INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Aufgaben und Lösungen 2020

Alle Stufen

https://www.informatik-biber.ch/

Herausgeber:

Susanne Datzko, Fabian Frei, Juraj Hromkovič,
Regula Lacher, Jean-Philippe Pellet

SVIA
www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikinderausbildung//sociétésuissepourl'informatiquedansl'enseignement//societàsvizzeraperl'informaticanell'insegnamento

Mitarbeit Informatik-Biber 2020

Susanne Datzko, Fabian Frei, Martin Guggisberg, Lucio Negrini, Gabriel Parriaux, Jean-Philippe Pellet

Projektleitung: Nora A. Escherle

Herzlichen Dank für die Aufgabenentwicklung für den Schweizer-Wettbewerb an:
Juraj Hromkovič, Michael Barot, Christian Datzko, Jens Gallenbacher, Dennis Komm, Regula Lacher, Peter Rossmanith: ETH Zürich, Ausbildunges- und Beratungszentrum für Informatikunterricht

Die Aufgabenauswahl wurde erstellt in Zusammenarbeit mit den Organisatoren von Bebras in Deutschland, Österreich, Ungarn, Slowakei und Litauen. Besonders danken wir:
Valentina Dagienė: Bebras.org
Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend: Bundesweite Informatikwettbewerbe (BWINF), Deutschland
Wilfried Baumann, Anoki Eischer: Österreichische Computer Gesellschaft
Gerald Futschek, Florentina Voboril: Technische Universität Wien
Zsuzsa Pluhár: ELTE Informatikai Kar, Ungarn
Michal Winzcer: Comenius University, Slowakei

Die Online-Version des Wettbewerbs wurde auf cuttle.org realisiert. Für die gute Zusammenarbeit danken wir:
Eljakim Schrijvers, Justina Dauksaite, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers, Alieke Stijf, Kyra Willekes: cuttle.org, Niederlande
Chris Roffey: University of Oxford, Vereinigtes Königreich

Für den Support während den Wettbewerbswochen danken wir:
Hanspeter Erni: Schulleitung Sekundarschule Rickenbach
Gabriel Thullen: Collège des Colombières
Beat Trachsler: Kantonsschule Kreuzlingen

Christoph Frei: Chragokyberneticks (Logo Informatik-Biber Schweiz)
Dr. Andrea Leu, Maggie Winter, Brigitte Manz-Brunner: Senarclens Leu + Partner AG

Die deutschsprachige Fassung der Aufgaben wurde ähnlich auch in Deutschland und Österreich verwendet.

Die französischsprachige Übersetzung wurde von Elsa Pellet und die italienischsprachige Übersetzung von Christian Giang erstellt.

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Der Informatik-Biber 2020 wurde vom Schweizerischen Verein für Informatik in der Ausbildung SVIA durchgeführt und von der Hasler Stiftung unterstützt.

HASLERSTIFTUNG

Dieses Aufgabenheft wurde am 9. September 2021 mit dem Textsatzsystem LaTeX erstellt. Wir bedanken uns bei Christian Datzko für die Entwicklung und langjährige Pflege des Systems zum Generieren der 36 Versionen dieser Broschüre (nach Sprachen und Schulstufen). Das System wurde analog zum Vorgänger-System neu programmiert, welches ab 2014 gemeinsam mit Ivo Blöchliger entwickelt wurde. Jean-Philippe Pellet danken wir für die Entwicklung der `bebras` Toolchain, die seit 2020 für die automatisierte Konvertierung der Markdown- und YAML-Quelldokumente verwendet wird.

Hinweis: Alle Links wurden am 1. Dezember 2020 geprüft.

Vorwort

Der Wettbewerb «Informatik-Biber», der in verschiedenen Ländern der Welt schon seit mehreren Jahren bestens etabliert ist, will das Interesse von Kindern und Jugendlichen an der Informatik wecken. Der Wettbewerb wird in der Schweiz in Deutsch, Französisch und Italienisch vom Schweizerischen Verein für Informatik in der Ausbildung SVIA durchgeführt und von der Hasler Stiftung im Rahmen des Förderprogramms FIT in IT unterstützt.

Der Informatik-Biber ist der Schweizer Partner der Wettbewerbs-Initiative «Bebras International Contest on Informatics and Computer Fluency» (`https://www.bebras.org/`), die in Litauen ins Leben gerufen wurde.

Der Wettbewerb wurde 2010 zum ersten Mal in der Schweiz durchgeführt. 2012 wurde zum ersten Mal der «Kleine Biber» (Stufen 3 und 4) angeboten.

Der Informatik-Biber regt Schülerinnen und Schüler an, sich aktiv mit Themen der Informatik auseinander zu setzen. Er will Berührungsängste mit dem Schulfach Informatik abbauen und das Interesse an Fragenstellungen dieses Fachs wecken. Der Wettbewerb setzt keine Anwenderkenntnisse im Umgang mit dem Computer voraus – ausser dem «Surfen» im Internet, denn der Wettbewerb findet online am Computer statt. Für die Fragen ist strukturiertes und logisches Denken, aber auch Phantasie notwendig. Die Aufgaben sind bewusst für eine weiterführende Beschäftigung mit Informatik über den Wettbewerb hinaus angelegt.

Der Informatik-Biber 2020 wurde in fünf Altersgruppen durchgeführt:

- Stufen 3 und 4 («Kleiner Biber»)
- Stufen 5 und 6
- Stufen 7 und 8
- Stufen 9 und 10
- Stufen 11 bis 13

In den Altersklassen 3 und 4 hatten 9 Aufgaben zu lösen, nämlich aus den drei Schwierigkeitsstufen leicht, mittel und schwer jeweils drei. Für die Altersklassen 5 und 6 waren es je vier Aufgaben aus jeder Schwierigkeitsstufe, also 12 insgesamt. Für die restlichen Altersklassen waren es 15 Aufgaben, nämlich fünf Aufgaben pro Schwierigkeitsstufe.

Für jede richtige Antwort wurden Punkte gutgeschrieben, für jede falsche Antwort wurden Punkte abgezogen. Wurde die Frage nicht beantwortet, blieb das Punktekonto unverändert. Je nach Schwierigkeitsgrad wurden unterschiedlich viele Punkte gutgeschrieben beziehungsweise abgezogen:

	leicht	**mittel**	**schwer**
richtige Antwort	6 Punkte	9 Punkte	12 Punkte
falsche Antwort	−2 Punkte	−3 Punkte	−4 Punkte

Dieses international angewandte System zur Punkteverteilung soll den Anreiz zum blossen Erraten der Lösung eliminieren.

Jede Teilnehmerin und jeder Teilnehmer hatte zu Beginn 45 Punkte («Kleiner Biber»: 27 Punkte, Stufen 5 und 6: 36 Punkte) auf dem Punktekonto.

Damit waren maximal 180 Punkte («Kleiner Biber»: 108 Punkte, Stufen 5 und 6: 144 Punkte) zu erreichen, das minimale Ergebnis betrug 0 Punkte.

Bei vielen Aufgaben wurden die Antwortalternativen am Bildschirm in zufälliger Reihenfolge angezeigt. Manche Aufgaben wurden in mehreren Altersgruppen gestellt.

Für weitere Informationen:

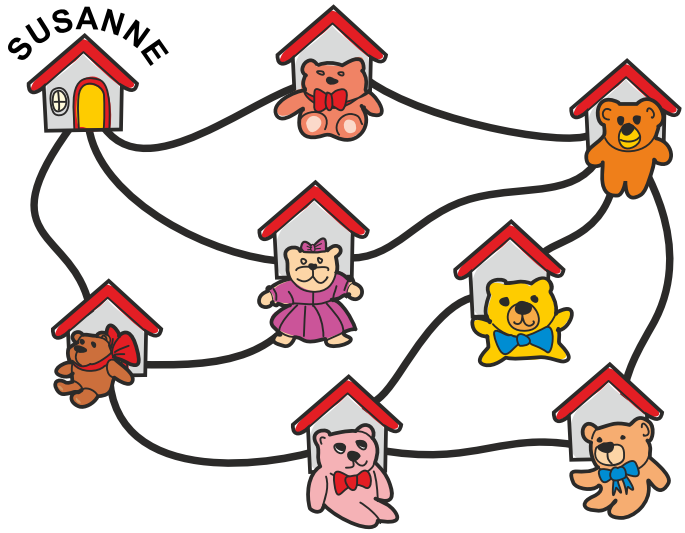
SVIA-SSIE-SSII Schweizerischer Verein für Informatik in der Ausbildung
Informatik-Biber
Nora A. Escherle

`https://www.informatik-biber.ch/de/kontaktieren/`
`https://www.informatik-biber.ch/`

Inhaltsverzeichnis

1. Teddybärenjagd

In Susannes Quartier sind folgende Teddybären vor den Häusern zu finden.



Susanne hat von ihrem eigenen Haus aus einen Rundgang an genau vier anderen Häusern vorbei gemacht. Sie ist an keinem Haus zweimal vorbeigegangen. Bei einem Haus hat sie den Teddybär übersehen. Die drei anderen Teddybären waren:

Welchen Teddybären hat Susanne übersehen?

A) 

B) 

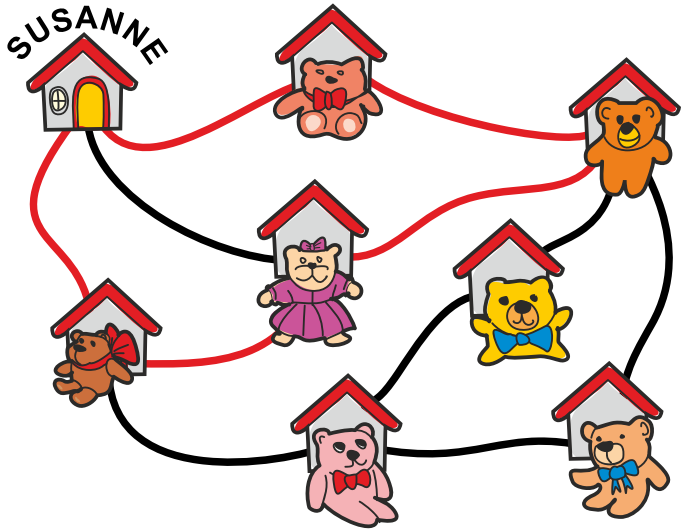
C) 

D) 

Lösung

Die richtige Antwort ist C) .

Bei ihrem Rundgang muss Susanne an den Häusern mit den drei Teddybären , und vorbeigegangen sein. Diese drei Teddybären sind direkt durch einen Weg verbunden. Zum ersten Teddybär kommt sie direkt von ihrem Daheim. Am Ende dieses Weges ist sie beim dritten Teddybär . Von dort aus gibt es nur einen Weg zurück zu ihrem Daheim, der über ein einziges viertes Haus geht, nämlich der Weg über den Teddybär . Andere mögliche Wege gehen an mindestens zwei weiteren Teddybären vorbei. Sie ist aber nur an vier Häusern vorbeigegangen. Die folgende Karte zeigt den Weg:



Susanne kann den Rundgang in beide möglichen Richtungen zurücklegen, das spielt keine Rolle.

Dies ist Informatik!

Lückentexte im Deutschunterricht, Mathematikaufgaben mit leeren Feldern oder eine Teddybärenjagd mit einem fehlenden Bild: das sind alles Aufgaben, bei denen eine fehlende Information gesucht ist. Die Aufgaben sind aber so aufgebaut (oder auch *strukturiert*), dass diese fehlende Information durch *logisches Denken* oder *Schlussfolgern* gefunden werden kann.

In der Informatik kommt so etwas immer wieder vor. Bei Datenübertragungen oder beim Speichern von Daten können Fehler passieren. Deshalb arbeitet man mit Verfahren zum *Erkennen von Fehlern* oder sogar zum *Korrigieren von Fehlern*. Wenn der Fehler nicht zu gross ist, schafft man das, indem man bewusst mehr Information speichert als eigentlich nötig. In dieser Aufgabe ist das die Karte und die Tatsache, dass Susanne genau an vier Teddybären vorbeiging. So kann sie die fehlende Information finden, nämlich welchen Teddybär sie übersehen hat.

Übrigens wurde in einigen Ländern der Welt im Jahr 2020 tatsächlich solche «Teddybärenjagden» (Englisch «Teddy Bear Hunt») angeboten, bei der in verschiedenen Häusern Teddybären versteckt wurde, die man von aussen her suchen konnte. Dies ermöglichte es den Kindern, sich gemeinsam am

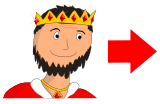
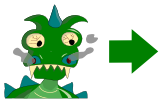
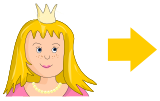
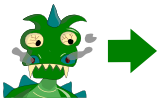
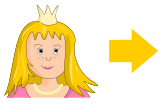
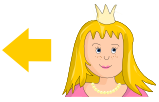
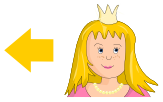
Verstecken und Entdecken zu erfreuen, trotz den Distanzregeln wegen des Corona-Virus. Die Idee, eine Bärenjagd nachzuspielen, stammt ursprünglich aus dem Bilderbuch «We're Going on a Bear Hunt» von Michael Rosen (1989). Bei uns kennt man das Buch und das zugehörige Spiel auch als «Wir gehen auf die Löwenjagd».

Stichwörter und Webseiten

- Fehlererkennung, Fehlerbehebung: `https://de.wikipedia.org/wiki/Fehlererkennung`
- Logisches Schlussfolgern
- Teddybärenjagd: `https://www.insider.com/coronavirus-pandemic-sparked-worldwide-bear-hunt-to-entertain-kids-2020-4`, `https://www.youtube.com/watch?v=0gyI6ykDwds`

2. Das Theaterstück

In einem Theaterstück spielen eine schöne Prinzessin, ein edler Ritter, der weise König und ein böser Drache mit. Am Anfang ist die Bühne leer. Während der Aufführung des Theaterstücks betreten und verlassen diese vier Figuren die Bühne in der folgenden Reihenfolge:

Erster Akt		**Zweiter Akt**
König betritt Bühne 		Drache betritt Bühne 
Prinzessin betritt Bühne 		Ritter betritt Bühne 
König verlässt Bühne 	P A U S E	Drache verlässt Bühne 
Drache betritt Bühne 		Prinzessin betritt Bühne 
Prinzessin verlässt Bühne 		Ritter verlässt Bühne 
Drache verlässt Bühne 		Prinzessin verlässt Bühne 
Pause		**Ende**

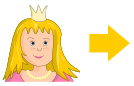
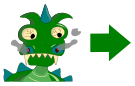
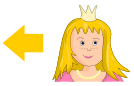
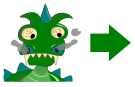
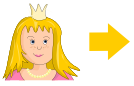
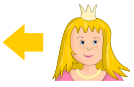
Was wird nicht passieren?

A) Die Prinzessin und der Ritter sind gemeinsam auf der Bühne.
B) Der König und der Drache sind gemeinsam auf der Bühne.
C) Der Ritter betritt die Bühne erst nach der Pause.
D) Der Ritter und der Drache sind gemeinsam auf der Bühne.

Lösung

Die richtige Antwort ist B) «Der König und der Drache sind gemeinsam auf der Bühne.», denn diese Behauptung stimmt während des ganzen Stückes nie.

Man kann sich das schrittweise überlegen:

Handlung	König auf Bühne?	Prinzessin auf Bühne?	Drache auf Bühne?	Ritter auf Bühne?	Übereinstimmung mit Behauptung der Antworten
Erster Akt					
	Ja	Nein	Nein	Nein	
	Ja	Ja	Nein	Nein	
	Nein	Ja	Nein	Nein	
	Nein	Ja	Ja	Nein	
	Nein	Nein	Ja	Nein	
	Nein	Nein	Nein	Nein	
Pause					
Zweiter Akt					
	Nein	Nein	Ja	Nein	
	Nein	Nein	Ja	Ja	C), D)
	Nein	Nein	Nein	Ja	
	Nein	Ja	Nein	Ja	A)
	Nein	Ja	Nein	Nein	
	Nein	Nein	Nein	Nein	
Ende					

Für jede Antwort kann geprüft werden, ob die darin gemachte Behauptung stimmt oder nicht, indem man die Zeilen der Tabellen durchgeht.

Bei der Antwort A) wird nach einer Zeile gesucht, in der sowohl die Prinzessin als auch der Ritter gemeinsam auf der Bühne sind. Das ist in der vierten Zeile des zweiten Aktes der Fall, denn dann betritt die Prinzessin die Bühne, wo der Ritter schon seit der zweiten Zeile ist und bis zur fünften Zeile bleibt. Die Behauptung von Antwort A) stimmt also zu mindestens einem Zeitpunkt.

Bei der Antwort D) wird nach einer Zeile gesucht, in der der Ritter und der Drache gemeinsam auf der Bühne sind. Das ist in der zweiten Zeile des zweiten Aktes der Fall, denn der Ritter betritt die Bühne in der zweiten Zeile, während der Drache die Bühne schon in der ersten Zeile betreten hat und bis zur dritten Zeile bleibt. Die Behauptung von Antwort D) stimmt also zu mindestens einem Zeitpunkt.

Bei der Antwort C) ist die Behauptung von einer anderen Art. Wenn diese stimmen soll, darf der Ritter die Bühne während des gesamten ersten Aktes nicht betreten haben. Hier muss man sich die Spalte des Ritters für den ersten Akt anschauen. Hier steht überall «Nein», also hat der Ritter die Bühne tatsächlich während des gesamten ersten Aktes nicht betreten. Er betritt sie dann aber in der zweiten Zeile des zweiten Aktes, also stimmt die Behauptung von Antwort C) ebenfalls.

Wenn die Behauptung von Antwort B) stimmen würde, müssten der König und der Drache in irgendeiner Zeile gemeinsam auf der Bühne stehen. Doch in keiner der zwölf Zeilen steht in beiden Spalten ein «Ja». Es ist sogar so, dass der König bereits in der dritten Zeile des ersten Aktes die Bühne verlässt und sie bis zum Ende nicht mehr betritt. Der Drache hingegen betritt die Bühne erst in der vierten Zeile des ersten Aktes. Vielleicht begegnen sich die beiden hinter der Bühne, aber auf der Bühne sind sie nie gemeinsam. Damit stimmt die Behauptung von Antwort B) nicht. Also ist B) die korrekte Antwort.

Dies ist Informatik!

Auch wenn man sich aufgrund des Ablaufs des Theaterstücks lebhaft eine Geschichte vorstellen kann, ist in dieser Aufgabe für jede Figur immer nur eine Eigenschaft wichtig: Befindet sie sich zu einem bestimmten Zeitpunkt auf der Bühne oder nicht? Dieses Einschränken des Blicks auf bestimmte Eigenschaften nennt man *Abstraktion*.

In der Informatik kann man solche Abstraktionen sehr gut formulieren. Für jede der vier Figuren definieren wir eine sogenannte *Variable*, die uns die Frage beantwortet, ob sich die Figur gerade auf der Bühne befindet. Die vier Variablen sind: «König auf Bühne?», «Prinzessin auf Bühne?», «Drache auf Bühne?» und «Ritter auf Bühne?». Während des Stückes ändern sich die Antworten auf diese Fragen immer wieder; für jede Frage ist die Antwort manchmal «ja» und manchmal «nein». In der Informatik nennen wir die aktuelle Antwort auf eine Frage den aktuellen *Wert* der zugehörigen Variablen. Der Wert einer Variablen kann sich in der Informatik also immer wieder ändern. (In der Mathematik ist das anders, dort ändern Variablen ihre Werte nicht über die Zeit hinweg.) Die Tabelle in der Antworterklärung zeigt die vier Variablen und die zugehörigen Werte zu jedem Zeitpunkt.

Es gibt noch eine andere Weise, das Theaterstück zu betrachten. Zu jedem Zeitpunkt schauen wir, welche Figuren gerade auf der Bühne sind. (Wir betrachten also die momentanen Werte der vier Variablen.) Jede mögliche Kombination von Figuren nennen wir einen *Zustand* der Bühne. Wenn

eine Figur die Bühne betritt oder verlässt, ändert sich also der Zustand der Bühne. Wir nennen das dann auch einen *Übergang* der Bühne von einem Zustand in einen anderen. Wenn man ein Blatt Papier nimmt und sich für jeden möglichen Zustand (also jede Figurenkombination) einen separaten Kreis zeichnet, kann man das Ganze als eine Abstraktion der Bühne betrachten.

Zusätzlich kann man noch die möglichen Übergänge als Pfeile einzeichnen, die von einem Zustand zu einem anderen führen. Wenn wir das auch noch tun, haben wir das, was wir in der Informatik den *Zustandsgraphen* der Bühne nennen würden.

Zu Beginn des Theaterstückes ist die Bühne leer. Den entsprechenden Zustand nennen wir deshalb *Anfangszustand*. Den Ablauf des Theaterstückes können wir jetzt als einen Weg im Zustandsgraphen einzeichnen. Der Weg beginnt im Anfangszustand und geht dann denjenigen Pfeilen entlang, die der Handlung entsprechen.

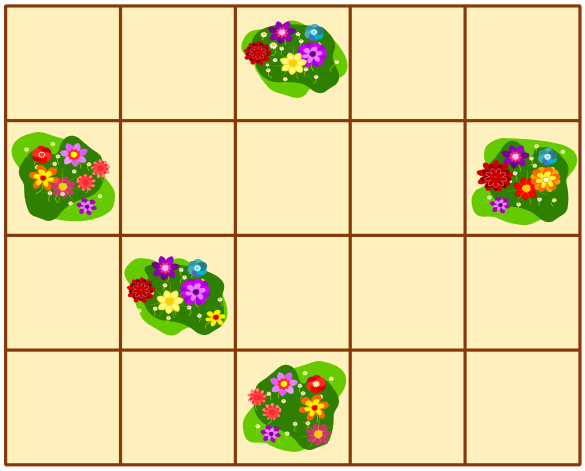
Zustandsgraphen sind in der Informatik sehr wichtig. Bei fast jedem komplizierten System muss man irgendwann über den Zustandsgraphen nachdenken. Für Menschen ist es aber oft sehr mühsam, mit solchen abstrakten Zuständen und Übergängen zu arbeiten. Computer können das hingegen extrem gut. Daher lohnt es sich, wenn Menschen ihre Probleme so mit Zustandsgraphen darstellen können, dass Computer sie dann lösen können.

Stichwörter und Webseiten

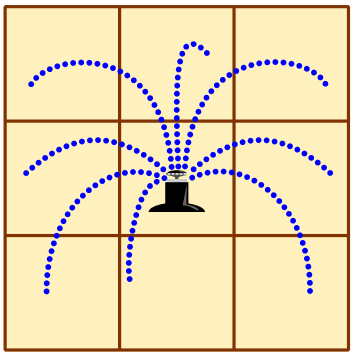
- Variablen: `https://de.wikipedia.org/wiki/Variable_(Programmierung)`
- Zustände, Übergänge, Zustandsgraph: `https://de.wikipedia.org/wiki/Zustandsübergangsdiagramm`

3. Beete bewässern

Daniels Garten besteht aus quadratischen Feldern. In einigen dieser Felder hat er Blumen gepflanzt.



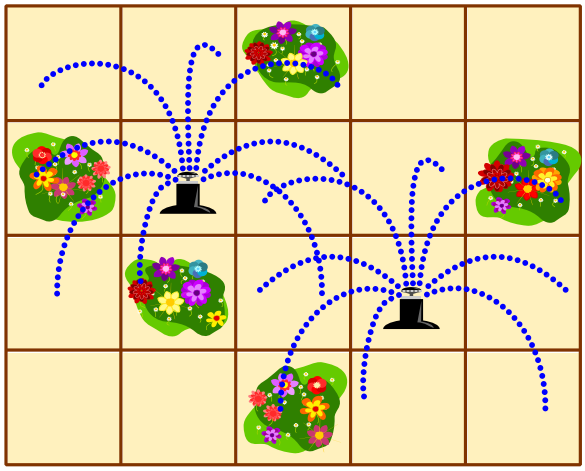
Im Sommer möchte er die Blumen mit Rasensprengern bewässern. Auf die Felder mit Blumen kann er keinen Rasensprenger stellen. Ein Rasensprenger bewässert alle Blumen in den 8 Feldern um ihn herum:



Platziere so wenige Rasensprenger wie nötig, um alle Blumenfelder zu bewässern.

Lösung

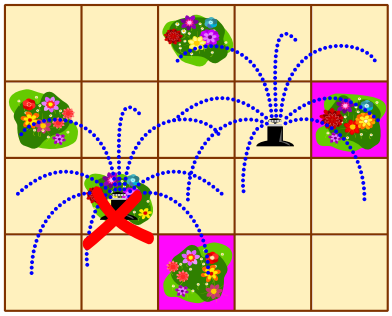
Die folgende Lösung braucht zwei Rasensprenger, um alle Blumenfelder zu bewässern:



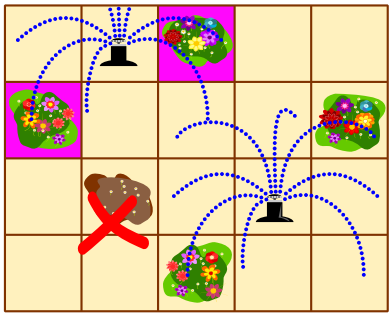
Zwischen dem Blumenfeld ganz links und dem Blumenfeld ganz rechts liegen drei Felder. Ein einzelner Rasensprenger kann keine Felder bewässern, die so weit auseinander liegen.

Es gibt auch keine weitere Lösung mit nur zwei Rasensprengern.

