ist es richtig:
Es gibt zwei Möglichkeiten, Schienenstücke so auf die freien Stellen zu legen, dass der Zug zum Bahnhof fahren kann.



Bei allen anderen Möglichkeiten entgleist der Zug oder fährt gegen den Prellbock.

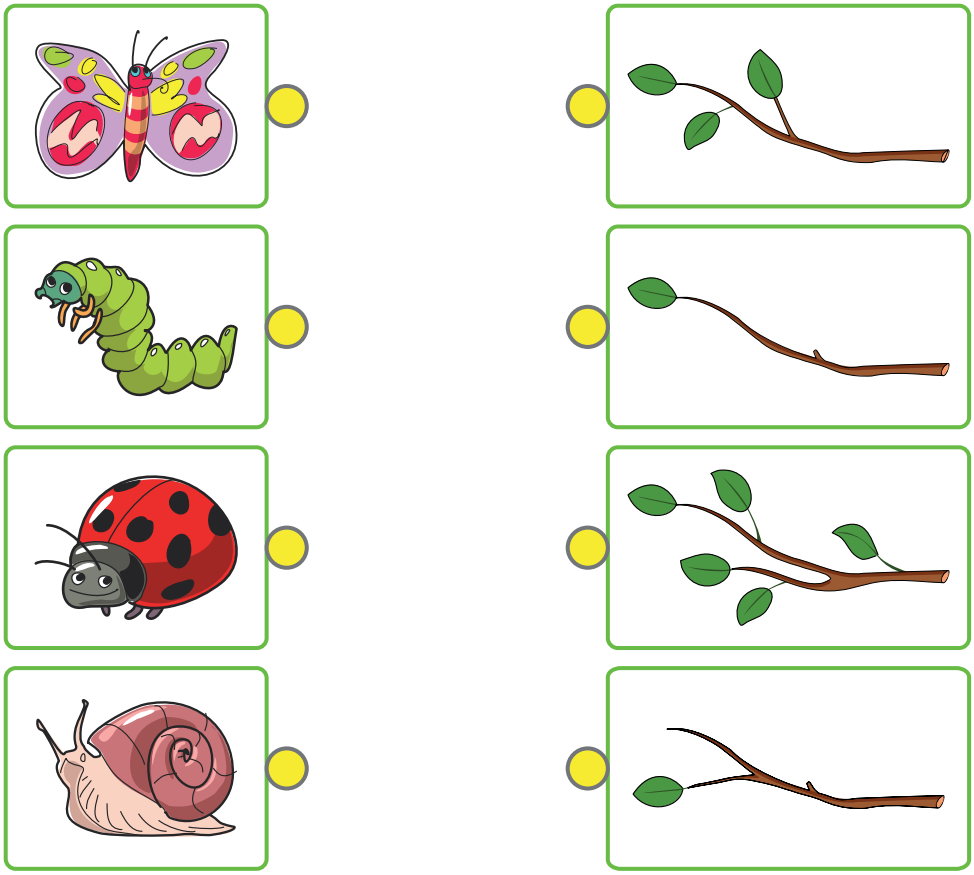
Das ist Informatik!
Wie der Zug in dieser Biberaufgabe genau den Schienen entlang fährt, führt ein Computer genau die Anweisungen eines Programms aus. Der Computer kann nicht erkennen, ob ein Programm einen Fehler enthält – so wie ein Zug nicht erkennen kann, ob Schienen falsch verlegt wurden. Informatikerinnen und Informatiker versuchen deshalb, in Programme „Notbremsen“ einzubauen, mit denen verhindert werden kann, dass Programmfehler Schäden verursachen. Wenn das nicht gelingt, kann der Computer „abstürzen“ und unbenutzbar werden, so wie ein Zug entgleisen kann. Wer ein Programm schreibt, muss also ähnlich sorgfältig sein wie Menschen, die Schienen für Züge verlegen.