Um das Blumenfeld ganz rechts und das unten in der Mitte gleichzeitig zu bewässern, muss ein Rasensprenger genau da stehen, wo er in der Lösung steht. Wenn er eines höher stehen würde, um das Blumenfeld oben in der Mitte auch noch zu bewässern, würde er das Blumenfeld unten in der Mitte nicht mehr bewässern und man könnte die verbleibenden drei Blumenfelder nicht mit einem Rasensprenger bewässern, denn auf einem Blumenfeld darf kein Rasensprenger stehen.



Um das Blumenfeld ganz links und das oben in der Mitte zu bewässern, müsste ein Sprinkler entweder so stehen, wie in der Lösung gezeigt, oder ein Feld darüber. Wenn dieser Sprinkler aber zusätzlich noch das Blumenfeld in der zweiten Spalte von links und in der dritten Zeile von oben bewässern soll, kann er nicht ganz oben stehen.



Dies ist Informatik!

Diese Aufgabe ist ein typisches Optimierungsproblem: Während klar ist, dass alle Blumenfelder bewässert werden sollen, ist die Anzahl der benötigten Rasensprenger variabel und sollte möglichst klein sein. Ähnliche Optimierungsprobleme tauchen auf, wenn man beispielsweise Ortschaften mit Feuerwehrstationen absichern oder Höfe mit Natelempfang versorgen möchte.

In der Informatik spricht man auch von *Überdeckungsproblemen*. Diese gehören zu einer Klasse von sehr schwierigen Problem in der Informatik. Die richtige Platzierung einer minimalen Anzahl von Rasensprengern war in der Aufgabe zwar noch recht einfach. Doch die Schwierigkeit steigt mit der Anzahl an Blumenfeldern so stark an, dass man bald einmal keine optimale Lösung mehr finden kann in vernünftiger Zeit, selbst mit Computerunterstützung.

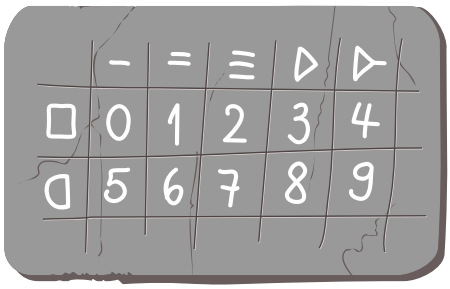
Eine Möglichkeit in solchen Fällen ist es dann, dass man sich mit Lösungen zufrieden gibt, die vielleicht nicht optimal sind, aber immer noch gut. Es macht keinen grossen Unterschied, ob man jetzt 101 statt nur 100 Feuerwehrstationen oder 1000 Natelsendemasten statt nur 990 positioniert; das Problem ist dadurch aber oft viel leichter zu lösen.

Stichwörter und Webseiten

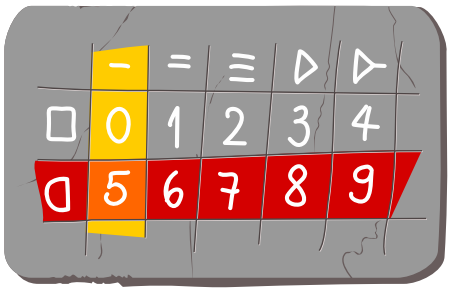
- Optimierung: `https://de.wikipedia.org/wiki/Optimierungsproblem`
- Überdeckungsproblem

4. Baujahr der Biberburg

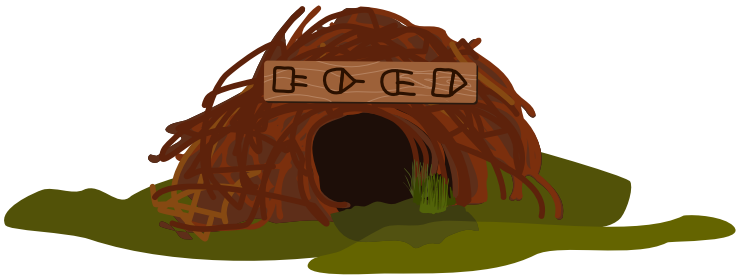
Auf dem Schild über dem Eingang jeder Biberburg steht das Baujahr. Die Biber verwenden für die Ziffern eigene Zeichen. Die Tabelle rechts zeigt, wie man aus den Ziffern die Zeichen der Biber zusammensetzen kann:



Beispielsweise setzen die Biber die Ziffer «5» so zu dem neuen Zeichen  zusammen:



So sieht Cleverias Biberburg aus:



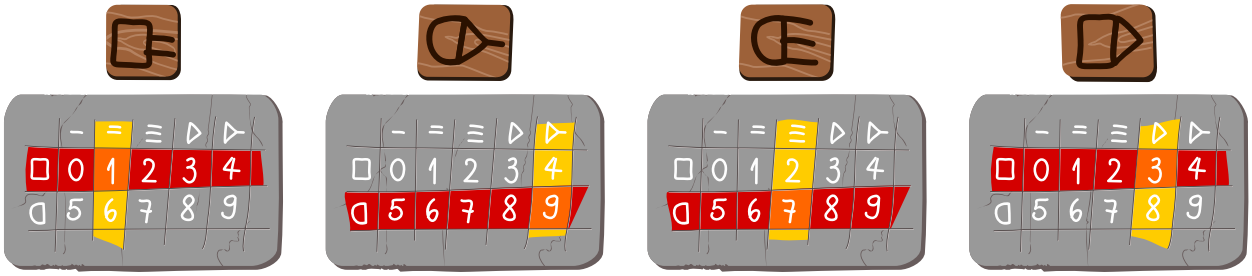
In welchem Jahr wurde Cleverias Biberburg gebaut?

A) 0978
B) 1574
C) 1923
D) 1973
E) 1993
F) 2973
G) 6378

Lösung

Das Baujahr der Biberburg findest du heraus, indem du für jedes Symbol die entsprechende Zeile und die entsprechende Spalte bestimmst. An der Kreuzung der Spalte und der Zeile findest du die gesuchte Ziffer.

Weil es vier Symbole sind, machst du das vier Mal.



Die vier Ziffern in der richtigen Reihenfolge ergeben die Zahl 1973.

Dies ist Informatik!

Das Geheimhalten von Informationen und das Schützen von Daten ist eine 4000 Jahre alte Aufgabe. Unzählige Geheimsprachen wurden zu diesem Zweck entwickelt und benutzt. Heute ist Datensicherheit eines der Kernthemen der Informatik. Eine der Methoden, Daten vor unbefugtem Lesen zu schützen, ist sie *zu chiffrieren*. Das Chiffrieren verwandelt einen *Klartext* in einen *Geheimtext*. Das Rekonstruieren des Klartextes aus dem Geheimtext nennt man *Dechiffrieren*. Die Lehre der Geheimschriften nennt man *Kryptologie*.

Die antiken Kulturen verwendeten meistens Geheimschriften, die durch Codierung von Buchstaben mit anderen Buchstaben oder ganz neuen Zeichen erzeugt worden sind. Die Geheimschrift hier ist speziell für den Informatik-Biber entwickelt worden, basiert aber auf einem Konzept aus dem antiken Palästina. Die damalige Sicherheitsregel war, dass nur Geheimschriften verwendet worden sind, die man leicht auswendig lernen kann. Eine schriftliche Beschreibung der Geheimschrift aufzubewahren, betrachtete man als zu grosses Risiko. Eine Tabelle, wie sie hier verwendet wird, kann man gut auswendig lernen. Die berühmte Geheimschrift der Freimaurer basiert auf diesem Prinzip.

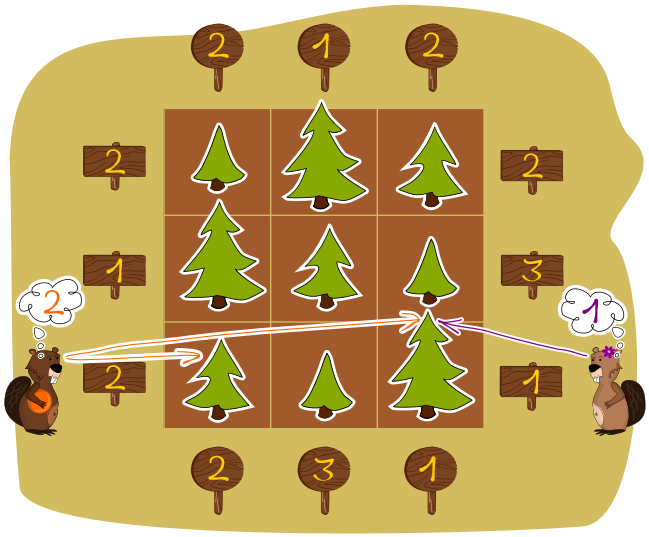
Statt nur neue Zeichen für Ziffern zusammenzustellen, kann man kann auch eigene neue Geheimschriften für Texte erfinden. Dazu schreibt man in eine Tabelle alle Buchstaben und erfindet neue Symbole für die Spalten und Zeilen und erhält so für alle Buchstaben neue Zeichen.

Stichwörter und Webseiten

- Kryptographie (Substitution): `https://de.wikipedia.org/wiki/Kryptographie`
- Geheimschriften

5. 3×3-Tannen-Sudoku

Biber pflanzen Tannen in Reihen. Die Tannen haben drei unterschiedliche Höhen (1 , 2 und 3) und in jeder Reihe gibt es genau eine Tanne von jeder Höhe.



Wenn sich die Biber eine Tannenreihe von einem Ende her anschauen, dann können sie niedrigere Tannen, die hinter höheren Tannen versteckt sind, **nicht** sehen.

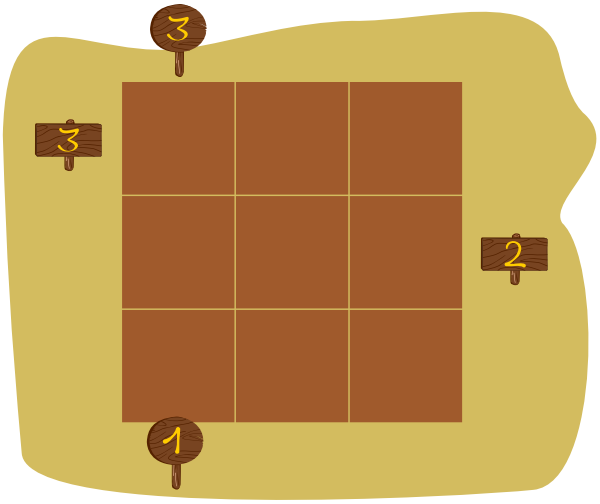
Am Ende jeder Tannenreihe steht auf einem Schild, wie viele Tannen ein Biber von dieser Stelle sehen kann.

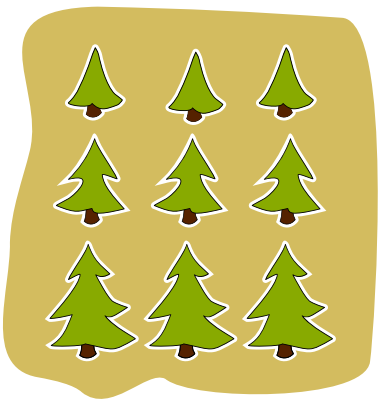
Nun pflanzen die Biber neun Tannen in ein 3×3-Feld, wie im Beispiel rechts.

Dabei gelten folgende Regeln:

- In jeder Zeile (horizontalen Reihe) gibt es genau eine Tanne von jeder Höhe.
- In jeder Spalte (vertikalen Reihe) gibt es genau eine Tanne von jeder Höhe.
- Die Schilder mit der Anzahl sichtbarer Tannen stehen rund um das 3×3-Feld.

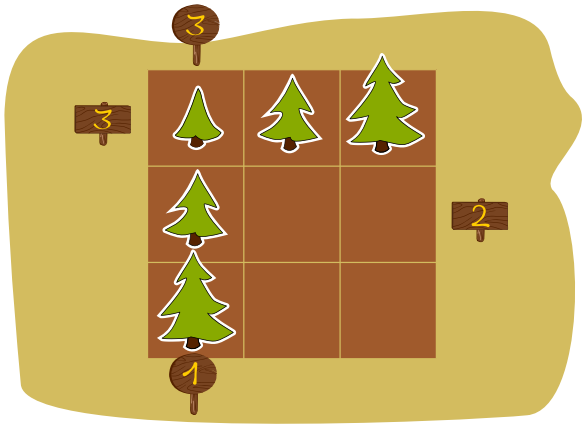
Verteile die Tannen auf die richtigen Felder.





Lösung

Im Feld zeigen zwei Schilder, dass von diesen Positionen drei Tannen gesehen werden können. Alle drei Tannen einer Reihe kann man nur sehen, wenn die Tannen so geordnet sind, dass ihre Höhe ansteigt, also von dieser Position weg. Damit sind die Spalte links und die oberste Zeile bestimmt:

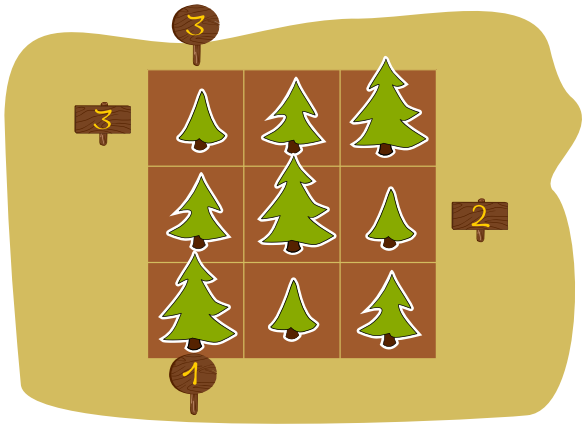


Das Schild rechts mit der 2 verlangt, dass von dort zwei Tannen sichtbar sind, also muss ganz in der Mitte eine Tanne der Höhe 3 sein und diese mittlere Zeile ist somit (), 3 (), 1 ().

Die weiteren Felder werden gemäss der «Sudoku»-Regel gefüllt, dass von jeder Höhe genau eine Tanne in jeder Reihe sein muss.

In der Mitte der untersten Zeile muss eine Tanne der Höhe 1 () stehen, weil für in der mittleren Spalte die beiden anderen Höhen bereits vergeben sind. Ganz rechts unten muss schliesslich eine Tanne der Höhe 2 () folgen, um die Reihe vollständig zu machen.

Die vollständige Lösung sieht so aus:



Dies ist Informatik!

Diese Aufgabe fokussiert auf drei grundlegenden Kompetenzen von Informatikerinnen und Informatikern.

Zuerst geht es darum, eine Lösung zu finden, die gegebene Einschränkungen einhält, oder nach Bedarf einen Lösungsvorschlag zu korrigieren.

Zweitens geht es um die Fähigkeit, Objekte über ihre Darstellung aus Teilinformationen rekonstruieren zu können. Das hängt mit der Generierung von Objekten (Objektdarstellungen) aus eingeschränkten verfügbaren Informationen zusammen, wenn man die Gesetzmässigkeit der Objekte kennt. Solche Vorgehensweisen kann man auch bei der Komprimierung anwenden.

Drittens kann man solche Baumfelder mit Schildern zur Erzeugung von selbst-verifizierenden Codierungen einsetzen. Vorkommende Fehler beim Eintragen oder beim Informationstransport können dann automatisiert erkannt oder sogar korrigiert werden.

Stichwörter und Webseiten

- Sudoku: `https://de.wikipedia.org/wiki/Sudoku`
- Fehlererkennung und Fehlerkorrektur:
  `https://de.wikipedia.org/wiki/Fehlerkorrekturverfahren`
- Rekonstruktion von Objekten aus Teilinformationen
- Überprüfung der Korrektheit von Datendarstellungen

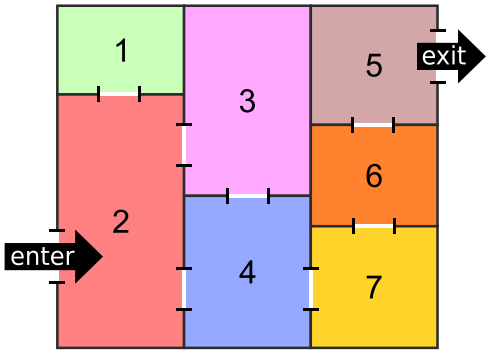
6. Museumsrundgang

Für ein neues Museum werden vier Grundrisse für die Räume vorgeschlagen. Jeder Grundriss enthält die sieben Räume 1 bis 7. Die Räume sollen so sein, dass die Besucher alle Räume besuchen können, ohne dabei einen Raum zweimal zu betreten.

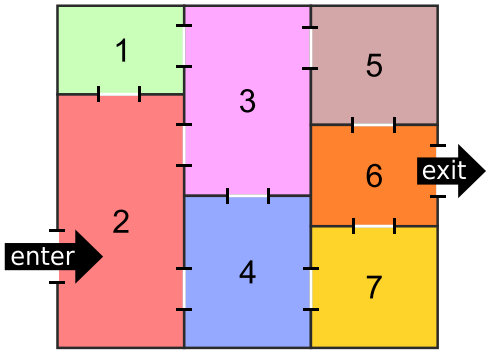
Die Besucher starten den Besuch bei «enter» und verlassen das Museum bei «exit».

Welcher Grundriss erlaubt es den Besuchern, jeden Raum genau einmal zu betreten und zu verlassen?

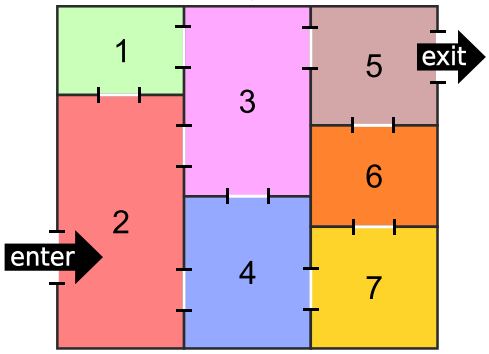
A)



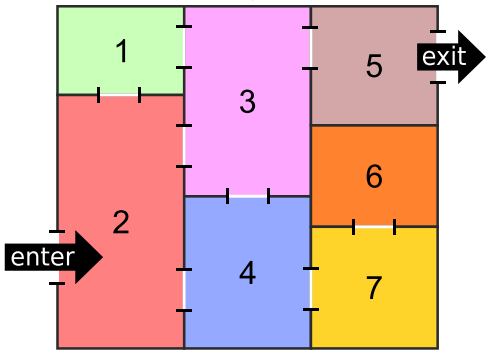
B)



C)

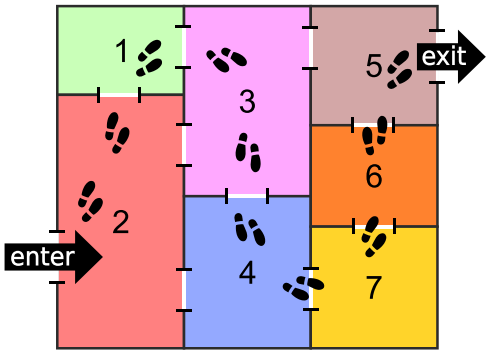


D)



Lösung

Nur der Grundriss C erlaubt es den Besuchern, jeden Raum genau einmal zu betreten und zu verlassen. Die Reihenfolge der Räume ist dabei: 2, 1, 3, 4, 7, 6, 5.



Generell ist eine solche immer Tour unmöglich, falls irgendeiner der Räume nur einen Eingang hat. Die Erklärung ist folgende: Falls ein Besucher diesen Raum betritt, muss er beim Verlassen dieses Raumes wieder zurück in den Raum, aus dem er gekommen ist, und verletzt dabei die Regel, dass er jeden Raum nur einmal betreten soll.

Im Grundriss A hat der Raum 1 nur einen Eingang.

Im Grundriss D hat der Raum 6 nur einen Eingang.

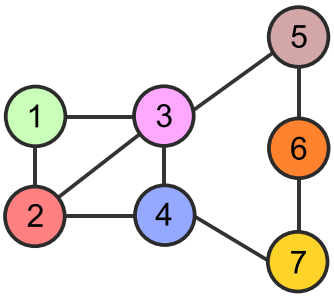
Im Grundriss B kann der letzte Raum 6 von Raum 5 oder von Raum 7 erreicht werden. Falls der Besucher vom Raum 5 kommt, kann er den Raum 7 betreten, aber danach nur durch den Raum 6 den Ausgang erreichen (oder umgekehrt), was die Regeln wieder verletzt.

Dies ist Informatik!

Die meisten Kinder oder Jugendliche lösen das Problem ohne zusätzliche abstrakte Darstellungen durch Probieren. Damit verwenden sie zu gewissem Grad die allgemeine Strategie des *Backtrackings*. Sie erkennen mindestens, dass man aus erfolglosen Versuchen lernen kann und in diesem Fall zurückgehen kann, um eine andere Möglichkeit auszuprobieren. Gleichzeitig ist man mit dem wichtigen Konzept des *Nichtdeterminismus* konfrontiert, weil man häufig mehrere Möglichkeiten zur Auswahl hat.

Die Aufgabe ist ein Beispiel für ein bekanntes Problem in der Informatik, der Suche nach einem *Hamiltonschen Weg*. In einer abstrakten Darstellung als *Graph* entspricht jeder Raum einem *Knoten* und jede Tür zwischen zwei Räumen einer *Kante* zwischen beiden entsprechenden Knoten.

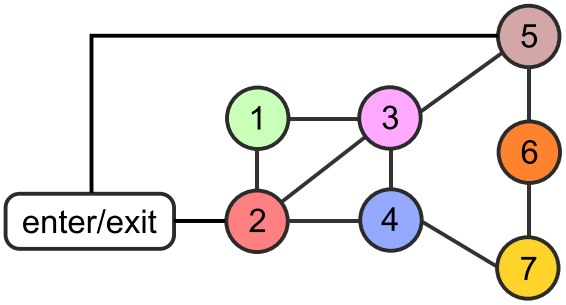
Abstrakt dargestellt sieht die Aufgabe folgendermassen aus:



Es geht jetzt darum, in diesem Graphen einen Weg mit folgenden Eigenschaften zu finden:

1. Der Weg startet in 2 («enter»).
2. Der Weg endet in 5 («exit»).
3. Jeder Knoten wird genau einmal besucht.

Falls man den Aussenraum zu einem Knoten zusammenfasst, dann entspricht das Ganze der Suche nach einem Hamiltonschen Kreis (einer Rundtour), wo auch jeder Knoten genau einmal durchlaufen wird und man am Ende dann aber wieder beim Anfangsknoten ist.



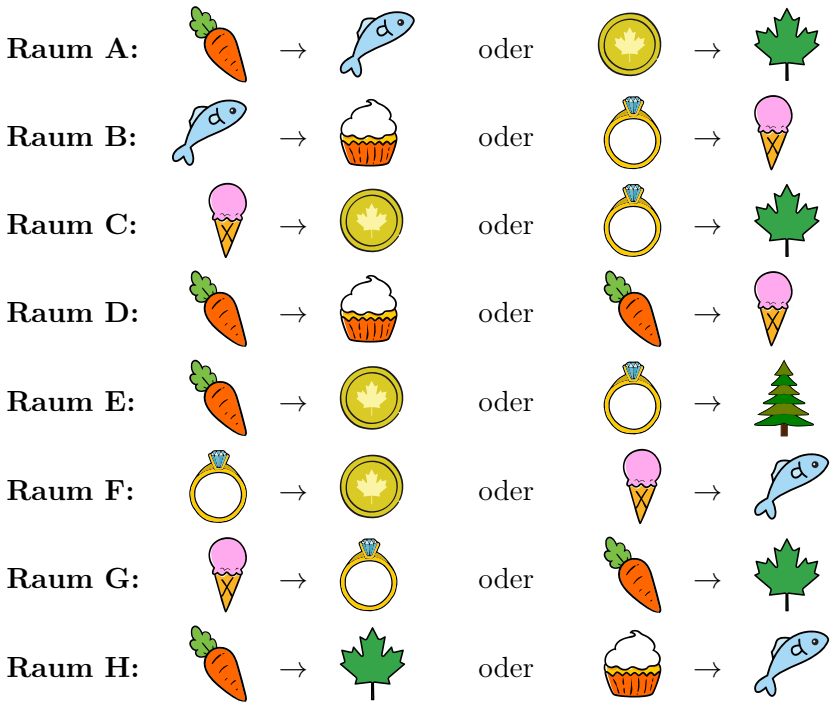
Stichwörter und Webseiten

- Graphentheorie, Knoten, Kante: `https://de.wikipedia.org/wiki/Graphentheorie`
- Hamiltonkreisproblem: `https://de.wikipedia.org/wiki/Hamiltonkreisproblem`

7. Biber im Schloss

Ein schlauer Biber braucht einen Tannenbaum um im Fluss einen Damm zu bauen. Leider hat er aber nur ein Rüebli . Im Schloss ist heute Markttag und der Biber will dort sein Rüebli gegen einen Tannenbaum eintauschen.

Jeder Raum des Schlosses bietet zwei Tauschangebote. Die Tabelle zeigt diese Angebote:



Zum Beispiel kann der Biber in Raum B für einen Ring ein Cornet bekommen, aber nicht umgekehrt.

In welcher Reihenfolge soll der schlaue Biber durch die Räume gehen, um letztlich den gewünschten Tannenbaum zu besitzen?

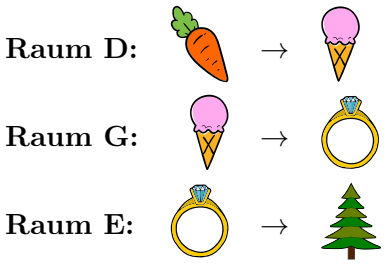
A) DGE: Zuerst Raum D, dann Raum G und zuletzt Raum E.
B) GGE: Zuerst Raum G, dann nochmal Raum G und zuletzt Raum E.
C) AGE: Zuerst Raum A, dann Raum G und zuletzt Raum E.
D) DBC: Zuerst Raum D, dann Raum B und zuletzt Raum C.

Lösung

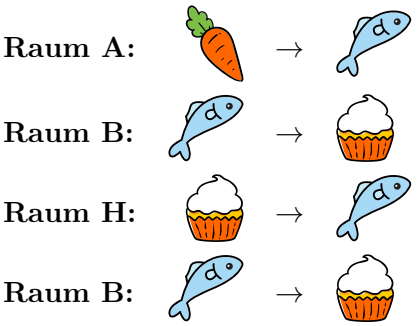
Die richtige Antwort ist A) DGE: Zuerst Raum D, dann Raum G und zuletzt Raum E.

Im Raum D tauscht der Biber sein Rüebli gegen ein Cornet. Danach geht er in Raum G, wo er das Cornet gegen einen Ring tauscht. Am Ende geht der Biber in den Raum E, um den Ring gegen einen Tannenbaum zu tauschen.

Diese Gesamtabfolge sieht so aus:



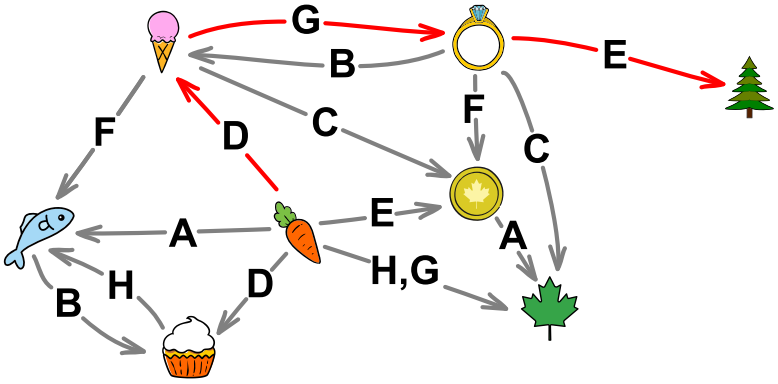
Um eine passende Reihenfolge der Räume zu finden, sind zwei unterschiedliche Strategien zielführend. Die erste Strategie versucht alle Tauschmöglichkeiten in Betracht zu ziehen. Sie beginnt damit, dass man beim ersten Tausch das Rüebli in fünf Räumen (A, D, E, G und H) gegen 6 verschiedene Objekte eintauschen kann. Danach werden für diese 6 Objekte wieder alle Eintauschmöglichkeiten betrachtet. Dies ist aufwendig und kann sogar im Kreis laufen, wie in folgendem Beispiel, wo der Biber beliebig oft die Räume B und H besuchen kann:



Somit ist diese erste Strategie sehr aufwendig und nur mit Glück in kurzer Zeit erfolgreich.

Die zweite Strategie führt in dieser konkreten Aufgabe schnell zum Ziel. Sie basiert darauf, die Suche vom gewünschten Ziel her, also dem Tannenbaum, zu beginnen. Nur im Raum E kann der Biber den gewünschten Tannenbaum bekommen. Den Tannenbaum bekommt man nur im Tausch gegen einen Ring. Das nächste gewünschte Objekt ist also ein Ring! Auch den Ring kann man nur in einem Raum bekommen, nämlich im Raum G im Tausch gegen ein Cornet. Ein Cornet erhält man entweder in Raum B gegen einen Ring oder in Raum D gegen ein Rüebli. Der schlaue Biber muss also seine Tauschvorgänge in Raum D beginnen.

Für eine bessere Übersicht kann die Tabelle der möglichen Tauschvorgänge als Graph mit gerichteten Kanten (Pfeilen) dargestellt werden. Jeder Knoten des Graphen steht für ein Tauschobjekt und jede ausgehende Kante steht für eine Tauschmöglichkeit. Zusätzlich steht auf der Kante, in welchem Raum diese Tauschmöglichkeit besteht.



Diese visuelle Darstellung der Tauschobjekte, Tauschmöglichkeiten und Räume erlaubt es, leicht herauszufinden, wie man vom Rüebli zum Tannenbaum kommt, nämlich auf einem Weg im gerichteten Graphen: Zuerst Raum D, dann Raum G und zuletzt Raum E.

Dies ist Informatik!

Berechnungsprozesse kann man auf einer allgemeinen Ebene betrachten als *Folgen von Transformationen* (hier sind es Tauschvorgänge) oder gleichwertig als *Folgen von Zuständen* eines Systems. Der Startzustand des Systems ist in unserem Fall das Rüebli und die Transformation (der *Berechnungsschritt*) vom Rüebli zum Cornet ändert diesen Zustand zum Cornet.

Ein Berechnungsschritt führt also von einem Zustand zu einem anderen. Eine Abfolge von Berechnungsschritten nennt man auch eine Berechnung.

Somit behandelt diese Aufgabe auch Berechnungen auf einer sehr allgemeinen Ebene. Das System ist im vorliegenden Fall *nichtdeterministisch*; das bedeutet, dass es manchmal mehrere mögliche Berechnungsschritte gibt, also wie in der Aufgabe mehrere Tauschmöglichkeiten. Nichtdeterminismus ist ein weiteres wichtiges Konzept der Modellierung in der Informatik. Die möglichen Berechnungsschritte werden durch *Transformationsregeln* (die Tabelle mit Tauschmöglichkeiten) beschrieben. Zu bestimmen, ob der Biber ein Rüebli in einen Tannenbaum eintauschen kann, also ob ein bestimmter *Zielzustand* des Systems von einem bestimmten *Startzustand* aus erreichbar ist, ist das berühmte *Erreichbarkeitsproblem* mit zahlreichen Anwendungen.

Die Aufgabe oben zeigt, dass es manchmal eine gute Idee ist, vom Zielzustand her den Startzustand zu suchen statt umgekehrt. Diese Strategie nennt man auch *Rückwärtssuche*.

Beim Vergleich der verschiedenen Lösungsstrategien erkennt man, dass der gerichtete Graph eine anschauliche Möglichkeit der Darstellung eines sogenannten *Zustandsraumes* eines Systems mit allen möglichen Übergängen zwischen Zuständen ist. In diesem Basismodell könnte man die bekannten grundlegenden *Suchalgorithmen* in Graphen, nämlich *Breitensuche* und *Tiefensuche* ansprechen.

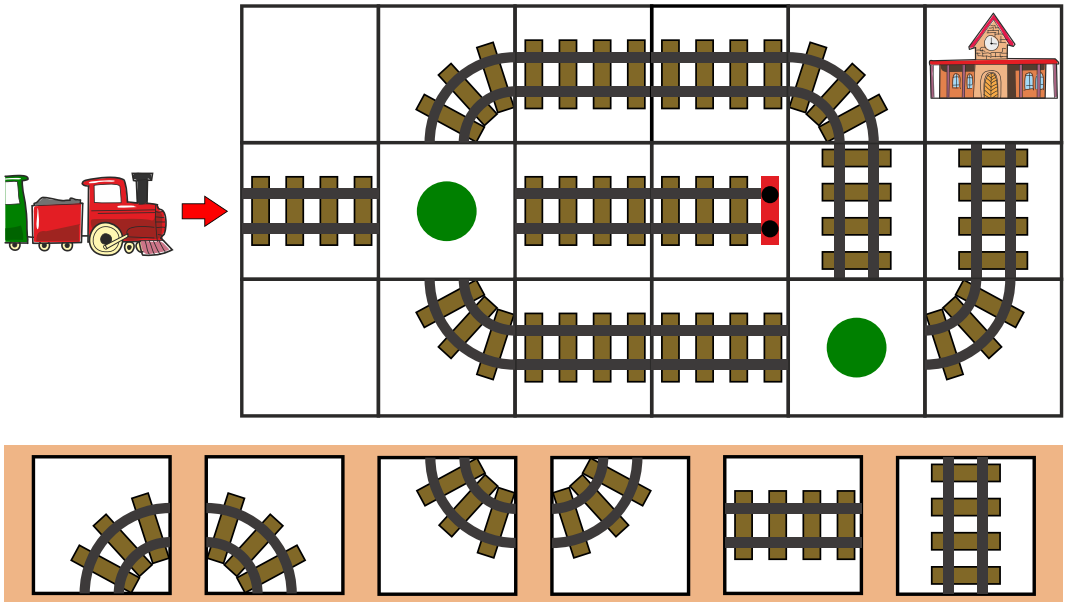
Stichwörter und Webseiten

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

- Erreichbarkeitsproblem:
  `https://de.wikipedia.org/wiki/Erreichbarkeitsproblem_in_Graphen`
- Tiefensuche: `https://de.wikipedia.org/wiki/Tiefensuche`
- Breitensuche: `https://de.wikipedia.org/wiki/Breitensuche`

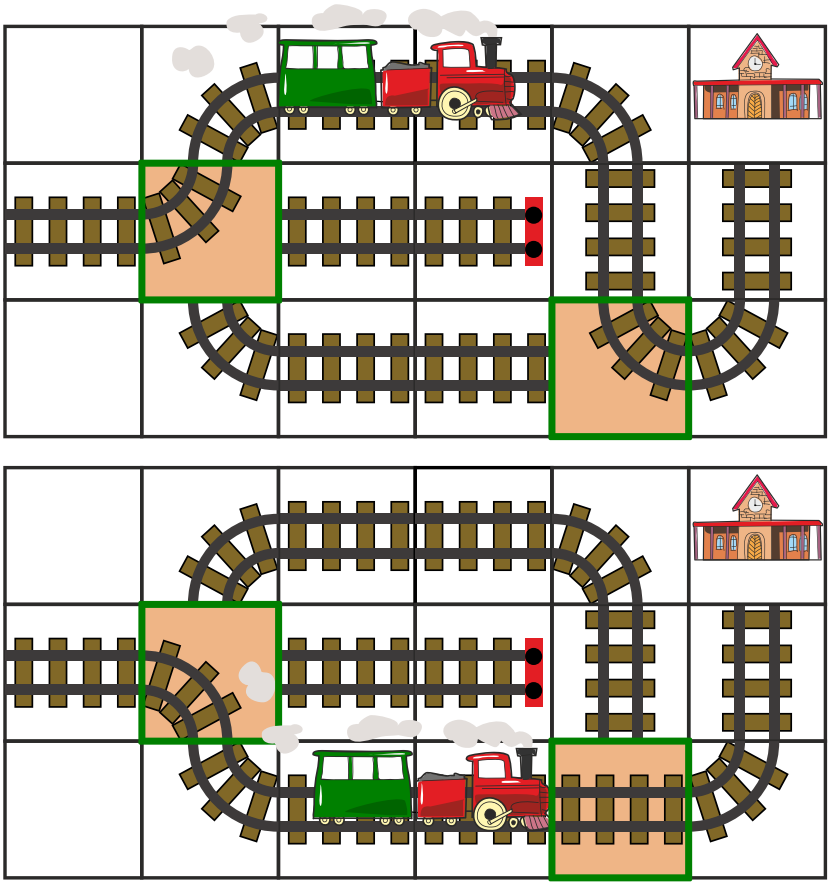
8. Nächster Halt, Bahnhof!

Lege Schienen auf die grünen Punkte, so dass der Zug zum Bahnhof fahren kann.



Lösung

Für dieses Problem gibt es folgende 2 Lösungen:



Bei anderen Kombinationen entgleist der Zug oder fährt gegen den Prellbock.

Dies ist Informatik!

Wie ein Zug stur entlang den Schienenstücken fährt, führt ein Computer stur die Anweisungen eines Programms aus. Er kann nicht erkennen, wenn das Programm einen Fehler enthält und kann dann «abstürzen», so wie ein Zug entgleisen kann, wenn die Schienen falsch gebaut wurden. Beim Schreiben eines Programms muss man also viel sorgfältiger sein, als wenn man beispielsweise einer Person den Weg zum Bahnhof erklärt.

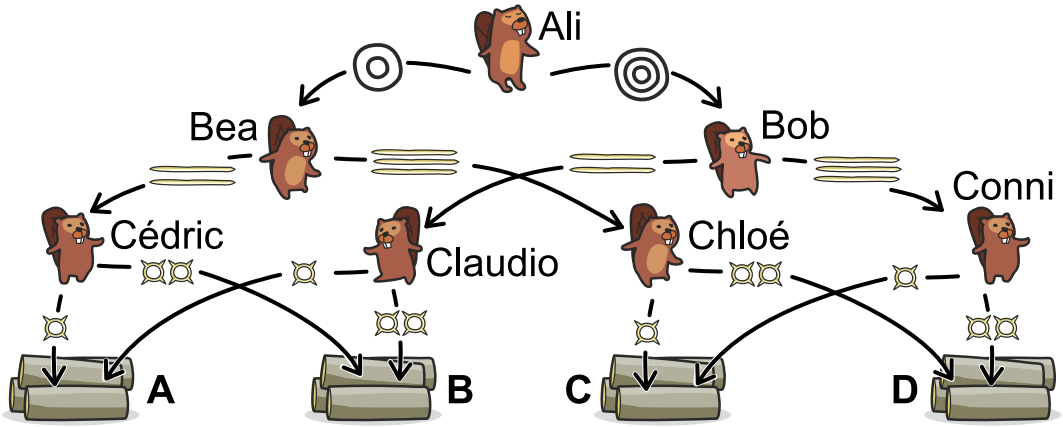
In der Aufgabe oben geht es darum, in einem Programm an den richtigen Stellen fehlende Befehle einzufügen, sodass die Zielsetzung erreicht wird.

Stichwörter und Webseiten

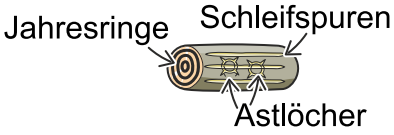
- Programm
- Anweisung: `https://de.wikipedia.org/wiki/Anweisung_(Programmierung)`
- `https://de.wikipedia.org/wiki/Algorithmus`

9. Baumstämme auf Stapel

Im Biberdorf werden die Stämme nach drei Eigenschaften (Anzahl Jahresringe, Anzahl Schleifspuren in der Rinde und Anzahl der Astlöcher) in vier Gruppen (A, B, C, D) verteilt. Wie das abläuft, zeigt das Entscheidungsdiagramm.



Beispielsweise wird dieser Stamm aufgrund folgender Entscheidungen auf den Stapel D gelegt:



- Ali sieht drei Jahresringe und gibt den Stamm an Bob.
- Bob sieht drei Schleifspuren und gibt den Stamm an Conni.
- Conni sieht zwei Astlöcher und legt den Stamm auf den Stapel D.

Auf welchem Stapel wird dieser Stamm abgelegt?

A) Stapel A
B) Stapel B
C) Stapel C
D) Stapel D

Lösung

Die korrekte Antwort ist Stapel C. Dies ist so, weil Ali zwei Jahresringe sieht und den Stamm an Bea gibt. Bea sieht drei Schleifspuren und gibt den Stamm an Chloé weiter. Chloé sieht ein Astloch und legt den Stamm auf den Stapel C.

Wenn man will, kann man für jeden Stapel bestimmen, welche Stämme auf den jeweiligen Stapel gehören. Auf jedem Stapel gibt zwei Arten von Stämmen.

Auf Stapel **A**:

- Stämme mit 2 Jahresringen, 2 Schleifspuren und 1 Astloch.
- Stämme mit 3 Jahresringen, 2 Schleifspuren und 1 Astloch.

Auf Stapel **B**:

- Stämme mit 2 Jahresringen, 2 Schleifspuren und 2 Astlöchern
- Stämme mit 3 Jahresringen, 2 Schleifspuren und 2 Astlöchern

Auf Stapel **C**:

- Stämme mit 2 Jahresringen, 3 Schleifspuren und 1 Astloch
- Stämme mit 3 Jahresringen, 3 Schleifspuren und 1 Astloch

Auf Stapel **D**:

- Stämme mit 2 Jahresringen, 3 Schleifspuren und 2 Astlöchern
- Stämme mit 3 Jahresringen, 3 Schleifspuren und 2 Astlöchern

Dies ist Informatik!

Diese Aufgabe berührt mehrere Konzepte der Informatik.

Vor allem wird das Konzept der *Entscheidungsdiagramme* angesprochen, die sehr vielseitige Anwendungen in der Informatik haben. Hier verwendet man sie zur *Klassifizierung* von Objekten in gewählte Kategorien. (Sehr häufig sind es *Entscheidungsbäume*, eine spezielle Art von Entscheidungsdiagrammen. Das Entscheidungsdiagramm der Aufgabe ist hier kein Entscheidungsbaum, weil auf der untersten Ebene je zwei Gruppen auf denselben Stapel gelegt werden.)

Man kann das Entscheidungsdiagramm hier auch als eine abstrakte Darstellung der Werte einer Funktion von mehreren Variablen ansehen. Terminologisch genau spricht man in der Informatik von «branching programs».

Zudem spricht man hier auch das Konzept der *Attribute* (Merkmale oder Eigenschaften) von Objekten an. Hier haben die Objekte drei Attribute (Jahresringe, Schleifspuren, Astlöcher), wobei jedes Attribut zwei mögliche Werte hat (zwei oder drei Jahresringe oder Schleifspuren und ein oder zwei Astlöcher).

Es gibt viele Anwendungsmöglichkeiten für solche Entscheidungsdiagramme. Eine davon ist die Klassifizierung von Paketen beim Versenden durch ein Netzwerk (mit Routern oder Switches).

Stichwörter und Webseiten

- Entscheidungsbaum: `https://de.wikipedia.org/wiki/Entscheidungsbaum`
- Klassifizierung

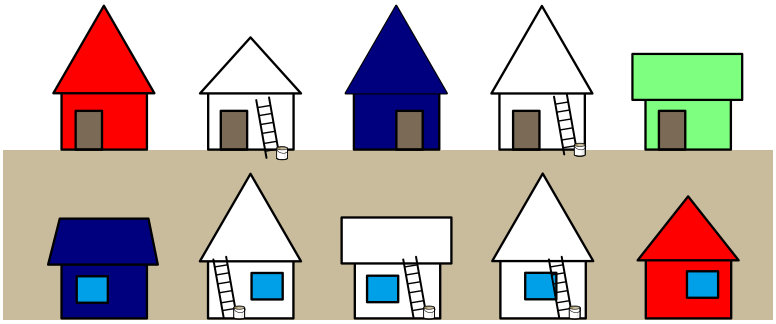
10. Farbiges Quartier

Die Anwohner einer Strasse wollen ihre weissen Häuser farbig anmalen. Jedes Haus soll eine von drei Farben bekommen: Hellgrün, Rot oder Dunkelblau. Damit es nicht langweilig aussieht, gelten folgende Regeln:

- Zwei direkt nebeneinander stehende Häuser dürfen nicht dieselbe Farbe haben.
- Zwei Häuser, die sich auf der Strasse direkt gegenüber stehen, dürfen nicht dieselbe Farbe haben.

Einige Anwohner haben ihre Häuser bereits farbig angemalt. Die restlichen Anwohner müssen jetzt ihre Häuser so anmalen, dass die Regeln nicht verletzt werden.

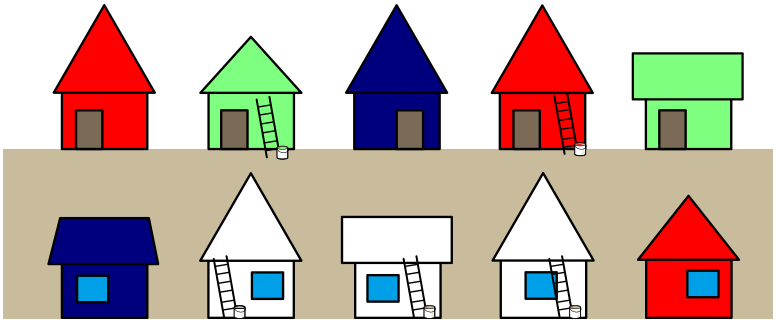
Finde für die Anwohner passende Farben.



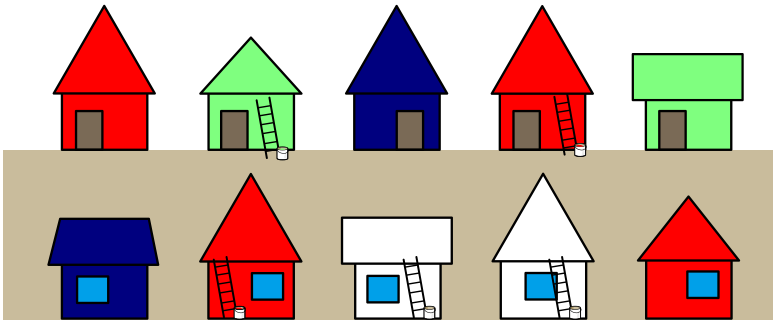
Lösung

Die Farben der Häuser lassen sich am einfachsten schrittweise herausfinden.

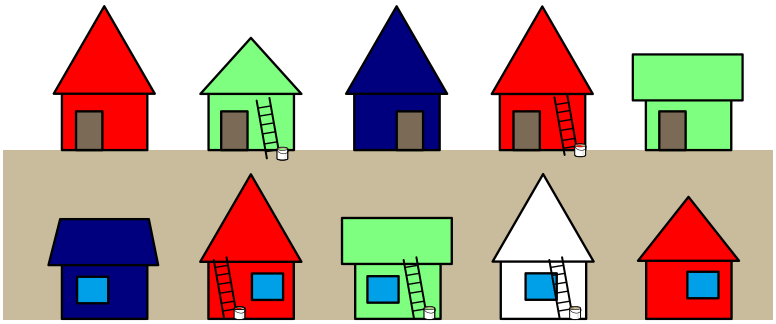
Die beiden weissen Häuser auf der oberen Strassenseite sind jeweils von zwei verschiedenfarbigen Häusern links und rechts umgeben. Deshalb können sie jeweils nur in einer ganz bestimmten Farbe angemalt werden, ohne die Regeln zu verletzen: das weisse Haus links oben in Hellgrün und das rechte weisse Haus oben in Rot.



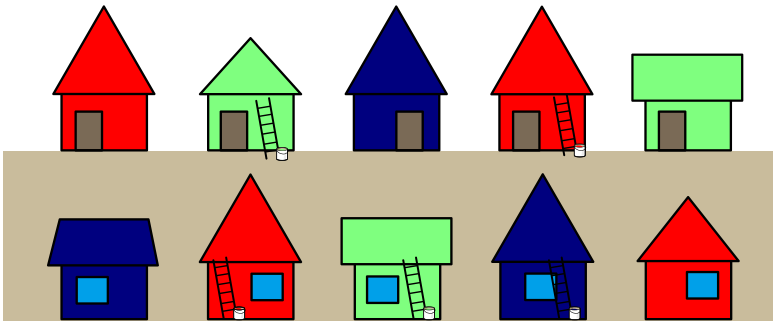
Als Nächstes kann man feststellen, dass das weisse Haus links unten rot angemalt werden muss, weil das Haus direkt links daneben dunkelblau und das Haus direkt gegenüber hellgrün ist:



Fast dieselbe Überlegung kann man jetzt für das mittlere Haus der unteren Strassenseite machen: Es muss hellgrün angemalt werden, weil direkt links davon das gerade eben rot angemalte Haus steht und gegenüber ein dunkelblaues Haus.



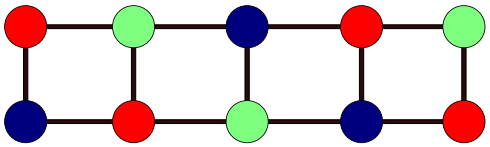
Zuletzt kann man auch für das rechte weisse Haus in der unteren Strassenseite die Farbe bestimmen: Das Haus direkt rechts daneben und das Haus direkt gegenüber sind zwar beide rot, da aber das Haus direkt links daneben jetzt hellgrün ist, bleibt nur noch die Möglichkeit, das Haus dunkelblau anzumalen:



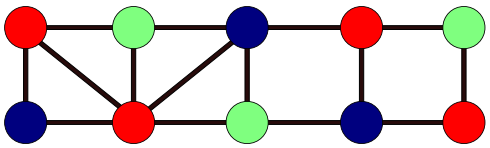
Dies ist Informatik!

Abstrakt gesehen, geht es in dieser Aufgabe darum, eine Lösung zu finden, die die vorgegebenen Einschränkungen (Regeln) erfüllt. Dies ist in der Informatik eine sehr häufige Problemstellung.

Die Häuser und deren direkte Nachbarschaften (sowohl nach links und nach rechts als auch quer über die Strasse hinweg) können gut mit Hilfe eines *Graphen* modelliert werden, einer in der Informatik weit verbreiteten Datenstruktur. Dabei ist jedes Haus ein *Knoten* und jede direkte Nachbarschaft eine *Kante*:



Im Bild sind die Knoten bereits so gefärbt wie die entsprechenden Häuser. Für die Häuser gab es die Regel, dass benachbarte Häuser unterschiedliche Farben haben müssen. Im Bild ist die Färbung der Knoten deshalb so, dass direkt über eine Kante verbundene Knoten nie dieselbe Farbe haben. Dass es eine solche *gültige Färbung* des Graphen mit drei Farben gibt, ist nicht selbstverständlich. Wenn man zwei Kanten so ergänzt, wie im nächsten Bild, dann gibt es keine gültige Färbung mehr: Egal wie man in diesem Graphen die drei Farben verteilt, es gibt immer zwei direkt verbundene Knoten mit derselben Farbe.