Zweige

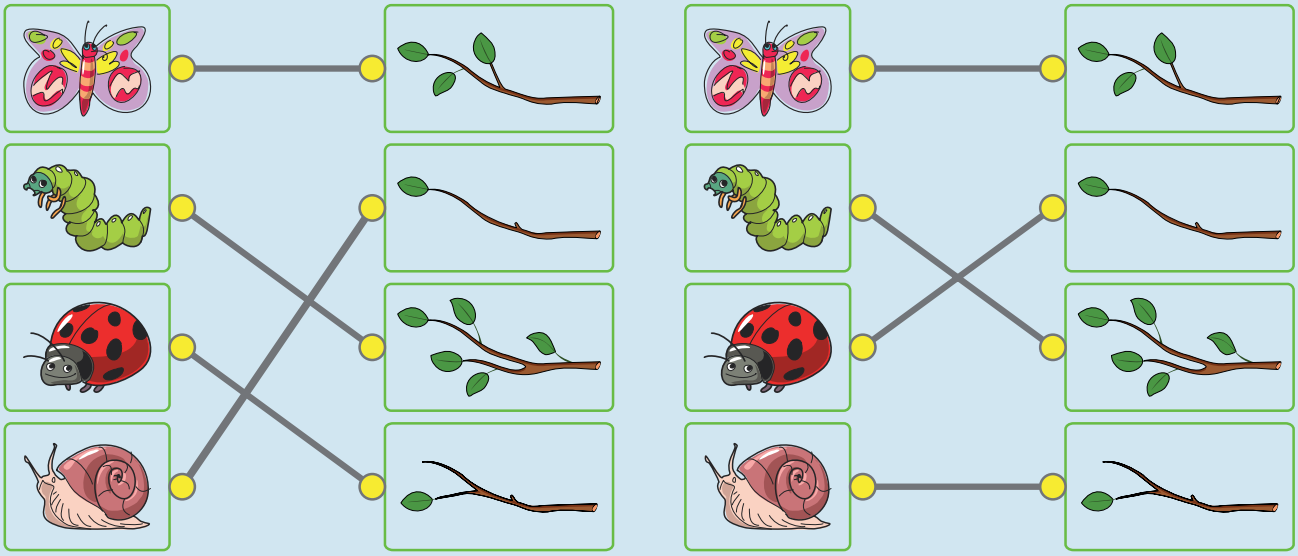
Emma und Felix entdecken in einem Busch vier Tiere.
Die Tiere sitzen auf vier verschiedenen Zweigen:

- Der Zweig mit der Raupe  hat die meisten Blätter.
- Der Zweig mit dem Schmetterling  hat mehr Blätter als der Zweig mit der Schnecke  .
- Der Zweig mit dem Marienkäfer  hat genau ein Blatt.

Verbinde jedes Tier mit seinem Zweig.



So ist es richtig:
Es gibt zwei richtige Antworten:



Der dritte Zweig von oben hat die meisten Blätter, nämlich fünf. Darauf sitzt die Raupe.
Der Marienkäfer sitzt auf einem Zweig mit einem Blatt. Das kann der zweite Zweig von oben sein, aber auch der untere Zweig. Es gibt also zwei richtige Lösungen.

Schmetterling und Schnecke sitzen auf den beiden anderen Zweigen. Der eine hat drei Blätter, und der andere hat ein Blatt. Die Schnecke sitzt auf dem zweiten Zweig mit einem Blatt. Dann sitzt der Schmetterling auf dem Zweig mit drei Blättern, also mit mehr Blättern als der Zweig mit der Schnecke.

Das ist Informatik!
Die richtige Zuordnung der Tiere zu den Zweigen hängt von der Anzahl der Blätter an den Zweigen ab. Eine besondere Rolle spielen Vergleiche zwischen diesen Blätteranzahlen, nämlich „mehr Blätter als“ und „die meisten Blätter“ (was so viel heißt wie „mehr Blätter als alle anderen“). Es wäre wohl einfacher gewesen, diese Biberaufgabe richtig zu beantworten, wenn die Zweige sortiert gewesen wären, nach der Anzahl der Blätter, von oben nach unten. Dann hätte man nämlich die Vergleichsergebnisse sehen können: Der Zweig mit den meisten Blättern wäre oben gewesen, und für einen Zweig oberhalb eines anderen wäre klar gewesen, dass er mehr Blätter hat als der andere.

Beim Umgang mit Daten, insbesondere mit vielen Daten, kann es sehr hilfreich sein, wenn die Daten sortiert sind. In der Kontaktliste auf dem Handy zum Beispiel sind die Daten alphabetisch sortiert; so ist es viel leichter, einen Kontakt zu finden, als wenn die Liste beliebig durcheinander wäre. Computer sind ständig mit dem Sortieren von Daten beschäftigt. Viele Informatikerinnen und Informatiker beschäftigen sich deshalb intensiv damit, wie Computer möglichst schnell auch große Datenmengen sortieren können.

Biber der Informatik 2020
Aufgaben und Lösungen

OESTERREICHISCHE COMPUTER GESELLSCHAFT®
AUSTRIAN COMPUTER SOCIETY
Ein Wettbewerb der OCG

ocg.at/biber