Mit vier Farben geht es aber wieder. Vielleicht geht es immer mit vier Farben? Die Antwort ist wieder nein. Doch zumindest eine bestimmte Art von Graphen kann man immer mit vier Farben gültig einfärben: nämlich sogenannte *planare Graphen*. Das sind Graphen, die man so zeichnen kann, dass sich keine Kanten überkreuzen. (Der Graph im letzten Bild ist nicht planar, nämlich wegen

den Verbindungen der vier Knoten ganz links.) Dass planare Graphen eine gültige Färbung mit vier Farben haben, nennt man den *Vier-Farben-Satz*.

Besonders interessant ist der Vier-Farben-Satz für das Erstellen von Landkarten. Wenn man sich jedes Land als Knoten vorstellt und dann benachbarte Länder mit einer Kante verbindet, erhält man immer einen planaren Graphen. (Genau genommen müssen wir dafür die Existenz von sogenannten Enklaven und Exklaven ausschliessen, also Teile von einem Land, die komplett in einem anderen Land liegen.) Diesen Graphen kann man also mit vier Farben gültig einfärben, und deshalb kann man auch die entsprechenden Länder auf der Karte mit vier Farben so einfärben, dass benachbarte Länder immer verschiedene Farben haben.

Der Beweis, dass vier Farben genügen, ist nicht einfach. Dass fünf Farben genügen, wusste man bereits vor 200 Jahren. Dass vier Farben genügen, haben die Mathematiker Kenneth Appel und Wolfgang Haken dann im Jahr 1976 bewiesen. Dabei haben einen Computer verwendet, um eine Vielzahl von Ausnahmen und Gegenbeispielen zu überprüfen. Der Computer hat über tausend Stunden dafür gebraucht. Alles von Hand zu prüfen, wäre völlig unmöglich gewesen. Viele Mathematiker stellten dann die Frage, ob ein solcher Beweis überhaupt gültig ist, weil man dem Computer vertrauen muss.

Stichwörter und Webseiten

- Dreifarbenproblem: `https://de.wikipedia.org/wiki/Dreifarbenproblem`
- Vier-Farben-Satz: `https://de.wikipedia.org/wiki/Vier-Farben-Satz`
- `https://de.wikipedia.org/wiki/Enklave`

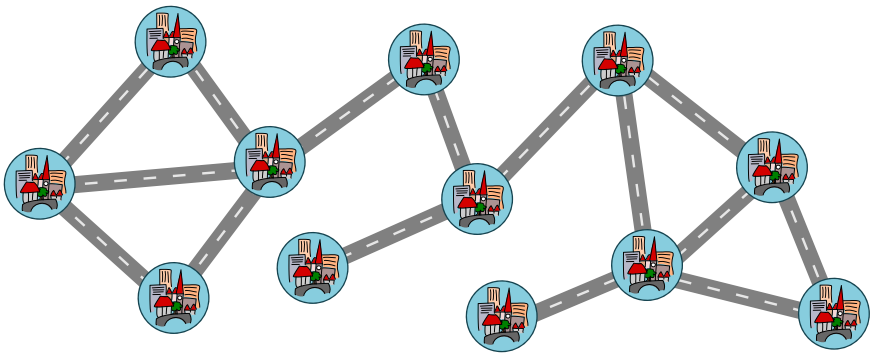
11. Epidemische Überlegungen

Biberland besteht aus 12 Städten, die durch Strassen miteinander verbunden sind. Städte, die direkt oder indirekt durch Strassen miteinander verbunden sind, bilden eine Handelsgemeinschaft. Die Karte zeigt also in der aktuellen Form eine einzelne Handelsgemeinschaft aus 12 Städten.

Um eine Epidemie einzudämmen, soll der Verkehr reduziert werden. Das Biberparlament beschliesst, genau zwei Strassen zu sperren, um die Städte in drei einzelne Handelsgemeinschaften aufzuteilen.

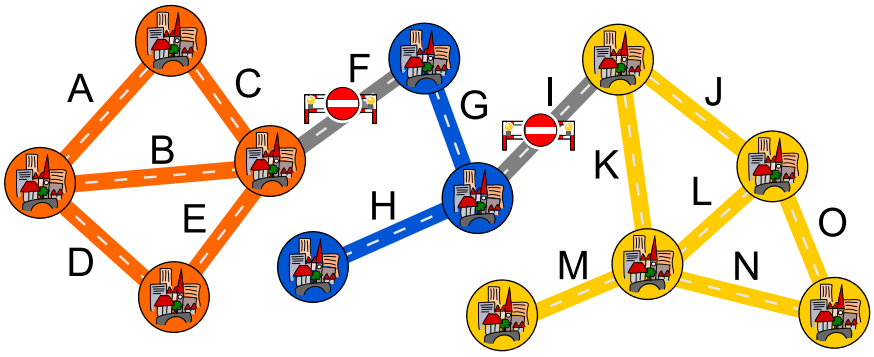
Um niemanden mehr als notwendig zu isolieren, soll die kleinste Handelsgemeinschaft aus möglichst vielen Städten bestehen.

Welche zwei Strassen sollen gesperrt werden?



Lösung

Die richtige Antwort ist: Die Strassen F und I im Bild unten werden gesperrt. So entsteht je eine Handelsgemeinschaft aus 3, 4 und 5 Städten.



Es liegt auf der Hand, dass wir nur Strassen betrachten müssen, die bei einer Sperrung auch die Teilung der Handelsgemeinschaft bewirken, weil sie die einzige Verbindung darstellen. Denn wir brauchen ja zwei echte Teilungen, um zu drei Einheiten zu gelangen. So bringt es zum Beispiel nichts, Strasse B zu sperren, weil man über A und C immer noch alle Städte erreichen kann. Es bleiben daher für die Sperrung nur die Kandidaten F, G, H, I und M übrig.

Probiert man alle 10 Möglichkeiten durch, zwei der fünf Strassen zu sperren, kommt man auf obige Antwort. Als Mensch sieht man zudem sofort, dass die Sperrung von H oder M nur eine einzelne Stadt abtrennen würde und daher nicht in Frage kommt. Das schränkt die Zahl der zu betrachtenden Möglichkeiten weiter ein.

Dies ist Informatik!

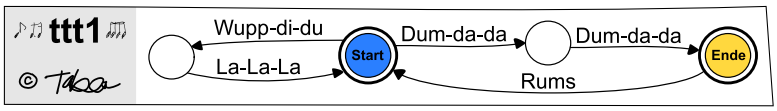
In der Informatik will man ein gegebenes Netzwerk häufig in sogenannte *Zusammenhangskomponenten* aufteilen. In einer Zusammenhangskomponente sind alle Teile mit über direkte oder indirekte Wege miteinander verbunden, während es zwischen verschiedenen Zusammenhangskomponenten überhaupt keine Verbindung gibt. Auf der Hand liegt die Anwendung bei Computernetzen, in denen relevant ist, welche Computer von welchen anderen erreicht werden können. Aber auch zum Beispiel bei der Schrifterkennung (OCR) ist es eine wichtige Information, welche Punkte «verbunden» sind.

Stichwörter und Webseiten

- Zusammenhangkomponenten:
  `https://de.wikipedia.org/wiki/Zusammenhang_(Graphentheorie)`
- Traversierung von Graphen

12. Tabeas taktvolle Texte

Tabea ist sehr erfolgreich mit ihren Liedtexten der Marke ttt: Tabeas Taktvolle Texte. Diese können mit dem folgenden Diagramm ttt1 produziert werden:



Um ein Lied zu produzieren, beginnt Tabea bei «Start» (Start) und folgt einem der ausgehenden Pfeile. Bei mehreren Möglichkeiten darf sie einen aussuchen. Sie singt die entsprechenden Silben auf dem Weg in der gegebenen Reihenfolge. Erreicht sie «Ende» (Ende), darf das Lied zu Ende sein, kann aber auch weitergehen.

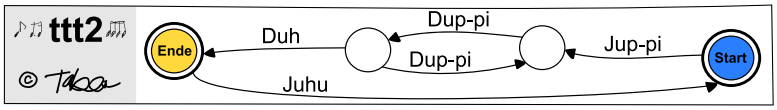
Mögliche Lieder sind zum Beispiel:

«Wupp-di-du La-La-La Wupp-di-du La-La-La
Dum-da-da Dum-da-da Rums Dum-da-da Dum-da-da»

oder

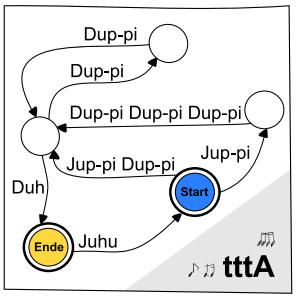
«Dum-da-da Dum-da-da Rums Wupp-di-du La-La-La
Dum-da-da Dum-da-da Rums Wupp-di-du La-La-La
Dum-da-da Dum-da-da Rums Dum-da-da Dum-da-da»

Tabea geht im November 2020 mit neuen Texten nach ttt2 in Produktion:

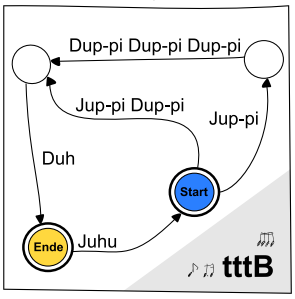


Mit welchem der folgenden Diagramme können genau dieselben Liedtexte wie mit ttt2 produziert werden?

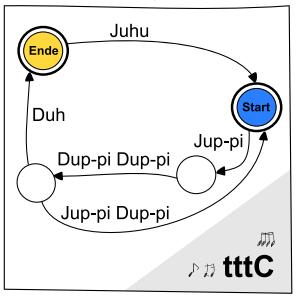
A)



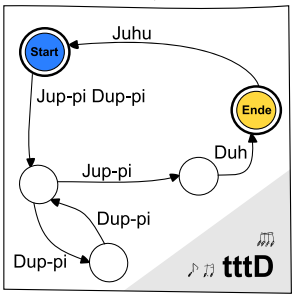
B)



C)

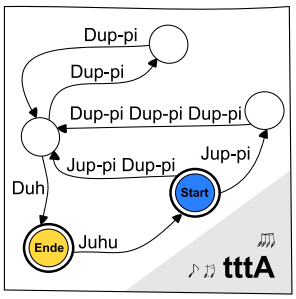


D)

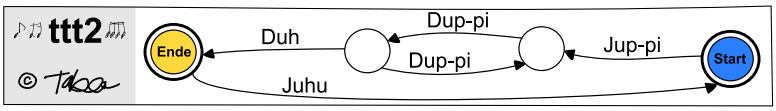


Lösung

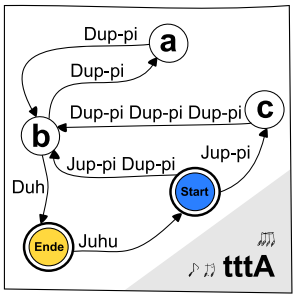
Die korrekte Antwort ist A) Diagramm tttA:



Produziert man ein Lied mit Diagramm ttt2, startet es in jedem Fall mit «Jup-pi» und es folgt mindestens ein «Dup-pi». Jetzt geht es entweder direkt mit «Duh» weiter oder einer geraden Anzahl weiterer «Dup-pi» und danach «Duh». Nun kann das Lied zu Ende sein oder mit einem «Juhu» fortfahren und wieder von vorne anfangen.



Das Diagramm tttA erreicht genau das Gleiche: Vom «Start» aus kann das Lied entweder direkt zu **b** und so mit «Jup-pi Dup-pi» beginnen oder über **c** mit «Jup-pi Dup-pi Dup-pi Dup-pi». Danach folgt mit einem Umweg über **a** alternativ noch eine beliebige gerade Zahl an «Dup-pi», dann kommt man mit «Duh» zum Ende des Liedes. Genau wie in ttt2 kann man nach «Juhu» wieder von Neuem beginnen.



Sowohl ttt2 als auch tttA können nach dem anfänglichen «Jup-pi» eine beliebige ungerade Anzahl von ununterbrochen aufeinanderfolgenden «Dup-pi» erzeugen. Im Gegensatz dazu kann tttB nur 1 oder 3 ununterbrochen aufeinanderfolgende «Dup-pi» erzeugen und tttC nur 1 oder 2. Und tttD kann zwar eine ungerade Anzahl von ununterbrochen aufeinanderfolgenden «Dup-pi» erzeugen, stellt aber dem abschliessenden «Duh» immer ein zusätzliches «Jup-pi» voran, das ttt2 dort nicht erzeugen kann.

Daher ist tttA die einzige mögliche Antwort.

Dies ist Informatik!

Eine wichtige Aufgabe der Informatik ist es, Strukturen in Daten zu erkennen, zum Beispiel in Sprache wie etwa einem Liedtext. Die Diagramme repräsentieren sogenannte *Endliche Automaten*, mit denen sehr strikte Regeln für das Erzeugen und Erkennen bestimmter Sprachen definiert werden können. Das ist wiederum entscheidend bei der Entwicklung von Programmiersprachen. In unserem Beispiel beschreibt der Endliche Automat die Menge von Liedern, die mit diesem erzeugt werden können.

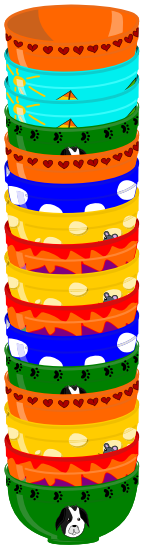
Mustererkennung ist aber auch in vielen anderen Bereichen wichtig, etwa der Verarbeitung natürlicher Sprache.

Stichwörter und Webseiten

- Endliche Automaten:
  `https://de.wikipedia.org/wiki/Deterministischer_endlicher_Automat`
- Formale Sprachen: `https://de.wikipedia.org/wiki/Formale_Sprache`
- `https://sites.google.com/isabc.ca/computationalthinking/pattern-recognition`

13. Schälchen-Stapel

Drei Geschwister wollen beim Morgenessen aus drei gleichen Schälchen essen. Sie haben einen hohen Stapel von Schälchen. Vorsichtshalber nehmen sie immer nur einzelne Schälchen vom oberen Ende des Stapels.

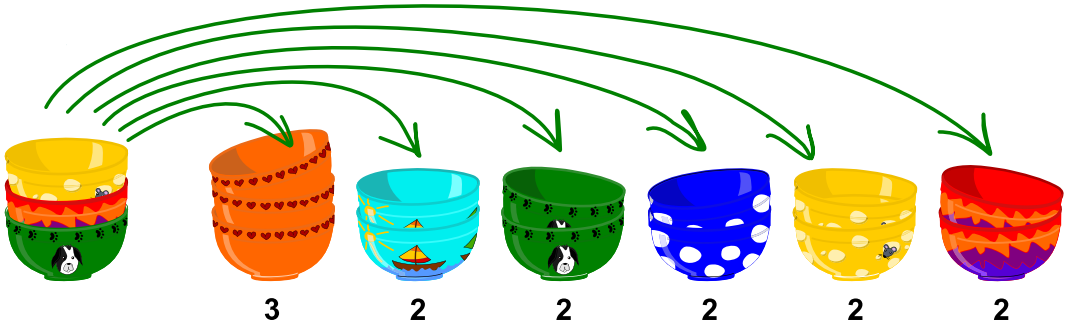


Was ist die kleinste Anzahl von Schälchen, die sie vom abgebildeten Stapel nehmen müssen, um drei gleiche zu haben?

A) 3 Schälchen
B) 4 Schälchen
C) 5 Schälchen
D) 6 Schälchen
E) 7 Schälchen
F) 8 Schälchen
G) 9 Schälchen
H) 10 Schälchen
I) 11 Schälchen
J) 12 Schälchen
K) 13 Schälchen
L) 14 Schälchen
M) 15 Schälchen
N) 16 Schälchen

Lösung

Antwort K): Mindestens 13 Schälchen müssen vom Stapel genommen werden, um drei gleichartige Schälchen zu erhalten.



Dies ist Informatik!

Ein *Stapel*, in der Informatik oft *Stack* und manchmal auch *LIFO-Speicher*, *Keller* oder *Kellerspeicher* genannt, ist eine sehr verbreitete Art Dinge zu speichern. Ein Stapel ist eine sehr einfache, aber trotzdem mächtige Struktur, die man beim Programmieren häufig verwendet. Es gibt Regeln, wie man Dinge auf einen Stapel legen kann und wie man sie wieder entnimmt, meistens nämlich nur von oben. In dieser Aufgabe haben wir es nur mit dem Wegnehmen vom Stapel zu tun. Die Regel besagt, dass immer nur das oberste Objekt vom Stapel genommen werden kann. Will man das zehnte Schälchen im Stapel bekommen, muss man also zehn Schälchen einzeln herunternehmen. Dabei ist es wichtig, einen Platz zu haben, wohin man die anderen neun Schälchen stellen kann; das ist beim Programmieren auch so. Haben wir einen zweiten Stapel und können wir die Stapel so hoch machen, wie wir wollen, dann könnten wir damit theoretisch schon alles berechnen, was man mit einem Computer berechnen könnte! (In der Informatik nennt man diese Eigenschaft auch *Turing-Mächtigkeit*.) So einfache Stapel sind wirklich mächtig!

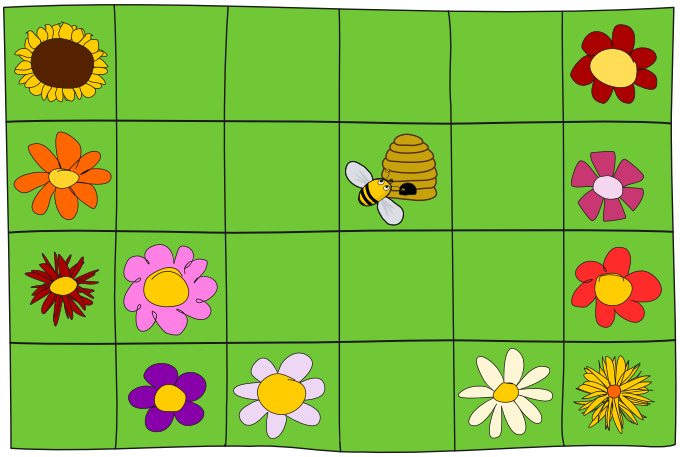
Stichwörter und Webseiten

- Stapelspeicher: `https://de.wikipedia.org/wiki/Stapelspeicher`
- Turingmaschine: `https://de.wikipedia.org/wiki/Turingmaschine`

14. Summ, summ, summ...

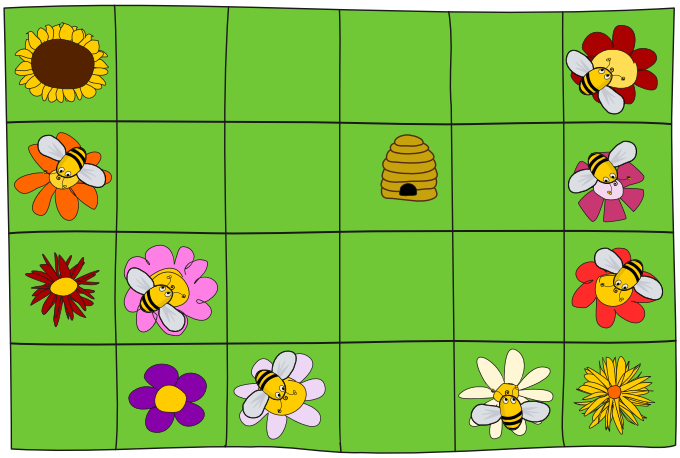
Eine Biene fliegt in 10 Minuten ein Feld nach oben, unten, links oder rechts. Sie fliegt vom Bienenstock aus höchstens 30 Minuten, bevor sie umkehrt.

Welche Blumen sind vom Bienenstock aus in höchstens 30 Minuten erreichbar?



Lösung

Die Blumen mit Biene darauf sind vom Bienenstock aus in höchstens 30 Minuten erreichbar:



Das Bild unten zeigt für jedes Feld, wie viele Minuten eine Biene braucht, um es vom Bienenstock aus zu erreichen. In einer halben Stunde sind also alle Felder erreichbar, in denen 10, 20 oder 30 steht.

Das Ausfüllen mit den Zahlen geht so: In die Felder neben dem Bienenstock schreiben wir 10 drin, weil die Biene 10 Minuten braucht, um vom Bienenstock dorthin zu fliegen. Dann schreiben wir 20 in alle leeren Felder neben einem Feld mit 10, weil die Biene in 10 Minuten von einem Feld zum nächsten kommt. Wir machen jetzt immer so weiter. Wir schreiben also 30 in alle leeren Felder neben einem Feld mit 20 drin. Dann schreiben wir 40 in alle leeren Felder neben einem Feld mit 30 drin. Schliesslich schreiben wir 50 in alle leeren Felder neben einem Feld mit 40 drin.

Dies ist Informatik!

Beim Lösen der Aufgabe haben wir für jedes Feld berechnet, in welcher Zeit eine Biene es vom Bienenstock aus erreichen kann. Zuerst werden die in 10 Minuten erreichbaren Felder bestimmt. Diese werden dann benützt, um die 20 Minuten entfernten Felder zu bestimmen. Mit Hilfe der 20 Minuten entfernten Felder werden dann die 30 Minuten entfernten Felder gefunden und so weiter.

Wir verwenden also bereits berechnete und gespeicherte Ergebnisse (die Zahlen der ausgefüllten Felder) zum Berechnen von weiteren Ergebnissen (die Zahlen der benachbarten, noch leeren Felder). Dieses sehr allgemeine Prinzip nennt man *dynamisches Programmieren*. Dabei ist es meist wichtig, in welcher Reihenfolge die Ergebnisse berechnet werden. Dies ist auch beim Bienenflug zu beachten.

In der Aufgabe fliegt eine Biene in 10 Minuten nur nach oben, unten, links oder rechts. Das ist etwas ungewöhnlich, denn in Realität fliegt eine Biene wahrscheinlich auch gerne schräg über die Felder. Mit dieser realistischeren Annahme wären die in einer halben Stunde erreichbaren Felder durch einen Kreis begrenzt anstatt durch eine Raute wie in der Aufgabe.

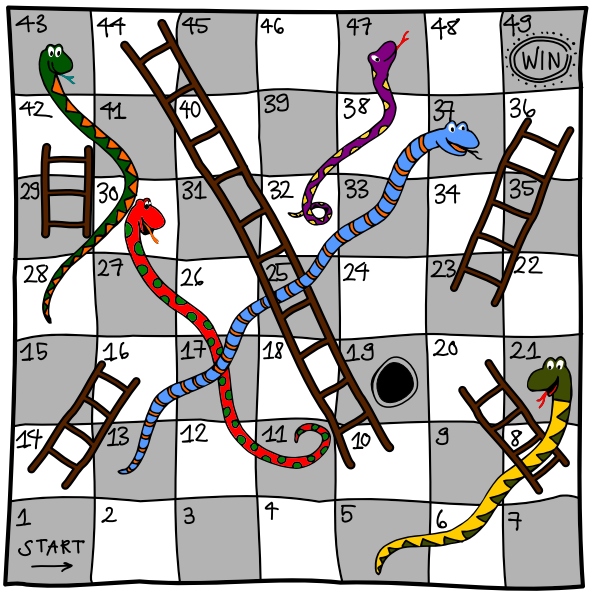
Die übliche Distanzmessung, die zu einem Kreis führt, heisst *Euklidische Distanz*. Die Distanzmessung aus der Aufgabe, bei der man sich nur so horizontal oder vertikal durch Quadrate bewegen darf, heisst *Manhattan-Metrik*. (Der Name kommt von den gitterförmigen Strassennetzen in modernen Städten wie Manhattan.)

Stichwörter und Webseiten

- Dynamisches Programmieren:
  `https://de.wikipedia.org/wiki/Dynamische_Programmierung`
- Euklidische Distanz: `https://de.wikipedia.org/wiki/Euklidischer_Abstand`
- Manhattan-Distanz: `https://de.wikipedia.org/wiki/Manhattan-Metrik`

15. Leiterspiel

Beim Leiterspiel starten alle Spieler auf Feld 1. Wer zuerst Feld 49 erreicht, gewinnt. In jeder Runde würfelt man und geht mit seiner Figur die entsprechende Zahl (zwischen 1 und 6) 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, so darf man sie noch in der gleichen Runde ganz hinaufklettern.

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

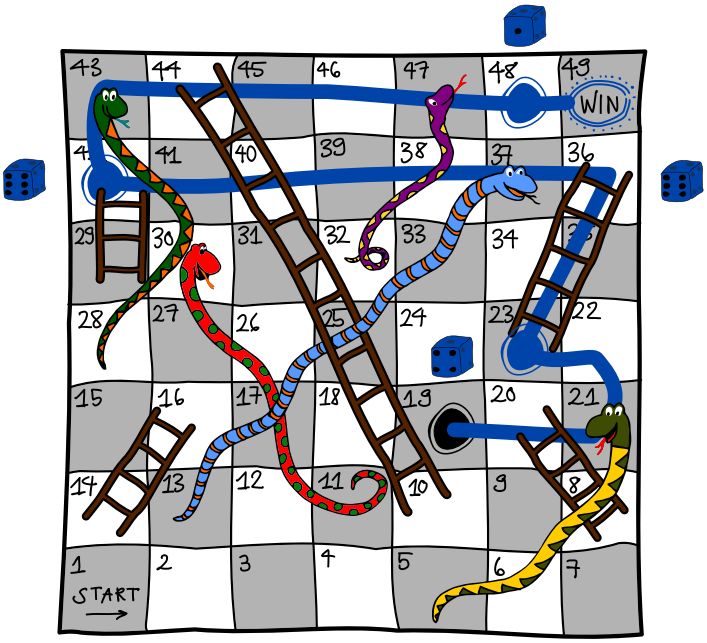
Deine Figur steht auf dem Feld 19. Wie viele Runden brauchst Du mindestens noch, um das Feld 49 zu erreichen?

A) 2 Runden
B) 3 Runden
C) 4 Runden
D) 5 Runden

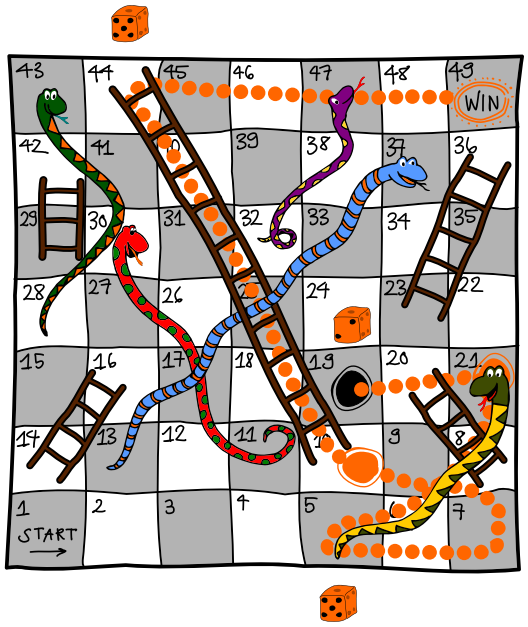
Lösung

Die richtige Antwort ist B) 3 Runden.

Wenn Du gierig bist und 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 und man braucht weitere 3 Würfe, zum Beispiel 6 – 6 – 1, um zum Ziel zu kommen.



Wenn Du allerdings 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 ganz hinauf zu 44 und dann ins Ziel.



In 2 Runden ist das Ziel nicht zu erreichen. Nur einen Wurf vom Ziel entfernt sind die Felder 48, 46, 45, 44 und keines dieser Felder ist von 19 aus in einer Runde zu erreichen.

Dies ist Informatik!

Viele Aufgaben lassen sich lösen, indem man den kürzesten Weg zwischen zwei Punkten sucht. Hierbei hat «kurz» oft nicht die intuitive Bedeutung. Hier haben wir zum Beispiel den Weg mit den wenigsten Runden gesucht und nicht den Weg, der am wenigsten Felder durchschreitet. Das kennt man auch vom Navigationssystem, das anbietet, nach der kürzesten Wegstrecke oder nach der kürzesten Zeit zu optimieren. Bei Logistikunternehmen berechnen die gleichen Geräte die Strecke mit den kleinsten Maut-Gebühren.

In der Informatik können oft die gleichen Verfahren (Algorithmen) für ganz unterschiedliche Aufgaben verwendet werden, wenn man diese entsprechend modelliert.

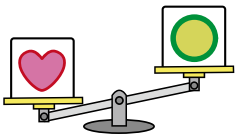
Stichwörter und Webseiten

- Kürzeste Wege: `https://de.wikipedia.org/wiki/Dijkstra-Algorithmus`
- Leiterspiel: `https://de.wikipedia.org/wiki/Leiterspiel`

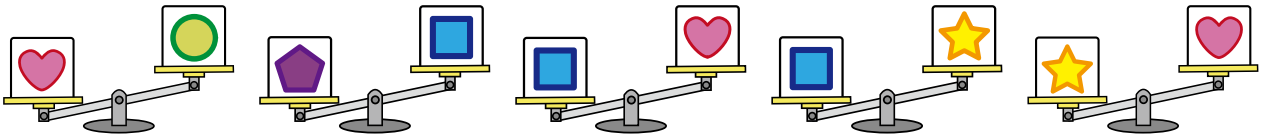
16. Schwere Vergleiche

Fünf Kisten sind mit fünf unterschiedlichen Symbolen gekennzeichnet:  ,  ,  ,  und  .

Mit Hilfe einer Waage werden jeweils zwei Kisten verglichen. Der folgende Vergleich ergibt beispielsweise, dass  schwerer als  ist:



Es werden insgesamt fünf Vergleiche gemacht:



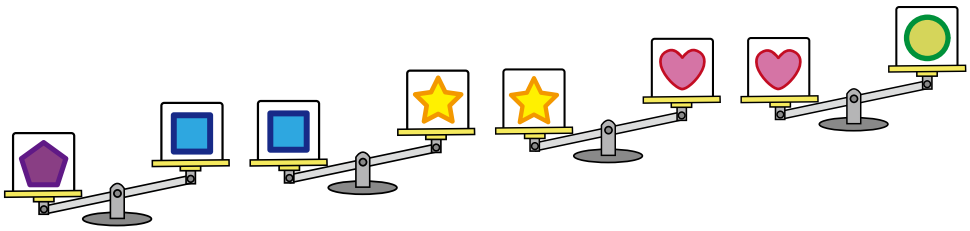
Welche Kiste ist am schwersten?

A) 
B) 
C) 
D) 
E) 

Lösung

Die Kiste C) mit dem Pentagon ist am schwersten.

Die folgende Abbildung enthält vier der fünf gemachten Vergleiche und alle fünf Kisten.



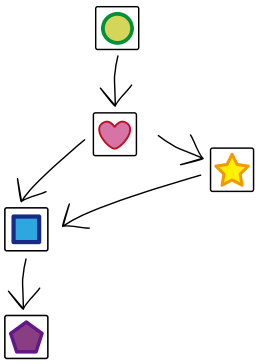
Damit sieht man sofort: Die Kiste mit dem Pentagon ist schwerer als die Kiste mit dem Quadrat . Die Kiste mit dem Quadrat ist schwerer als die Kiste mit dem Stern . Die Kiste mit dem Stern ist schwerer als die Kiste mit dem Herz . Und die Kiste mit dem Herz ist schwerer als die Kiste mit dem Kreis .

Daraus kann man jetzt schliessen, dass die Kiste mit dem Pentagon schwerer ist als alle anderen. Dies liegt an einer speziellen Eigenschaft des Vergleichens von Gewichten: Wenn A schwerer ist als B und B schwerer als C, dann ist auch A schwerer als C. Diese sehr einleuchtende Eigenschaft nennt man *Transitivität*.

Übrigens gibt es einen cleveren Weg, diese Aufgabe abkürzen. Da nach der einen schwersten Kiste gesucht wird, reicht es, einfach nach der Kiste suchen, die keinmal leichter als eine andere Kiste ist, und das ist nur die Kiste mit dem Pentagon .

Dies ist Informatik!

Letztlich geht es in dieser Aufgabe darum, irgendwelche Objekte zu sortieren. Zum Sortieren benützt man in der Informatik häufig spezielle *Graphen*, die aus *Knoten* (den zu sortierenden Objekten) und *Kanten* (Vergleichen zwischen zwei Objekten) bestehen. Die Objekte sind in dieser Aufgabe die Kisten und die Vergleiche sind die Wägungen. Wenn man die Kanten als Pfeile zeichnet, die auf das schwerere Objekt zeigen, dann sieht der Graph für diese Aufgabe so aus:



Die Objekte sollen jetzt so in einer Reihe angeordnet werden, dass die Pfeile immer nur von den Objekten weiter links zu Objekten weiter rechts gehen. Eine solche Anordnung nennt man dann

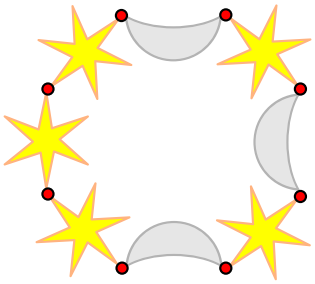
eine *topologische Sortierung*. Eine topologische Sortierung erhält man sehr einfach, indem man immer wieder einen Knoten aus dem Graphen herausnimmt, auf den kein Pfeil zeigt, und die herausgenommenen Knoten in dieser Reihenfolge hintereinander stellt.

Aber Achtung: Nicht zu jedem Graphen gibt es eine topologische Sortierung. Zum Beispiel existiert keine, wenn es irgendwo drei Kanten gibt, die im Kreis zeigen.

Stichwörter und Webseiten

- Transitivität: `https://de.wikipedia.org/wiki/Transitive_Relation`
- Graph: `https://de.wikipedia.org/wiki/Graph_(Graphentheorie)`
- Topologische Sortierung: `https://de.wikipedia.org/wiki/Topologische_Sortierung`

17. Armband



Marie möchte das Armband rechts.

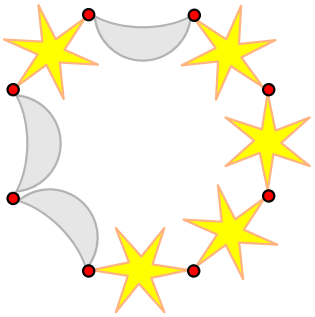
Daher gibt sie Jonas folgende Anweisungen:

- Nimm einen Stern () und einen Mond () und verbinde die beiden irgendwie zu einem Paar. Mach dies insgesamt dreimal, sodass du also drei Paare hast.
- Nimm diese drei Paare und verbinde sie zu einer langen Kette.
- Füge an einem Ende der Kette zwei weitere Sterne hinzu. Verbinde jetzt die beide Enden der Kette, um ein Armband zu erhalten.

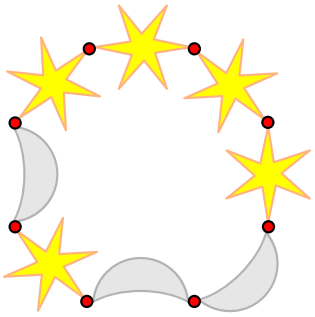
Jonas hat kein Bild des gewünschten Armbands. Es kann sein, dass ein ganz anders aussehendes Armband herauskommt, obwohl sich Jonas exakt an Maries Anweisungen hält.

*Eines der vier Armbänder kann **nicht** herauskommen, wenn sich Jonas genau an Maries Anweisungen hält. Welches?*

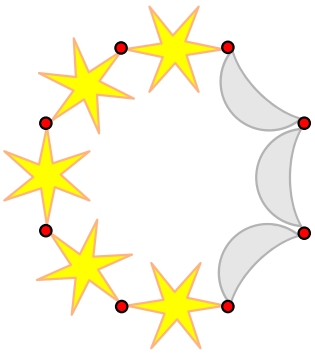
A)



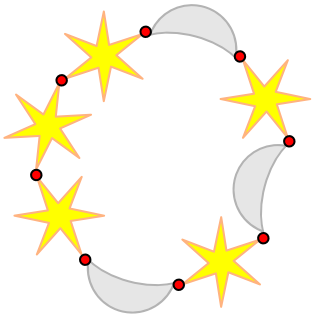
B)



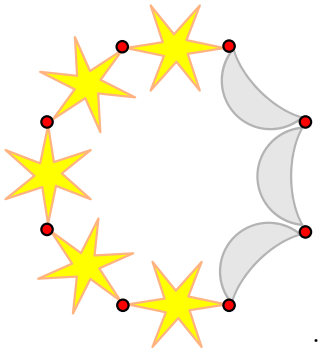
C)



D)

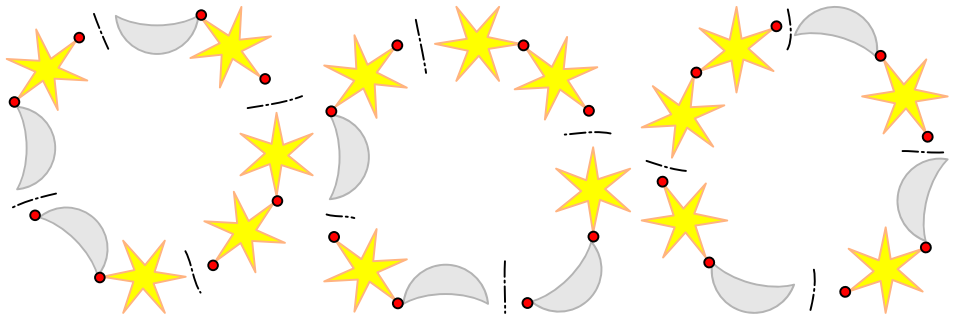


Lösung

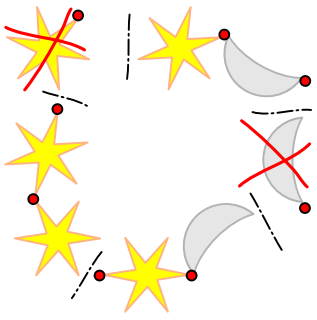
Die richtige Antwort ist C)  .

Nur dieses Armband kann nach Maries Anweisungen nicht herauskommen.

Die Armbänder der anderen drei Antworten hingegen sind nach Maries Anweisungen korrekt. Dies sieht man zum Beispiel, weil jedes von diesen Armbändern in drei Stern-Mond-Paare und ein Stern-Stern-Paar aufgeteilt werden kann, so wie im Bild gezeigt.



Ein Mond kann nur als Teil von einem Mond-Stern-Paar in das Armband kommen. Deshalb hat jeder Mond mindestens einen Stern neben sich. Drei Monde hintereinander wie in Armband C können also nicht entstehen. Auch fünf oder mehr Sterne hintereinander sind unmöglich.



Dies ist Informatik!

Wenn Programmierer einem Computer Anweisungen geben, dann ist es wichtig, dass sie exakt spezifizieren, was der Computer zu tun hat. Andernfalls könnte man ein unerwünschtes Ergebnis erhalten. Zum Beispiel vergass Marie in ihrer Anweisungsliste genau zu sagen, wie die drei Stern-Mond-Paare aneinandergefügt werden müssen. Im von ihr gewünschten Armband ist ein Mond stets von Sternen umgeben. Es fehlte also etwas, obwohl die Anweisungen sehr genau aussehen.

Gäbe es einen Computer, der eine Maschine für die Herstellung von Armbändern steuerte, wären Maries Anweisungen nicht genau genug. Glücklicherweise würden reale Computer gewöhnlich einfach anhalten, mit der Meldung: «Ich weiss nicht, was du meinst, weil die Anweisungen nicht ausreichend klar sind.»

In der Informatik gibt es viele Mechanismen, um Dinge sehr exakt zu beschreiben. Ein Mechanismus sind sogenannte *(formale) Grammatiken*. Eine Grammatik enthält *Regeln*, die genau beschreiben, wie man bestimmte *Wörter* (eine Abfolge von Buchstaben) erzeugen kann. Zum Beispiel kann man die Anweisungen von Marie in einer Grammatik so ausdrücken:

$$\begin{aligned} &\text{A} \to \text{KSS} && (1)\\ &\text{K} \to \text{PPP} && (2)\\ &\text{P} \to \text{SM} && (3) \end{aligned}$$

Hier steht A für Armband, K für Kette, P für Paar, S für Stern und M für Mond. Man beginnt mit A und kann dann neue Wörter erzeugen, indem man die drei Ersetzungsregeln beliebig oft anwendet. Dies macht man so lange, bis das Wort nur noch aus den Symbolen S und M bestehen. Zum Beispiel:

$$\begin{aligned} &\text{A} \Rightarrow \text{KSS} && \text{mittels Regel (1)}\\ &\text{KSS} \Rightarrow \text{PPPSS} && \text{mittels Regel (2)}\\ &\text{PPPSS} \Rightarrow \text{SMPPSS} \Rightarrow \text{SMSMPSS} \Rightarrow \text{SMSMSMSS} && \text{mittels Regel (3)} \end{aligned}$$

Man kann sich überlegen, dass die obige Grammatik genau den Anweisungen von Marie entspricht.

In der Informatik geht es nicht nur um das Programmieren. Oft geht es um das Beschreiben von Objekten. Eine Menge von Erzeugungsregeln (die Grammatik bzw. Maries Anweisungen) kann eine Klasse von Objekten (bestimmte Wörter bzw. die möglichen Armbänder) genau beschreiben. In der Klasse sind nämlich genau diejenigen Objekte, die man mit den Regeln erzeugen kann.

Stichwörter und Webseiten

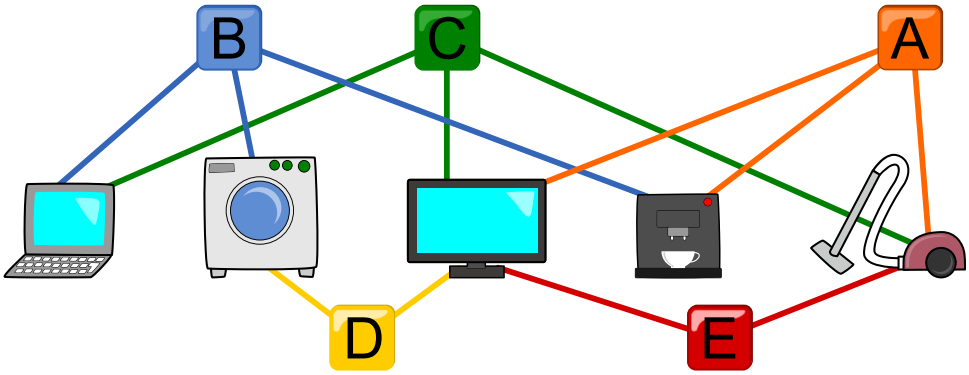
- Formale Grammatik: `https://de.wikipedia.org/wiki/Formale_Grammatik`

18. Haushaltsgeräte

Im Haus von Biber Bruno gibt es fünf elektrische Geräte (Computer, Waschmaschine, Fernseher, Kaffeemaschine und Staubsauger) und fünf Knöpfe (A, B, C, D und E) zum Ein- und Ausschalten. Die Verkabelung ist aber sehr ungewöhnlich. Jeder Knopf ist mit mehreren Geräten verbunden, so wie im Bild unten gezeigt. Jedes Mal, wenn man einen Knopf drückt, schaltet er alle verbundenen Geräte um: Die ausgeschalteten werden eingeschaltet und die eingeschalteten werden ausgeschaltet.

Zu Beginn sind alle Geräte ausgeschaltet. Werden zum Beispiel die Knöpfe A, C und E gedrückt, so ist der Staubsauger eingeschaltet, denn durch den ersten Knopf wird er eingeschaltet, durch den zweiten dann ausgeschaltet und durch den dritten Knopf wieder eingeschaltet.

Welche Knöpfe muss Bruno drücken, damit am Ende nur der Fernseher und die Kaffeemaschine eingeschaltet sind?



Lösung

Wenn man die Schalter B, C, D, E drückt (in beliebiger Reihenfolge), dann werden nur der Fernseher und die Kaffeemaschine eingeschaltet.

Wir können auch systematisch herausfinden, wie man jedes Gerät einzeln ein- und ausschaltet. Wir beginnen mit zwei einfachen Kombinationen:

- A + E (das Drücken von A und E) kontrolliert die Kaffeemaschine alleine.
- C + E (das Drücken von C und E) kontrolliert den Computer alleine.

Als Nächstes beobachten wir, dass die Waschmaschine einzeln kontrolliert werden kann, indem man zuerst B drückt und sofort danach den Computer und die Kaffeemaschine wieder so umschaltet, wie sie zuvor waren, nämlich durch Drücken von A + E sowie C + E. Insgesamt wird die Waschmaschine also durch B + A + E + C + E einzeln kontrolliert. Hier kommt E doppelt vor. Zweimal denselben Schalter zu drücken, ist so, als hätte man ihn gar nicht gedrückt. Deshalb kann man die Waschmaschine auch mit B + A + C einzeln kontrollieren. Mit dieser Methode erhalten wir folgende Liste von Knopf-Kombinationen, um die einzelnen Geräte zu kontrollieren:

- Computer: C + E
- Kaffeemaschine: A + E
- Waschmaschine: A + B + C
- Fernseher: A + B + C + D
- Staubsauger: A + B + C + D + E

Um den Fernseher und die Kaffeemaschine einzuschalten, müssen wir daher A + B + C + D + A + E drücken, was sich zu B + C + D + E vereinfacht, da sich die beiden A gegenseitig aufheben.

Dies ist Informatik!

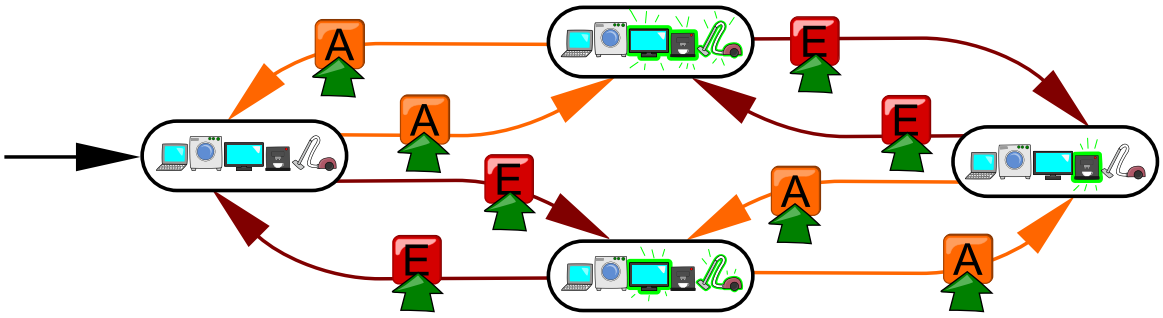
Das System der Geräte und der Knöpfe zum Ein- und Ausschalten kann als sogenannter *endlicher Automat* modelliert werden. Das geht wie folgt.

Das System der fünf Gerät hat viele verschiedene *Zustände*. Ein Zustand ist zum Beispiel, wenn nur der Fernseher eingeschaltet ist. Ein anderer Zustand ist es, wenn alle Geräte ausgeschaltet sind. (Weil am Beginn alle Geräte ausgeschaltet sind, nennen wir das den *Anfangszustand*.) Und ein weiterer Zustand ist es, wenn nur der Fernseher und die Kaffeemaschine eingeschaltet sind. (In unserem Beispiel ist das der *Zielzustand*, weil wir das erreichen wollen.)

Das Drücken eines Knopfes bringt das System von einem Zustand in einen anderen. Zum Beispiel: Wenn das System im Anfangszustand ist, wechselt es beim Drücken von E in den Zustand, wo nur Fernseher und Staubsauger eingeschaltet sind. Einen solchen Wechsel des Zustandes nennt man auch einen *Übergang*.

Wenn man alle Zustände des Systems einzeln hinzeichnet, die Übergänge zwischen ihnen mit Pfeilen einzeichnet und den Anfangszustand mit einem speziellen Pfeil markiert, dann kommt ein Bild wie

das unten heraus. (Aus Platzgründen sind nur vier Zustände und die Übergänge zwischen ihnen gezeichnet.) So etwas nennt man in der Informatik einen endlichen Automaten. (Ein endlicher Automat ist übrigens einfach ein spezieller Graph; die Zustände sind die *Knoten* und die Übergänge sind die *Kanten*.) Das Bild zeigt alle Zustände, die vom Anfangszustand her erreichbar sind, wenn nur die Schalter A und E gedrückt werden können.



In der Aufgabe geht es darum, wie man vom Anfangszustand (alle Geräte sind aus) zum Zielzustand (nur Fernseher und Kaffeemaschine sind ein) kommt. Es ist also ein Weg vom Anfangszustand zum Zielzustand gesucht. Das Finden von Wegen in Graphen ist eine Grundaufgabe der Informatik.

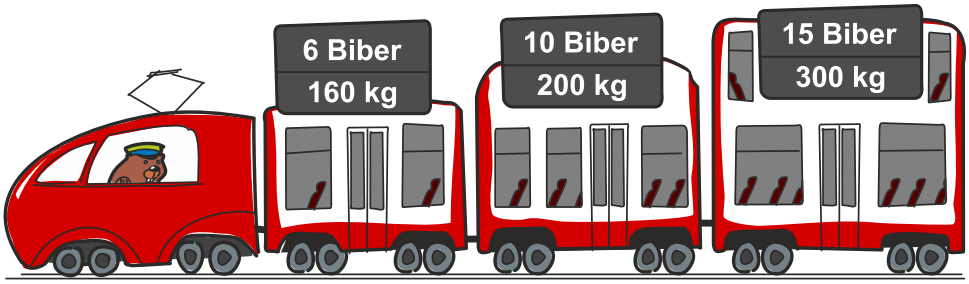
Stichwörter und Webseiten

- Endlicher Automat: `https://de.wikipedia.org/wiki/Endlicher_Automat`

19. Maximalausflug

Acht Biberfamilien möchten mit dem «Gletscher-Express» fahren. Die Familien sind mit der Anzahl ihrer Mitglieder und dem Gewicht ihres Gepäcks in der folgenden Tabelle aufgeführt:

Familienname	**Anzahl Mitglieder**	**Gewicht des Gepäcks in kg**
Ammann	3	50
Bernasconi	4	80
Camenzind	5	110
Donetta	4	80
Emery	2	40
Favre	3	70
Gerber	6	130
Huber	5	100



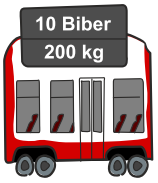
Das Bild zeigt für jeden Waggon, wie viele Biber und wie viel Gepäck in ihm höchstens transportiert werden dürfen. Zudem müssen Familien mit ihrem Gepäck komplett in einem Waggon fahren und dürfen sich nicht aufteilen.

Wie viele Biberfamilien können maximal mit dem «Gletscher-Express» fahren?

A) 1 Biberfamilie
B) 2 Biberfamilien
C) 3 Biberfamilien
D) 4 Biberfamilien
E) 5 Biberfamilien
F) 6 Biberfamilien
G) 7 Biberfamilien
H) 8 Biberfamilien

Lösung

Es können maximal 7 Biberfamilien mitfahren. Eine der möglichen Verteilungen dafür ist:

	Familienname	Anzahl Mitglieder	Gepäck in kg
	Gerber	6	130
	Total:	**6**	**130**
	Ammann	3	50
	Camenzind	5	110
	Emery	2	40
	Total:	**10**	**200**
	Bernasconi	4	80
	Donetta	4	80
	Huber	5	100
	Total:	**13**	**260**

Die 8 Biberfamilien sind zusammen insgesamt 32 Personen, während im Zug nur 31 Sitze verfügbar sind. Es ist also ausgeschlossen, dass alle 8 Biberfamilien mitfahren können.

Dies ist Informatik!

Die Informatik kümmert sich oft um *Optimierungsprobleme*, bei denen begrenzte Ressourcen – wie hier die Platz- und Gewichtskapazität – möglichst gut ausgenutzt werden soll. In der Realität sollte natürlich kein Fahrgast zurückbleiben, aber die Bahngesellschaft kann zum Beispiel durchaus kalkulieren, lieber einzelne Reisende bequem per Taxi zu transportieren als einen kompletten Zug einzusetzen, der dann fast leer fährt.

Aufgabenstellungen dieser Art sind als *Packprobleme* bekannt. Zu dieser Kategorie gehört auch das bekannte *Rucksackproblem*.

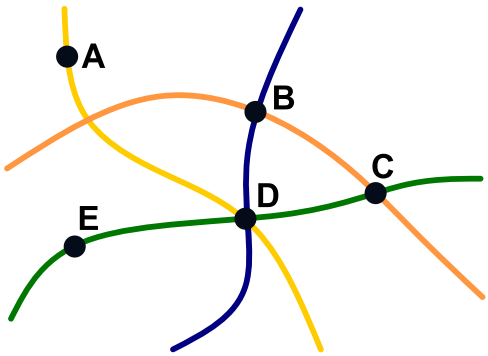
Manchmal lassen sich solche Probleme so reduzieren, dass sie mit Hilfe *Dynamischer Programmierung* gelöst werden können, also indem zunächst mögliche Teillösungen identifiziert werden, die sich dann immer weiter zu einer Gesamtlösung ausbauen lassen. In vielen Fällen gehören die Aufgaben allerdings zu den sogenannten *NP-vollständigen* Problemen, was bedeutet, dass man momentan keine bessere Lösung kennt, als geschickt auszuprobieren. Auf diese Weise werden die allermeisten auch diese Aufgabe gelöst haben.

Stichwörter und Webseiten

- Rucksackproblem: `https://de.wikipedia.org/wiki/Rucksackproblem`
- Dynamische Programmierung:
  `https://de.wikipedia.org/wiki/Dynamische_Programmierung`
- Packprobleme
- NP-vollständig: `https://de.wikipedia.org/wiki/NP-Vollständigkeit`

20. Bahnnetz

Dies ist eine Karte von 5 Städten und 4 Bahngleisen. Die schwarzen Punkte sind die Städte, die farbigen Linien sind Bahngleise.

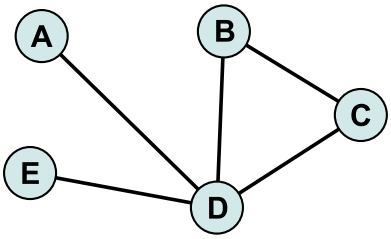


Ein Diagramm soll diese Karte so darstellen, dass:

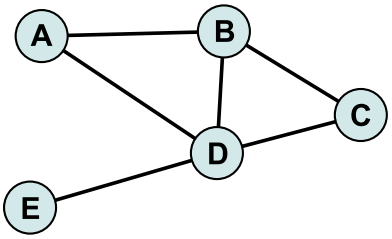
- die Städte durch Kreise dargestellt sind und
- zwei Städte genau dann durch eine Gerade verbunden sind, wenn sie an einem gemeinsamen Bahngleis liegen.

Welches Diagramm stellt die Karte richtig dar?

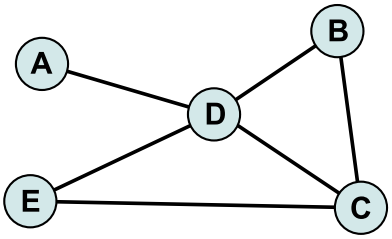
A)



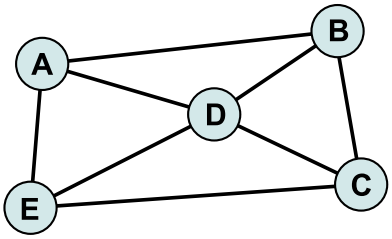
B)



C)

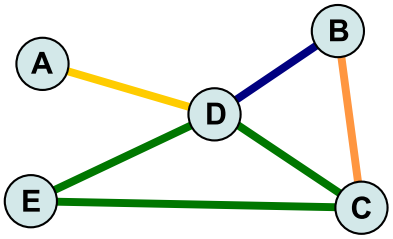


D)



Lösung

Die richtige Antwort ist C).



Genaues Anschauen der Karte zeigt, dass:

- Städte A und D gemeinsam am gelben Bahngleis liegen,
- Städte B und C gemeinsam am orangefarbenen Bahngleis liegen,
- Städte B und D gemeinsam am blauen Bahngleis liegen und
- Städte C, D und E gemeinsam am grünen Bahngleis liegen.

Alle anderen Antworten sind falsch:

- In den Antwort A fehlt die Gerade zwischen Städten C und E, die aufgrund des grünen Bahngleises bestehen muss.
- Antwort B hat dasselbe Problem wie Antwort A und zusätzlich gibt es eine Gerade zwischen den Städten A und B, obwohl die nicht gemeinsam an einem Bahngleis liegen.
- In Antwort D gibt es zwei Geraden von Stadt A zu Stadt B und von Stadt A zu Stadt E, obwohl Stadt A weder mit Stadt B noch mit Stadt E an einem gemeinsamen Bahngleis liegt.

Besondere Beachtung verdienen die beiden folgenden Punkte:

- Obwohl man von Stadt A zu Stadt B gelangen kann, wenn man mehrere Bahngleise benützt, liegen die beiden Städte nicht an einem gemeinsamen Bahngleis.
- Obwohl auf dem Weg von Stadt C nach Stadt E auf dem grünen Bahngleis noch eine dritte Stadt liegt, liegen Städte C und E dennoch an einem gemeinsamen Bahngleis.

Dies ist Informatik!

Es gibt viele verschiedene Möglichkeiten, wie man die Realität abbilden kann. Zum Beispiel ist die obige Karte mit den farbigen Linien schon eine ziemlich abstrakte Darstellung der realen Situation. Eine sehr wichtige Darstellungsart ist ein *Graph* – ein Diagramm, das aus *Knoten* besteht (kleine Kreise) und aus *Kanten* (Geraden zwischen Knoten). Diese Darstellungsart wird in der Lösung verwendet.

Vieles wird einfacher, wenn man eine gute Darstellungsart wählt. Deshalb ist es beim Programmieren wichtig, viele Darstellungsarten zu kennen. Oft kann man gar nicht sagen, dass eine Darstellungsart besser ist als die andere. Je nach Anwendungszweck ist die eine oder die andere besser geeignet. Der Graph in der Lösung ist zum Beispiel praktisch, weil man direkt ablesen kann, dass man mit nur

einem Bahngleis von C nach E kommt. Gegenüber der Karte verliert man aber die Information, dass man auf der Fahrt auf diesem Bahngleis von Stadt C nach Stadt E an der Stadt D vorbeikommt.

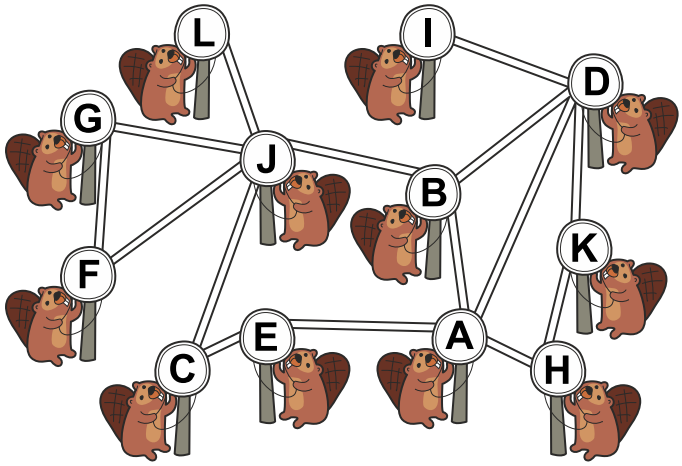
Stichwörter und Webseiten

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

21. Kommunikationsnetzwerk

Biber verbreiten gerne Nachrichten untereinander. Wenn ein Biber eine neue Nachricht erhält, versendet er sie gleichzeitig an alle Nachbarn. (Nachbarn sind mit einer direkten weissen Linie verbundene Biber.) Das Versenden verläuft in Runden: Vom Absenden an die Nachbarn bis zum Erhalt vergeht immer eine Runde und es können beliebig viele Nachrichten gleichzeitig unterwegs sein.

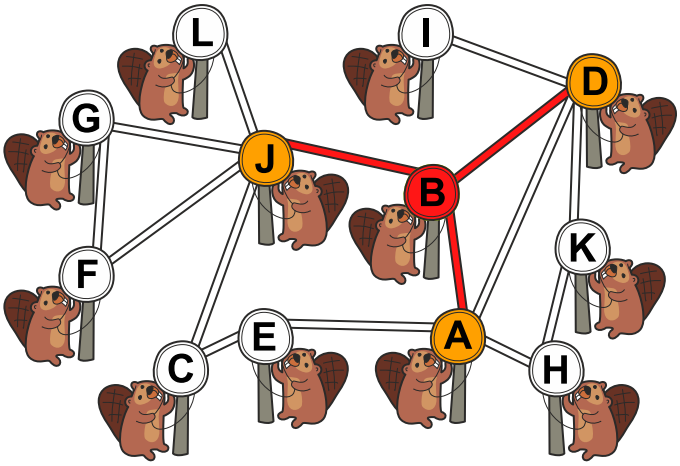
Von welchem Biber aus erreicht eine Nachricht am schnellsten, also in der kleinsten Anzahl Runden, alle anderen Biber?



Lösung

Die richtige Antwort ist Biber B. Er kann in zwei Runden eine Nachricht an alle anderen Biber verbreiten.

In der ersten Runde versendet Biber B die Nachricht seinen Nachbarn, also den mit einem direkten Kommunikationskanal verbunden Bibern A, D und J. Das Bild unten zeigt, wer nach dieser ersten Runde die Nachricht hat.

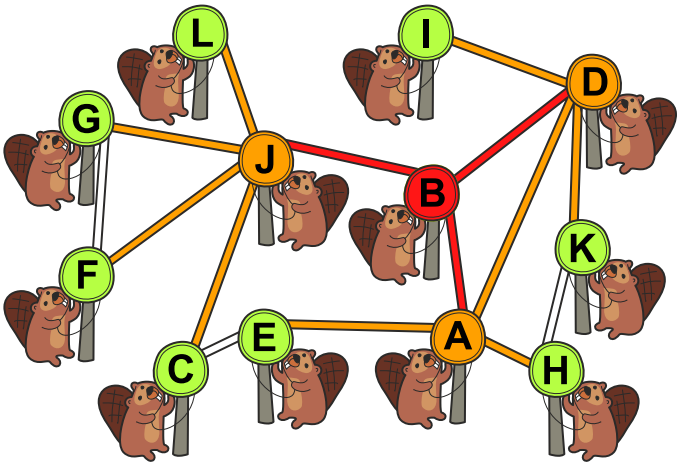


In der zweiten Runde versenden Biber A, D und J die Nachricht jeweils ihren Nachbarn:

- Biber A versendet die Nachricht an die Biber E und H.
- Biber D versendet die Nachricht an die Biber I und K.
- Biber J versendet die Nachricht an die Biber C, F, G und L.

Zusätzlich erhält Biber B die Nachricht gleich dreimal zurück, weil ja auch er ein Nachbar der drei Biber A, D und J ist. Da dies für ihn keine neue Nachricht ist, wird Biber B die Nachricht in den kommenden Runden jedoch nicht mehr versenden. Auch die Biber A und D versenden sich die Nachricht gegenseitig über ihren direkten Kommunikationskanal nochmals zu, danach aber auch nicht mehr weiter, weil die Nachricht für sie dann nicht mehr neu ist.

Das Bild unten zeigt die Situation nach der zweiten Runde.



Die Nachricht hat also alle Biber in nur zwei Runden erreicht.

Schneller geht es nicht, denn sonst müsste ein Biber mit allen anderen Bibern mit einer weissen Linie verbunden sein, um die Nachricht in einer Runde direkt an alle anderen Bibern zu versenden.

Biber B ist der einzige, von dem aus eine Nachricht alle anderen Biber in nur zwei Runden erreicht: Für die Biber C, E, F, G, H, J und L wäre der Biber I nicht in zwei Runden erreichbar. Und für die Biber A, D, E, H, I und K ist der Biber L nicht in zwei Runden erreichbar.

Dies ist Informatik!

Das Kommunikationsnetzwerk der Biber kann man durch einen *Graphen* beschreiben. Jeder Biber befindet sich an einem sogenannten *Knoten*, der in diesem Fall durch einen Buchstaben benannt ist. Die weissen Linien nennt man *Kanten*, sie verbinden jeweils zwei Knoten. Die Nachrichten verbreiten sich im Kommunikationsnetzwerk durch *synchronisierte* Runden, alle Biber versenden also jeweils gleichzeitig. In einer Runde versendet jeder Biber neue Nachrichten an alle Nachbarn. Was die Biber hier tun, nennen Informatiker ein *Broadcasting in einem Kommunikationsnetz*. In der Aufgabe oben hat man untersucht, wie schnell ein solches Broadcasting abgeschlossen werden kann, also wie schnell eine neue Nachricht alle Teilnehmer erreichen kann.

Eine noch anspruchsvollere Aufgabe ist es, Netzwerke so zu gestalten, dass von allen Knoten aus ein schnelles Broadcasting möglich ist, aber die Anzahl der Verbindungen nicht zu hoch ist.

Der Knoten des gesuchten Bibers B nennt man dann das *Zentrum* des Graphen. Abstrakt gesprochen ist das Zentrum ein Knoten, der die Entfernung zu den vom ihm am weitesten entfernten Knoten minimiert. Es gibt also keinen anderen Knoten, der zu allen anderen Knoten eine kleinere Entfernung hätte. In der vorliegenden Aufgabe gibt es nur ein Zentrum. Je nach Graph kann es aber auch mehrere Knoten geben, so dass jeder von ihnen die Entfernung zu den am weitesten von entfernten Knoten minimiert; ein Graph kann also mehrere Zentren haben.

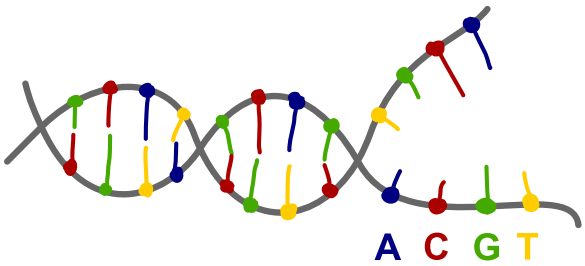
Das Finden eines Zentrums ist nicht immer so einfach wie in der vorliegenden Aufgabe. Zum einen könnte es sein, dass die Übertragung zwischen gewissen direkt verbundenen Knoten mehrere Runden dauert. Zum anderen können die Graphen einfach viel grösser und komplexer sein. Für solche Graphen kann man beispielsweise den *Algorithmus von Floyd und Warshall* verwenden, um effizient ein Zentrum zu finden.

Stichwörter und Webseiten

- Graph: `https://de.wikipedia.org/wiki/Graph_(Graphentheorie)`, `https://de.wikipedia.org/wiki/Weg_(Graphentheorie)#Länge_und_Abstand`
- Zentrum eines Graphen
- Algorithmus von Floyd und Warshall: `https://de.wikipedia.org/wiki/Algorithmus_von_Floyd_und_Warshall`

22. DNA-Sequenz

Unser Erbgut ist in DNA-Sequenzen gespeichert. Eine DNA-Sequenz ist im Wesentlichen eine Abfolge von Basen, die in den vier Typen A, C, G und T auftreten.



Wir betrachten folgende drei Arten von Mutationen:

Mutationsart	**Beschreibung**	**Beispiel**
Ersetzung	Eine einzelne Base wird durch eine andere ersetzt.	AT**G**GT → AT**A**GT
Löschung	Eine einzelne Base wird ersatzlos gelöscht.	ATG**G**T → ATGT
Einfügung	Eine einzelne Base wird irgendwo eingefügt.	ATGGT → A**C**TGGT

*Genau eine der vier folgenden DNA-Sequenzen kann **nicht** entstehen, wenn die Sequenz GTATCG drei Mutationen durchläuft. Welche ist es?*

A) GCAATG
B) ATTATCCG
C) GAATGC
D) GGTAAAC

Lösung

Die richtige Antwort ist D) GGTAAAC.

Auf diese Antwort kommt man am besten durch das Ausschlussverfahren, denn für alle anderen Sequenzen reichen 3 Mutationen aus.

Antwort A: GTATCG ⇒ G**C**ATCG ⇒ GCA**A**CG ⇒ GCAATG
Antwort B: GTATCG ⇒ **A**TATCG ⇒ AT**T**ATCG ⇒ ATTAT**C**CG
Antwort C: GTATCG ⇒ G**A**ATCG ⇒ GAAT**G**G ⇒ GAATG**C**

Hingegen sind 4 Mutationen notwendig, um die Sequenz aus Antwort D) zu erreichen, beispielsweise folgende:

GTATCG ⇒ G**G**TATCG ⇒ GGT**A**ATCG ⇒ GGTAA**A**CG ⇒ GGTAAAC

Dass weniger Mutationen nicht ausreichen, ist nicht ganz einfach zu beweisen.

Dies ist Informatik!

Das Darstellen von Informationen mit *Zeichenketten* (Sequenzen von Buchstaben) und das Arbeiten mit ihnen ist eine zentrale Aufgabe der Informatik.

Ein wichtige Frage ist es, wie stark sich zwei Zeichenketten voneinander unterscheiden. Es gibt verschiedene Methoden, wie man die Unterschiedlichkeit zweier Zeichenketten messen kann. Eine häufige verwendete Messmethode ist die sogenannte *Levenshtein-Distanz*, die mit Hilfe der drei beschriebenen *Mutationsarten* definiert ist: Die Levenshtein-Distanz zwischen zwei Zeichenketten ist die minimale Anzahl von Mutationen, mit der man die eine Zeichenkette in die andere umwandeln kann.

Der übliche Algorithmus zur Berechnung der Levenshtein-Distanz zweier Wörter basiert auf *dynamischer Programmierung*: Dabei schreibt die Levenshtein-Distanzen zwischen immer längeren Präfixen der beiden Wörter in eine Tabelle, bis man am Ende die beiden Präfixe den ganzen Wörtern entsprechen und man das Resultat ablesen kann.

Wenn die Korrektheit des Algorithmus bewiesen ist, kann man damit ausrechnen, dass die Levenshtein-Distanz zwischen der ursprünglichen DNA-Sequenz und jener aus Antwort D) genau 4 ist. Damit ist dann bewiesen, dass weniger Mutationen nicht ausreichen.

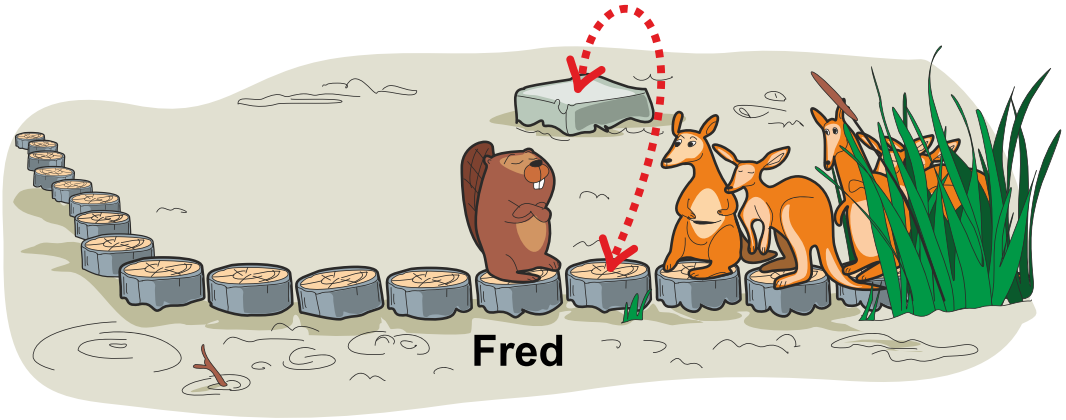
Stichwörter und Webseiten

- Levenshtein-Distanz: `https://de.wikipedia.org/wiki/Levenshtein-Distanz`

23. Sturer Fred

Dem Biber Fred kommen auf einem Baumstumpfpfad Kängurus entgegen. Der Pfad ist ziemlich eng, so dass er und die Kängurus nicht direkt aneinander vorbei können. Es gibt aber einen bestimmten Baumstumpf, von dem aus die Kängurus auf einen Stein ausweichen und von dort wieder zurück zu diesem Baumstumpf hüpfen können, wie im Bild gezeigt. Auf jedem Baumstumpf und dem Stein kann jeweils nur ein einzelnes Tier stehen.

Fred will vorwärts. Er ist ziemlich stur und nur bereit, insgesamt höchstens 10 Mal einen Baumstumpf rückwärts zu gehen. Vorwärts geht er hingegen beliebig oft



Wie viele Kängurus kann Fred maximal passieren lassen?

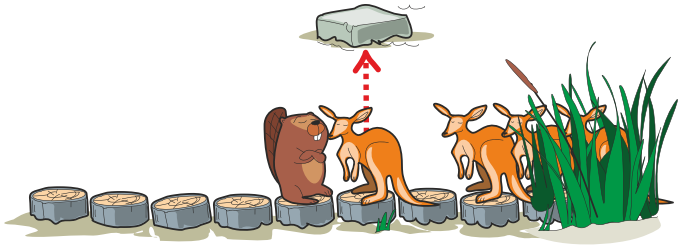
A) Mehr als 10 Kängurus.
B) Genau 10 Kängurus.
C) Genau 6 Kängurus.
D) Genau 4 Kängurus.
E) Weniger als 4 Kängurus.
F) Das kann man nicht genau sagen.

Lösung

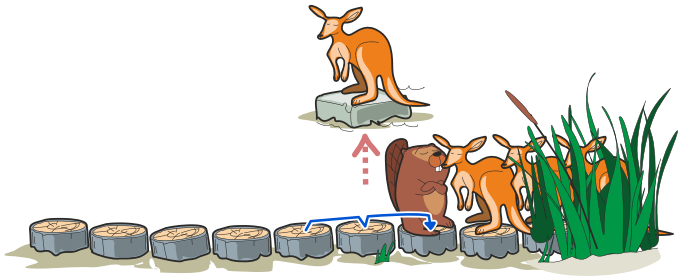
Fred kann maximal genau 6 Kängurus vorbeilassen.

Ein Känguru kommt wie folgt an Fred vorbei:

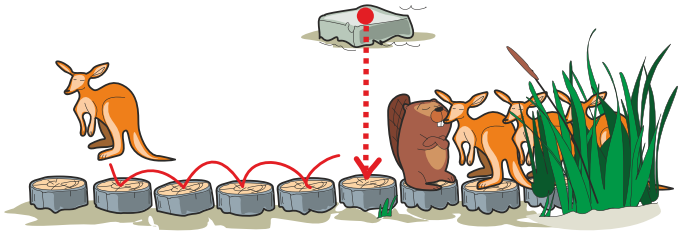
Das Känguru springt auf den Stein.



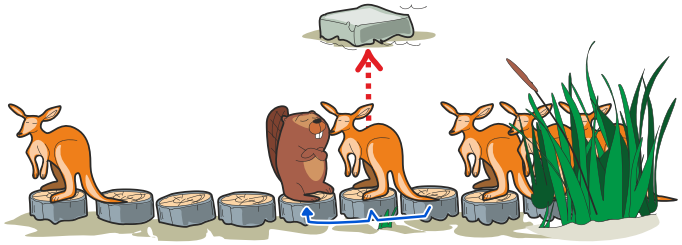
Fred geht zwei Baumstümpfe nach vorne.



Das Känguru springt zurück und setzt seinen Weg fort.



Wenn Fred nun zwei Baumstümpfe zurück geht, ist er wieder auf der Ausgangsposition und kann das Verfahren wiederholen, um jeweils ein weiteres Känguru vorbeizulassen.



Da er maximal 10 Baumstümpfe zurück geht, kann er das fünf Mal tun und zusammen mit dem ersten Känguru maximal 6 Kängurus passieren lassen.

Dies ist Informatik!

In der Informatik werden Aufgaben unter anderem durch Algorithmen gelöst: Folgen einfacher *Anweisungen* und *Befehle*, die Schritt für Schritt ausgeführt werden – genau wie «Fred geht einen Baumstumpf nach vorne» oder «ein Känguru springt auf den Stein».

In einer sogenannten *Schleife* können Folgen von Anweisungen wiederholt werden. Auf diese Weise können gleichförmige Aufgaben zuverlässig mehrfach erledigt werden. Dabei ist es meistens von Vorteil, bei jedem Schleifendurchlauf die gleiche Situation herzustellen – die sogenannte *Schleifeninvariante*.

In unserem Fall muss Fred immer wieder auf seine Ausgangsposition, damit dasselbe Verfahren für das nächste Känguru wieder funktioniert.

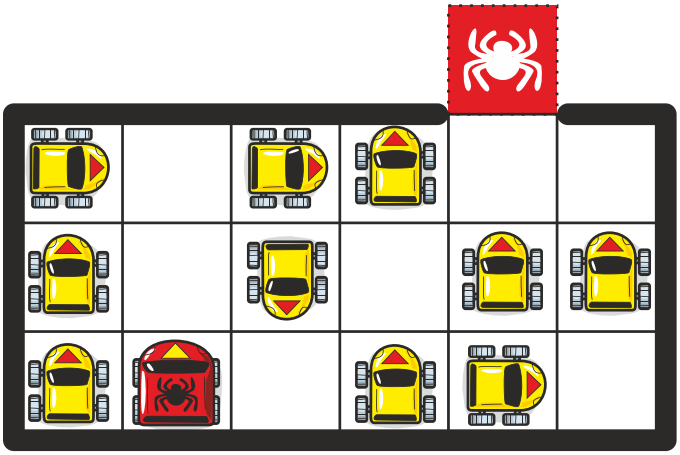
Stichwörter und Webseiten

- Algorithmus: `https://de.wikipedia.org/wiki/Algorithmus`
- `https://de.wikipedia.org/wiki/Strukturierte_Programmierung`
- Schleife
- Invariante

24. Spinnenauto

11 Autos parkieren in einem ummauerten Platz mit einem Ausgang. Jedes Auto hat folgende Möglichkeiten für eine Bewegung:

- Ein Feld vorwärts
- Ein Feld rückwärts
- Eine Vierteldrehung im aktuellen Feld nach rechts oder links



Ein Auto kann auch mehrere Bewegungen ausführen. Auf jedem Feld kann immer nur ein Auto stehen.

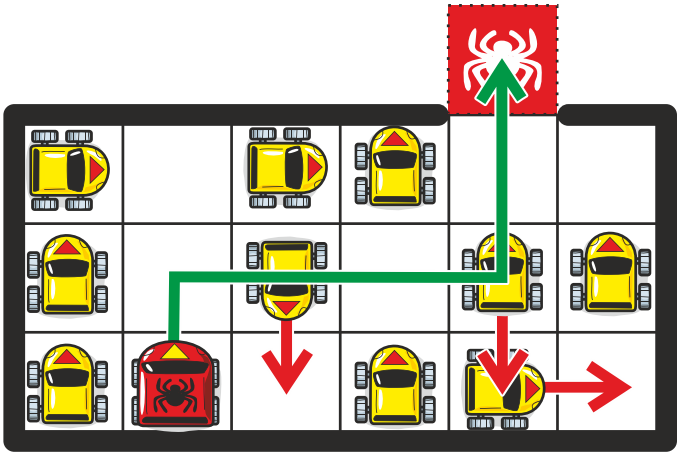
Wie viele Bewegungen der Autos sind insgesamt nötig, um das rote Spinnenauto zum roten Spinnenfeld zu bringen?

A) 9 Bewegungen
B) 11 Bewegungen
C) 13 Bewegungen
D) 15 Bewegungen

Lösung

Die richtige Antwort ist: B) 11 Bewegungen.

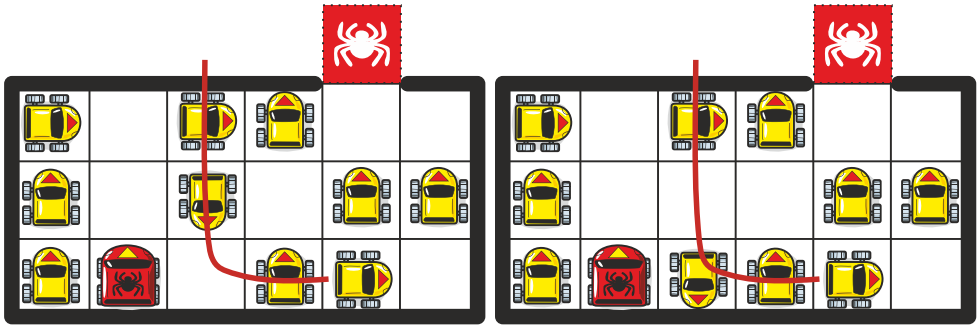
Das Bild zeigt die 11 Bewegungen, um das Spinnenauto zum roten Spinnenfeld zu bringen:



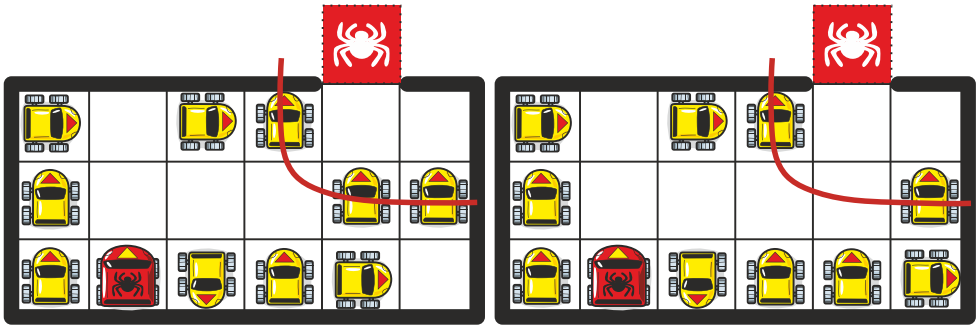
Es muss noch gezeigt werden, dass 11 die minimale Anzahl von Bewegungen ist, die benötigt wird.

Dazu nehmen wir zuerst an, das Spinnenauto sei das einzige Auto auf dem Platz. Um zum roten Spinnenfeld ausserhalb zu gelangen, muss sich das Spinnenauto 3 Mal nach oben und 3 Mal nach rechts bewegen, ausserdem muss es sich 2 Mal drehen. Obwohl dies auf verschiedene Arten erreicht werden kann, benötigt man dazu mindestens $3 + 3 + 2 = 8$ Bewegungen. Das Spinnenauto ist aber nicht das einzige Auto auf dem Platz und es braucht weitere Bewegungen, um den Weg frei zu legen.

Zuerst müssen wir einen Weg durch die L-förmige Barrikade im nächsten Bild finden. Dies kann in einer Bewegung wie folgt geschehen:



Dann müssen wir einen Weg durch eine zweite L-förmige Barrikade finden. Diese Barrikade kann mit nur 1 Bewegung nicht geöffnet werden, 2 reichen aber aus, wie unten gezeigt.



Daher ist die minimale Anzahl Bewegungen $8 + 1 + 2 = 11$ Bewegungen.

Dies ist Informatik!

Dass eine gefundene Lösung optimal ist, ist oft sehr schwierig zu beweisen. Ob es nicht eine bessere Lösung gibt, findet man oft nur heraus, indem man alle möglichen Lösungen durchgeht. Dies Methode nennt man die *Brute Force* (Englisch für *rohe Gewalt*) oder auch *erschöpfende Suche* (Englisch: *Exhaustive Search*), weil man alle Möglichkeiten ausschöpft. Von Hand ist diese Methode zwar meist nicht praktikabel, für den Computer es aber häufig eine einfach umzusetzende Strategie.

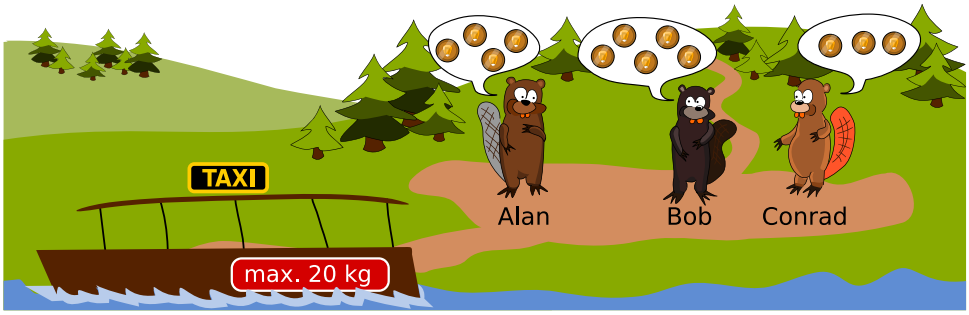
Manchmal gibt es aber so viele verschiedene Lösungen, dass selbst ein Computer damit überfordert ist, alles durchzugehen. In diesen Fällen muss nach einer geeigneteren Strategie gesucht werden. Oft kommen zum Beispiel *Greedy*-Algorithmen (Englisch für *gierig*) oder das *Branch-and-Bound*-Prinzip (Englisch für *Verzweigen und Begrenzen*) zum Einsatz.

Die Aufgabe ist eine Variante des Spiels *Rush Hour*. Auch der Computerspiel-Klassiker *Sokoban* hat viele Ähnlichkeiten.

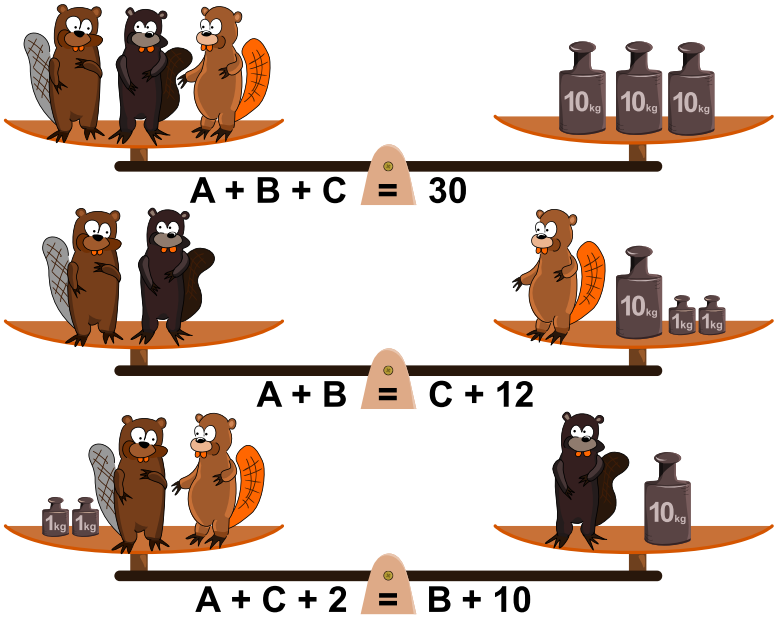
Stichwörter und Webseiten

- Brute Force: `https://de.wikipedia.org/wiki/Brute-Force-Methode`
- Branch and Bound: `https://de.wikipedia.org/wiki/Branch-and-Bound`
- Greedy-Algorithmen: `https://de.wikipedia.org/wiki/Greedy-Algorithmus`
- Rush Hour: `https://de.wikipedia.org/wiki/Rush_Hour_(Spiel)`

25. Wassertaxi



Die drei Biber Alan, Bob und Conrad wollen ein Wassertaxi nehmen. Es gibt nur ein Wassertaxi. Alan würde 4 Bibertaler (4×) bezahlen, Bob jedoch 5 Bibertaler (5×) und Conrad nur 3 Bibertaler (3×). Das Taxi kann höchstens 20 kg tragen. Daher macht der Taxifahrer die folgenden Wägungen:



Welche Biber nimmt der Taxifahrer mit, wenn er möglichst viel verdienen will?

A) Nur Bob
B) Alan und Bob
C) Bob und Conrad
D) Alan und Conrad
E) Alle drei: Alan, Bob und Conrad

Lösung

Die korrekte Antwort ist: C) Bob und Conrad.

Um alle möglichen Lösungen auflisten und dann bewerten zu können, müssen wir zuerst wissen, wie viel die einzelnen Biber wiegen.

Wir wissen, dass alle drei zusammen 30 kg wiegen und daher vom Taxifahrer nicht alle mitgenommen werden können. Stellen wir auf der rechten und linken Seite der zweiten Waage nochmals eine Kopie von C(onrad) drauf, so ergibt sich links $A + B + C = 30$ kg und rechts $C + C + 12$ kg. Daher muss $2C = 18$ kg gelten und daher $C = 9$ kg.

Stellen wir auf der rechten und linken Seite der dritten Waage nochmals eine Kopie von B(ob) drauf, so erhalten wir links $A + B + C + 2$ kg $= 32$ kg und rechts $2B + 10$ kg. Daher gilt $2B = 22$ kg und somit $B = 11$ kg.

Weil $A + B + C = 30$ kg, muss also $A = 10$ kg sein.

Der Taxifahrer kann also:

- Alan und Conrad mitnehmen, dann verdient er $4 + 3 = 7$ Bibertaler.
- Bob und Conrad mitnehmen, dann verdient er $5 + 3 = 8$ Bibertaler.
- Alan und Bob mitnehmen, dann verdient er zwar mit 9 Bibertaler am meisten, aber leider wiegen die beiden zusammen 21 kg und überlasten damit das Wassertaxi.

Daher ist die korrekte Antwort C).

Dies ist aber nicht die einzige Möglichkeit, wie man die Gewichte der Biber bestimmen kann. Ebenso gut hätte man im ersten Schritt auf der ersten Waage links $A + B$ durch $C + 12$ ersetzen könne. Man erhält dann auf der linken Seite $2C + 12$ kg, was gleich 30 kg ist. So schliesst man wieder, dass $C = 9$ kg.

Etwas formaler können die drei Wägungen als ein Gleichungssystem geschrieben werden:

I. $A + B + C = 30$ kg
II. $A + B - C = 12$ kg
III. $A - B + C = 8$ kg

Diese Gleichungen können dann voneinander subtrahiert werden. So liefert die Differenz I – II die Gleichung:

$$2C = 18 \text{ kg} \rightarrow C = 9 \text{ kg}$$

Die Differenz I – III ergibt

$$2B = 22 \text{ kg} \rightarrow B = 11 \text{ kg}$$

Aus I folgt somit $A = 10$ kg.

Dies ist Informatik!

Alle diskreten Optimierungsprobleme aus NP kann man in der Sprache von linearen Gleichungen und Ungleichungen darstellen. (Man spricht dann auch von *linearer Programmierung*.) Die Gleichungen und Ungleichungen sind sogenannte *Einschränkungen*, die die Variablenwerte erfüllen müssen. Man optimiert dann den Wert einer Funktion der Variablen, wobei die Einschränkungen erfüllt sein müssen. In der vorliegenden Aufgabe hat man drei boolesche Variablen x_A, x_B und x_C. Falls $x_A = 1$, wird der Biber A ins Boot genommen, sonst ist $x_A = 0$. Man optimiert die lineare Funktion $4x_A + 5x_B + 3x_C$, wobei man den maximalen Wert sucht. Die einzige Einschränkung ist:

$$Gewicht(A) \cdot x_A + Gewicht(B) \cdot x_B + Gewicht(C) \cdot x_C \leq 20.$$

Man kann die Aufgabe nur vollständig ausformulieren, wenn man die Gewichte der Biber bestimmt. Diese Probleminstanz ist ein Fall des allgemeinen *Rucksackproblems*. Man so viel Wert wie möglich in den Rucksack einpacken, ohne das Gesamtgewicht zu überschreiten.

Noch vor 80 Jahren waren solche Fragestellungen Aufgabe der Mathematiker, doch da immer leistungsfähigere Computer zur Verfügung standen, wurden Lösungsverfahren (z.B. das *Branch-and-Bound-* oder *Schnittebenenverfahren*) entwickelt, mit deren Hilfe man solche Problem lösen kann. Heute werden diese Lösungsverfahren zum Beispiel zur Produktionsoptimierung, in der Logistik oder in Nahverkehrsnetzen eingesetzt.

Trotzdem ist die Lösung von Optimierungsproblemen in der Praxis immer noch eine schwierige Aufgabe, die je nach Grösse und Struktur des Problems eine geschickte Modellierung und speziell entwickelte Algorithmen erfordert. Oft werden mehrere Lösungsverfahren miteinander kombiniert.

Stichwörter und Webseiten

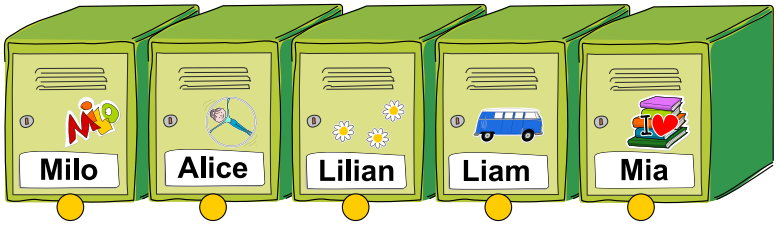
- Ganzzahlige lineare Optimierung: `https://de.wikipedia.org/wiki/Ganzzahlige_lineare_Optimierung`
- Nebenbedingung: `https://de.wikipedia.org/wiki/Nebenbedingung`
- Branch- and Boundverfahren: `https://de.wikipedia.org/wiki/Branch-and-Bound`
- Schnittebenenverfahren: `https://de.wikipedia.org/wiki/Schnittebenenverfahren`

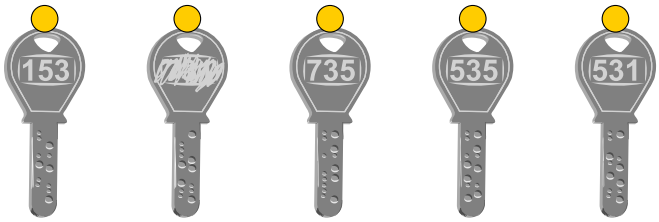
26. Schliessfächer

Fünf Kinder haben an ihrer Schule je ein beschriftetes Schliessfach. Die fünf zugehörigen Schlüssel tragen dreistellige Zahlen. Auf einem Schlüssel ist die Zahl leider zerkratzt.

Jede dreistellige Zahl steht für die ersten drei Buchstaben eines Namens. Eine Ziffer steht überall für denselben Buchstaben, zum Beispiel 8 immer für «C» oder «c».

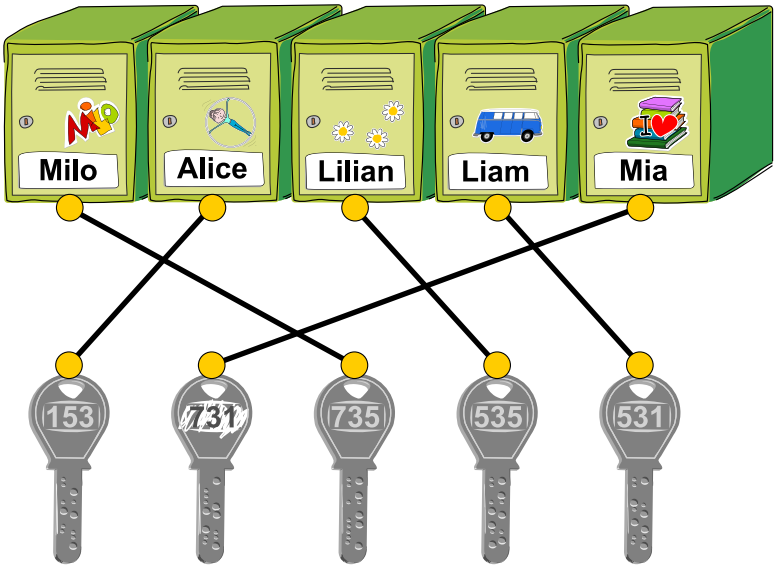
Ordne die Schlüssel den richtigen Schliessfächern zu. Zeichne dazu Linien zwischen den gelben Punkten.





Lösung

Die richtige Lösung ist unten abgebildet:



Die vier bekannten Zahlen sind: 153, 735, 535, 735. Die ersten drei Buchstaben der fünf Namen sind MIL, ALI, LIL, LIA, MIA.

Nur LIL beginnt und endet mit dem gleichen Buchstaben. Dazu muss also eine dreistellige Zahl gehören, die mit der gleichen Ziffer beginnt und endet, und es kann nur eine solche Zahl geben. Die Zahl 535 passt zu diesem Muster, sie muss also zu LIL gehören. Deshalb steht 5 für L und I für 3. Jetzt können wir sehen, dass 531 für LIA stehen muss, denn sonst gibt es keine Namen, die mit L beginnen. Also steht 1 für A. Zudem muss 153 für ALI stehen, weil sonst kein Name ein L an zweiter Stelle hat. Jetzt sind nur noch die Ziffer 7 und der Buchstabe M nicht zugeordnet. Sie müssen also zusammengehören. Wir haben somit folgende eindeutige Zuordnung: 1 = A, 3 = I, 5 = L und 7 = M. Somit steht 735 für MIL und 531 für LIA. Jetzt sehen wir zudem, dass der Schlüssel mit der zerkratzen Zahl Mia gehört und dass die zerkratzte Zahl 731 lauten muss.

Ein alternative Lösungsidee zum Herausfinden der richtigen Zuordnung ist das Zählen der Häufigkeit der Buchstaben und Ziffern. In MIL, ALI, LIL, LIA, MIA kommen die beiden Buchstaben A und M je zweimal vor und die Buchstaben I und L je fünfmal. Leider reicht dies noch nicht für eine eindeutige Zuordnung von Buchstaben zu Ziffern. Man muss deshalb noch mehr Beobachtungen anstellen, zum Beispiel die oben beschriebenen.

Dies ist Informatik!

In der Informatik werden Namen und Texte sehr oft mit Hilfe von Zahlen codiert.

In der Aufgabenstellung ist angegeben, dass man die Zahlen auf den Schüsseln eindeutig aus den ersten drei Buchstaben der jeweiligen Namen ableiten kann. Das funktioniert dadurch, dass man jedem Buchstaben genau einen Ziffer als ihre Codierung zuordnet und nur wenige Buchstaben verwendet. Man spricht von einer *monoalphabetischen Codierung*, weil jeder Buchstabe überall durch dasselbe

Zeichen ersetzt wird. Hingegen war nicht angegeben, welche Ziffer konkret welchem Buchstaben zugeordnet ist. Die Lösung zeigt aber, wie man die richtige Zuordnung schon mit Hilfe weniger struktureller Hinweise herausfinden kann.

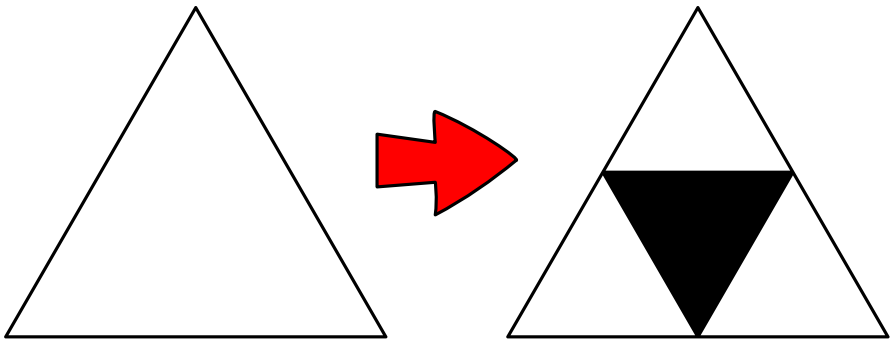
Wenn man nicht nur 10 Ziffern zur Codierung verwendet, sondern ein Symbol für jeden Buchstaben, dann kann man eine solche monoalphabetische Substitution auch als einfache Geheimschrift verwenden. Leider ist die monoalphabetische Verschlüsselungsmethode nicht sehr sicher, weil man die Zuordnung mit ein paar Tricks oft schnell herausfinden kann. Die Aufgabe ist ein Beispiel dafür. Zum Glück gibt es viele bessere Verschlüsselungssysteme. Die *Kryptographie* ist ein wichtiges Teilgebiet der Informatik, in dem man verschiedene Geheimschriften entwickelt und analysiert.

Stichwörter und Webseiten

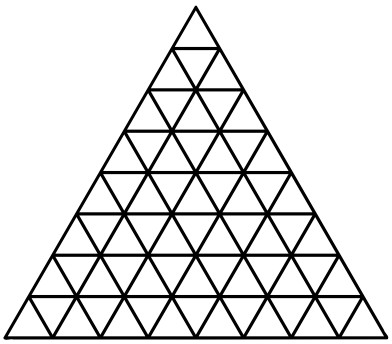
- Codierung, Monoalphabetische Substitution: `https://de.wikipedia.org/wiki/Monoalphabetische_Substitution`
- Kryptographie: `https://de.wikipedia.org/wiki/Kryptographie`

27. Sierpiński-Dreieck

Um ein sogenanntes Sierpiński-Dreieck zu bekommen, zeichnet man zuerst ein gleichseitiges weisses Dreieck. Dann wird schrittweise vorgegangen. In jedem Schritt wird jedes vorhandene weisse Dreieck in vier kleinere unterteilt und das mittlere davon schwarz eingefärbt, so wie es die folgende Abbildung zeigt:

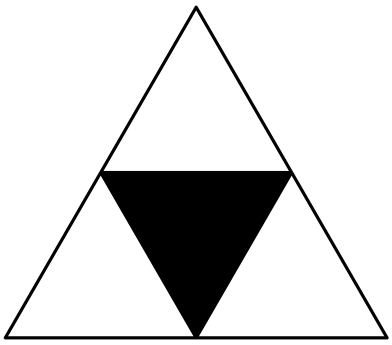


Zeichne die Figur, die nach drei Schritten entsteht. Male dazu die richtigen Teildreiecke an.

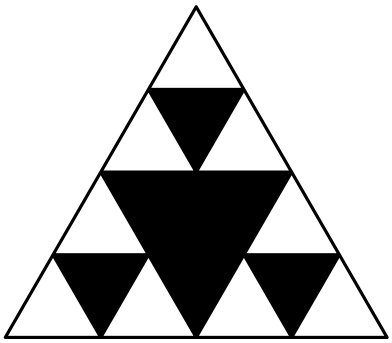


Lösung

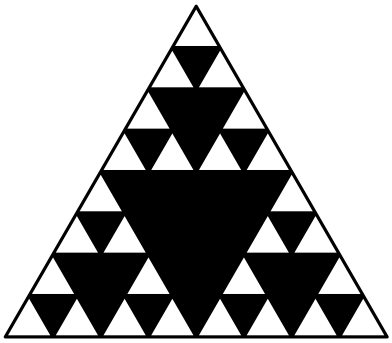
Nach dem ersten Schritt ist das mittlere Teildreieck schwarz und es es bleiben drei weisse Teildreiecke:



Im zweiten Schritt werden diese drei Teildreiecke nochmals in je vier kleinere Teildreiecke unterteilt, wobei das mittlere jeweils schwarz gefärbt wird. Es bleiben $3 \cdot 3 = 9$ kleinere weisse Teildreiecke:



Im dritten und letzten Schritt werden dann diese 9 weissen Teildreiecke nochmals in je vier noch kleinere Teildreiecke unterteilt und jeweils das mittlere angemalt. Es entsteht die folgende Figur mit $3 \cdot 9 = 27$ weissen Teildreiecken:



Dies ist Informatik!

Das Sierpiński-Dreieck ist ein *Fraktal*, das zuerst vom polnischen Mathematiker Wacław Franciszek Sierpiński (1882–1969) im Jahr 1915 beschrieben wurde. Fraktale sind Figuren, in denen immer kleinere und kleinere Teile auftauchen, die der gesamten Figur ähnlich sind. Genaue Bilder von Fraktalen zu zeichnen, ist extrem aufwendig. Als im 20. Jahrhundert Computer aufkamen, die die

notwendigen Berechnungen durchführen konnten, wurde Fraktale sehr populär. Bekannte Fraktale sind die *Koch-Schneeflocke* und die *Mandelbrot-Menge*.

Die Konstruktion des Sierpiński-Dreiecks ist *rekursiv* (vom Lateinischen *re-currere*: zurückrennen, wiederkehren). Das bedeutet Folgendes: Die Anleitung zur Konstruktion enthält eine Anweisung, die besagt, dass man nochmals die gesamte Anleitung ausführen muss. Im Beispiel besagt diese Anleitung Folgendes: «Teile das weisse Dreieck in vier kleinere Dreiecke auf, färbe das mittlere davon schwarz ein, und wiederhole diese Anleitung für die drei anderen Dreiecke.» Einen Durchgang der Anleitung nennt man einen *Rekursionsschritt*, und die Anweisungen zum erneuten Durchgehen der Anleitung nennt man *Rekursionsaufrufe*. (Im Beispiel gibt es drei Rekursionsaufrufe pro Rekursionsschritt.) Weil in jedem Rekursionsaufruf wieder neue Rekursionsaufrufe stecken, muss man den Rekursionsschritt immer und immer wieder ausführen, was unendlich lange dauert. Vermeiden kann man das mit einer *Abbruchbedingung*. Im Beispiel stoppen die rekursiven Aufrufe, wenn die Dreiecke zu klein werden.

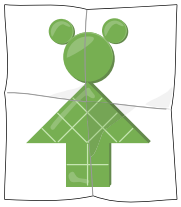
Das Konzept der *Rekursion* wird in der Informatik breit eingesetzt. Denn viele komplexe Objekte – zum Beispiel Fraktale – können mit Rekursion kompakt beschrieben werden und viele komplizierte Aufgaben – zum Beispiel das Problem der *Türme von Hanoi* – können mit sehr einfachen rekursiven Algorithmen gelöst werden.

Stichwörter und Webseiten

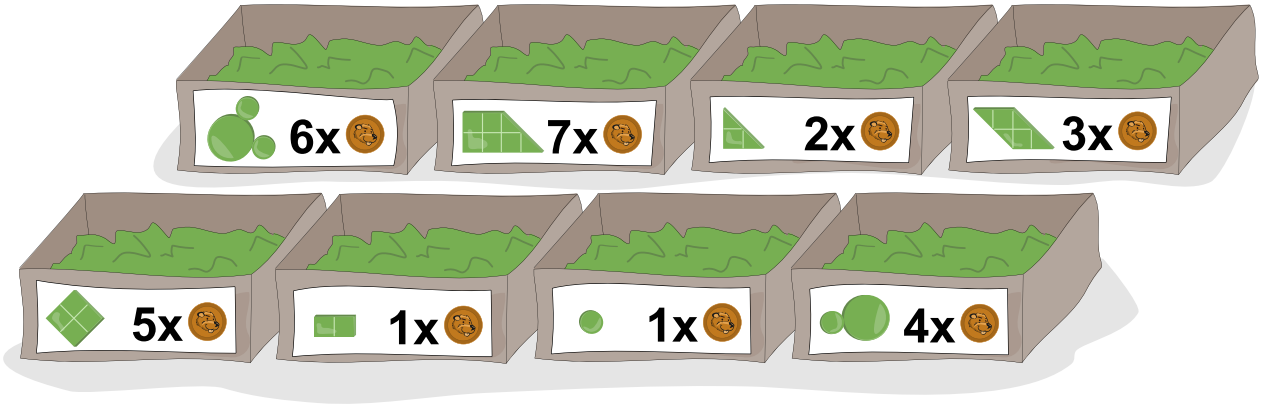
- Sierpiński-Dreieck: `https://de.wikipedia.org/wiki/Sierpinski-Dreieck`
- Rekursion: `https://de.wikipedia.org/wiki/Rekursion`
- Fraktal: `https://de.wikipedia.org/wiki/Fraktal`
- `https://de.wikipedia.org/wiki/Wacław_Sierpiński`
- `https://de.wikipedia.org/wiki/Türme_von_Hanoi#Rekursiver_Algorithmus`
- `https://de.wikipedia.org/wiki/Koch-Kurve`
- `https://de.wikipedia.org/wiki/Mandelbrot-Menge`

28. Legespiel

Giulia will Plättchen kaufen, um damit diese Figur legen:



Der Spielzeugladen bietet verschiedene Plättchen in beliebiger Menge an. Die Preise pro Plättchen variieren von 1 bis 7 Münzen.



Die Plättchen können beim Legen beliebig gedreht und gewendet werden, sie dürfen sich aber nicht überlappen.

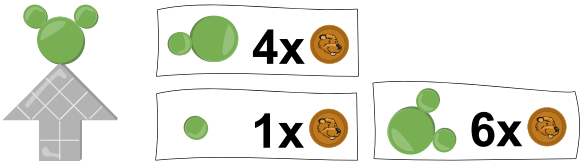
Wie viele Münzen muss Giulia ausgeben, wenn sie die günstigste Option wählt?

A) 13 Münzen
B) 14 Münzen
C) 16 Münzen
D) 20 Münzen

Lösung

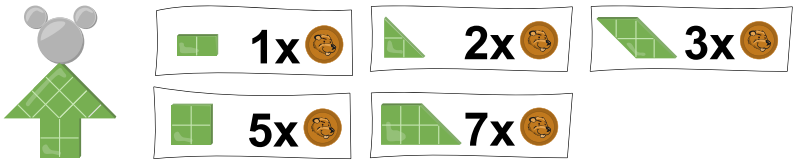
Die richtige Antwort lautet 13 Münzen.

Ein Lösungsansatz ist, dass man einzelne Teile der Figur separat betrachtet. Am einfachsten beginnt man beim Kopf, der nur mit runden Plättchen gelegt werden kann:

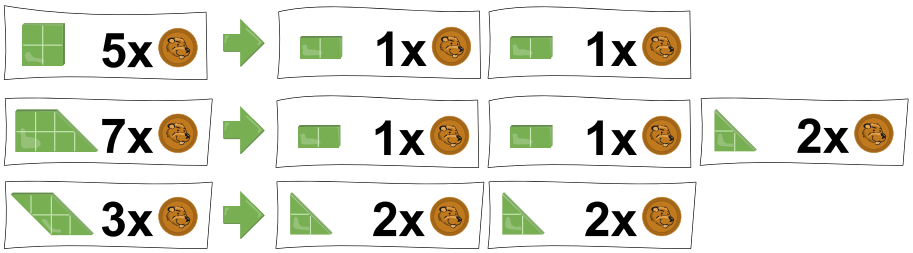


Für den Kopf gibt es nur zwei Möglichkeiten: Entweder benutzt man direkt das passende Plättchen für 6 Münzen oder aber man setzt es aus den anderen beiden runden Plättchen zusammen, die zusammen $4 + 1 = 5$ Münzen kosten. Die zweite Option ist günstiger, wir benutzen also diese.

Der Rest der Figur kann nur aus eckigen Plättchen zusammengesetzt werden.



Man könnte jetzt alle möglichen Varianten zum Legen der Figur durchprobieren und für alle den Preis ausrechnen. Das ist aber sehr aufwendig. Folgende Beobachtungen zu den eckigen Plättchen führen schneller auf die Lösung:



- Ein Quadrat-Plättchen für 5 Münzen kann immer durch zwei Rechtecke für $1 + 1 = 2$ Münzen ersetzt werden. Das macht es immer günstiger.
- Man könnte ein Quadrat-Plättchen stattdessen auch durch zwei Dreiecke ersetzen, das wäre mit $2 + 2 = 4$ Münzen aber wieder etwas teurer, dies ist also die schlechtere Option.

Deshalb kauft Giulia nie ein Quadrat, auch wenn es irgendwo gut passen würde, sondern stattdessen immer zwei Rechtecke.

- Ein Trapez für 7 Münzen kann durch ein Quadrat und ein Dreieck zusammengesetzt werden. Wenn wir das Quadrat durch zwei Rechtecke ersetzen, reichen also $1 + 1 + 2 = 4$ Münzen für ein Trapez.

Also kauft Giulia nie direkt ein Trapez, auch wenn eines gut passen würde, sondern setzt es stattdessen immer mit zwei Rechtecken und einem Dreieck zusammen.

- Das Parallelogramm für 3 Münzen könnte durch zwei kleinere Dreiecke für $2 + 2 = 4$ Münzen ersetzt werden. Das macht es aber nur teurer, ist also keine gute Option. Ein Parallelogramm könnte nützlich sein für Giulia, aber ob das wirklich so ist, zeigt erst eine genauere Untersuchung.

Version A

- Kopf aus 5 Münzen
- Körper aus 4 Rechtecken und 2 Dreiecken:
  $1 + 1 + 1 + 1 + 2 + 2 = 8$ Münzen

Version B

- Kopf aus 5 Münzen
- Körper aus 1 Parallelogramm, 2 Rechtecken und 2 Dreiecken:
  $3 + 1 + 1 + 2 + 2 = 9$ Münzen

Verwendet Giulia kein Parallelogramm, dann benötigt sie zwei Dreiecke, um die dreieckigen Spitzen links und rechts in der Figur zu legen. Den Rest kann sie dann mit Rechtecken legen, so wie in der Version A, die total $5 + 8 = 13$ Münzen kostet.

Das Parallelogramm passt nur auf eine Weise in die Figur, wie in Version B gezeigt (oder spiegelverkehrt dazu). Wenn ein Parallelogramm so gelegt wird und der Rest mit Rechtecken und Dreiecken aufgefüllt wird, kostet die Figur $5 + 9 = 14$ Münzen. Alle anderen Platzierungen des Parallelogramms würden zu Lücken führen, die nicht aufgefüllt werden können.

Somit ist klar, dass 13 Münzen die günstigste Lösung ist.

Dies ist Informatik!

Die Aufgabe mit vorgegebenen Plättchen eine gewisse Figur zu legen, kann schon für sehr wenige Teile extrem kompliziert sein. Ein Beispiel ist das Legespiel Tangram.

Das vorliegende Problem ist sogar noch komplizierter, weil man zusätzlich den Gesamtpreis der Plättchen optimieren muss. Ein solches Problem nennt man in der Informatik ein *Optimierungsproblem*.

Gelöst wurde das Problem mit einem wichtigen Prinzip der Informatik: Ein Problem in kleinere Teilprobleme aufteilt, die man unabhängig voneinander lösen kann und deren Lösungen sich dann zu einer Gesamtlösung zusammensetzen lassen. Konkret wurde das Problem in zwei unabhängig lösbare Teilprobleme aufgeteilt, eines für die runden Plättchen und eines für die eckigen Plättchen. Bei den eckigen Plättchen kann man wiederum die günstigste Plättchenkombination für ein Quadrat überall wiederverwenden, ohne immer wieder darüber nachdenken zu müssen. Ebenso beim Parallelogramm.

Das Aufteilen in unabhängige Teilprobleme ist beim Programmieren sehr wichtig. Das Wiederverwenden von Lösungen für mehrfache auftretende Teilprobleme kann viel Zeit sparen. Man spricht hier vom Prinzip der *Modularität*. Das Aufteilen in kleinere Teilprobleme ist auch die Grundlage für Programme nach dem Prinzip «*Teile und herrsche*» (Lateinisch «Divide et impera», Englisch «Divide and Conquer»).

Stichwörter und Webseiten

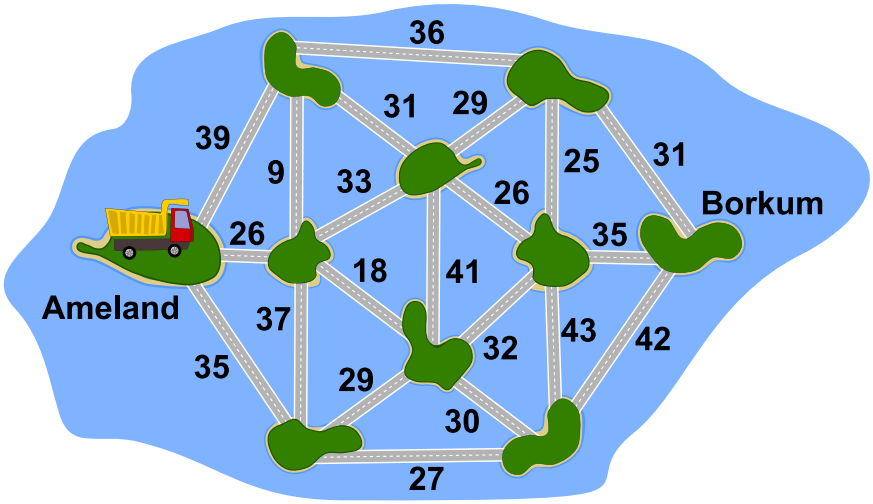
- Optimierungsproblem: `https://de.wikipedia.org/wiki/Optimierungsproblem`
- Teile und Herrsche: `https://de.wikipedia.org/wiki/Teile-und-herrsche-Verfahren`
- Modularität: `https://de.wikipedia.org/wiki/Modularität`
- Tangram: `https://de.wikipedia.org/wiki/Tangram`

29. Biberseeland

Biberseeland besteht aus zehn Inseln, die durch Brücken verbunden sind. Unten ist eine Karte. Die Zahl an jeder Brücke zeigt das maximal zulässige Gesamtgewicht in Tonnen für einen Lastwagen, der diese Brücke überqueren möchte.

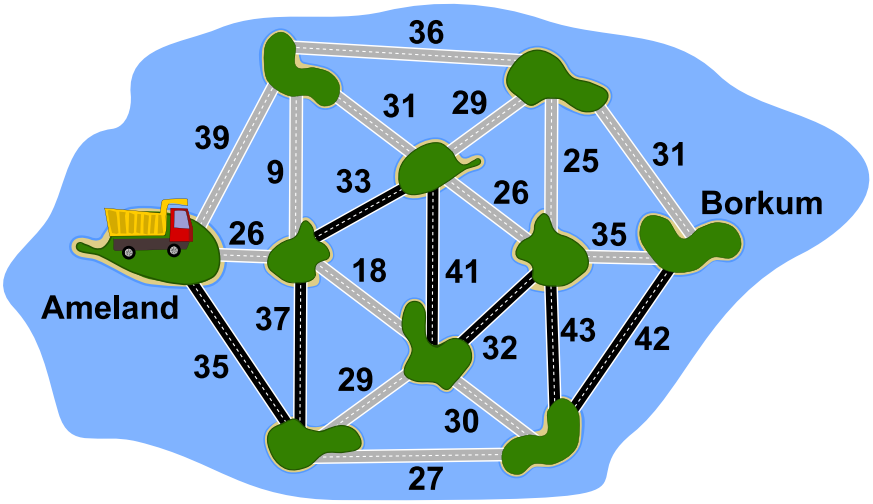
Biber Knuth möchte auf der Insel Borkum einen Strand aufschütten. Mit einer Fahrt will er daher möglichst viel Sand von der Insel Ameland zur Insel Borkum transportieren. Dabei ist ihm die Länge der Fahrtstrecke egal, er will aber über keine Brücke zweimal fahren.

Welchen Weg nach Borkum sollte er mit seinem Lastwagen nehmen?

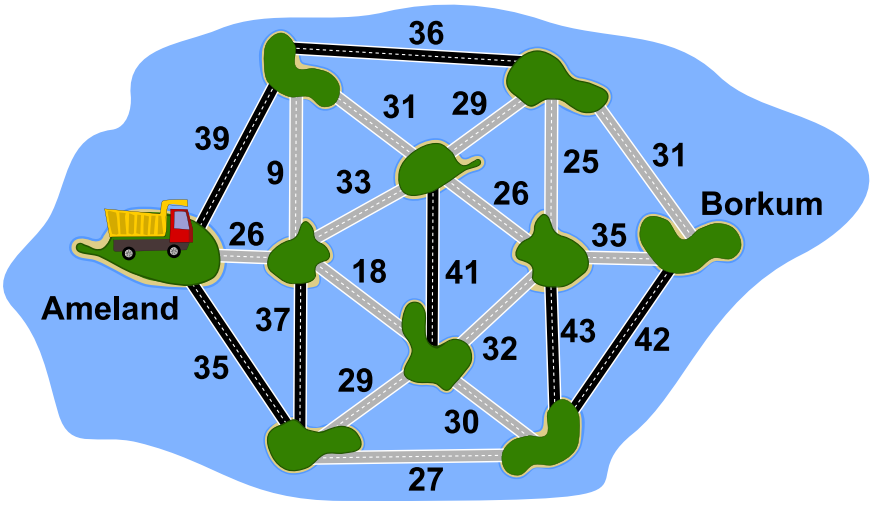


Lösung

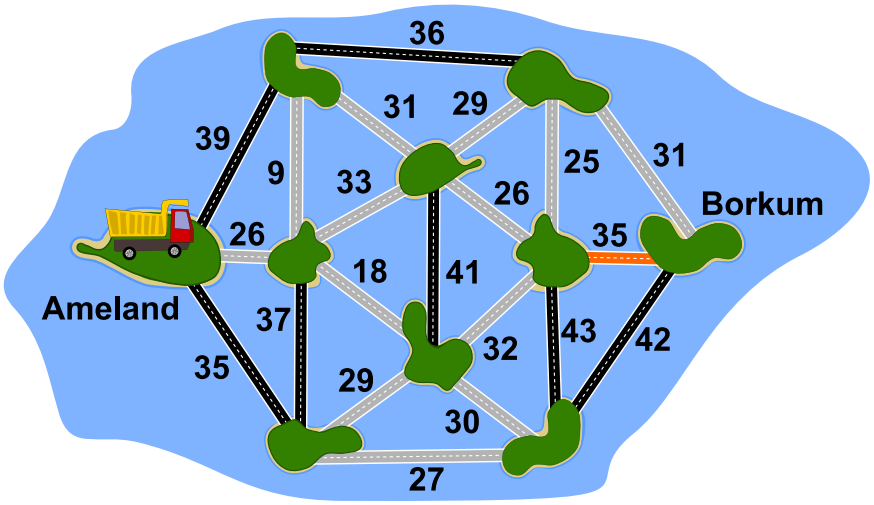
Für die Fahrt beträgt das maximale Gesamtgewicht eines Lastwagens 32 Tonnen. Er nimmt zum Beispiel den folgenden Weg:



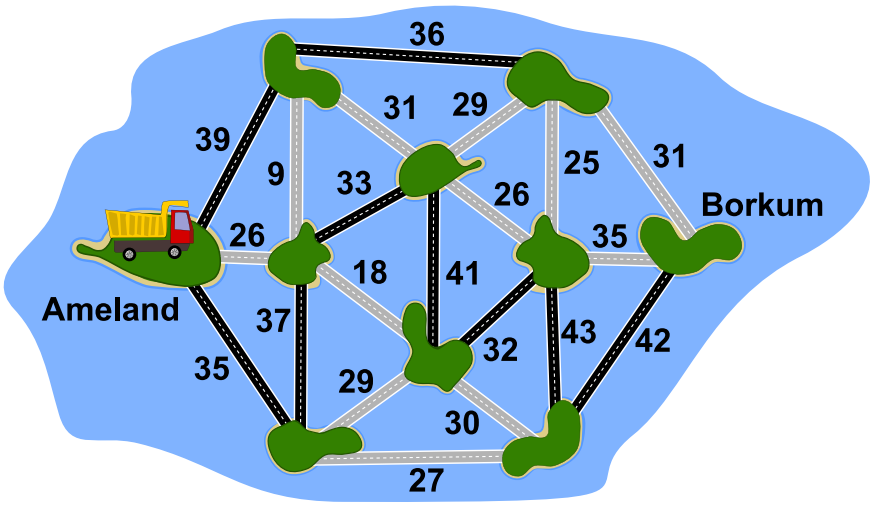
Um diesen zu bestimmen, können wir zum Beispiel virtuell zunächst alle Brücken aus der Karte entnehmen. Alle Brücken werden nach ihrer Belastbarkeit sortiert. Wir fangen mit denjenigen mit der grössten Belastbarkeit an und fügen diese der Karte zu. Danach kommen diejenigen mit der nächstgrössten Belastbarkeit und so weiter. Im folgenden Diagramm sind die eingefügten Brücken mit den Belastbarkeiten 43, 42, 41, 39, 37, 36, 35 schwarz markiert.



Würden wir allerdings mit dem Einfügen einer Brücke einen sogenannten Zyklus bilden, also einen Rundweg, lassen wir diese doch weg, denn dann sind ja alle Inseln dieses Zyklus bereits durch Brücken höherer Kapazität erschlossen. In folgendem Diagramm würde die nächste Brücke mit Belastbarkeit 35 eingetragen, diese würde aber nur einen Weg abkürzen, den es bereits gibt.



Das machen wir, bis alle Inseln miteinander verbunden sind. Nun gibt es nur einen möglichen Weg zwischen zwei beliebigen Inseln und die Brücke mit der kleinsten Kapazität gibt das gesuchte maximale Gewicht an.



Dies ist Informatik!

Eine reale Anwendung für die Lösung der Biberseeland-Aufgabe ist es, in Computernetzen den «Flaschenhals» zu identifizieren, also die grösste überhaupt mögliche Übertragungsrate zwischen zwei Computern im Netzwerk. Die Aufgabe hier betrachtet als Flaschenhals das maximale Gesamtgewicht eines Lastwagens auf dem Weg zwischen zwei Inseln. Dieses wird durch die Tragfähigkeit der schwächsten Brücke bestimmt. In Computernetzen wäre das also die Verbindung mit der geringsten Bandbreite.

Für eine Lösung kann man wie hier präsentiert das Netzwerk zunächst modellieren, also vereinfachen. In unserem Fall wird durch den *Kruskal-Algorithmus* ein *maximaler Spannbaum* erstellt, in dem der Flaschenhals direkt ersichtlich ist.

Stichwörter und Webseiten

- Graph: `https://de.wikipedia.org/wiki/Graph_(Graphentheorie)`
- Minimaler Spannbaum: `https://de.wikipedia.org/wiki/Spannbaum`
- Kruskal-Algorithmus: `https://de.wikipedia.org/wiki/Algorithmus_von_Kruskal`

30. Beschädigte Tabelle

Die Biber verwenden eine Geheimschrift, in der man jeden Buchstaben durch ein ganz neues Zeichen ersetzt. Wie man die neuen Zeichen erzeugt, ist in der Tabelle unten beschrieben. Leider ist die Tabelle nicht vollständig, weil einige Teile verwischt worden sind.

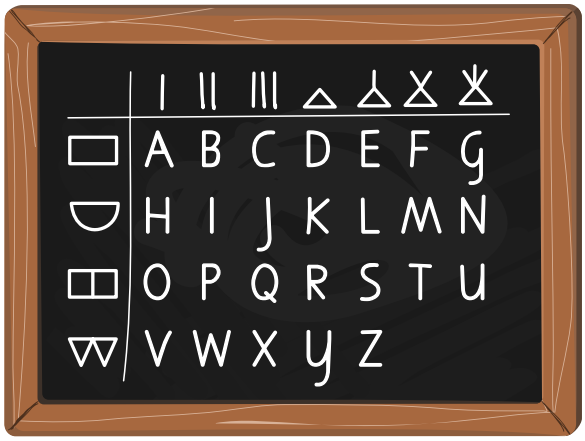
Rekonstruiere den ursprünglichen Text aus dem vorliegenden Geheimtext (dechiffriere den Geheimtext). Welcher der 4 Lösungsvorschläge stimmt?

A) INFORMATIK IST TOLL
B) MATHEMATIK IST TOLL
C) INFORMATION GEHEIM
D) INFORMIERE UNS HIER

Lösung

Die richtige Antwort ist A), der Klartext lautet: INFORMATIK IST TOLL.

Hier ist die vollständige Geheimschrift-Tabelle:



Man kann die Tabelle einfach rekonstruieren. Die Buchstaben des lateinischen Alphabets sind zeilenweise in der Reihe von links nach rechts gesetzt. Man bemerkt, dass neue Zeichen so zusammengesetzt sind, dass die Zeilenbezeichnung den unteren Teil und die Spaltenbezeichnung den oberen Teil ausmachen. Der einzige fehlende untere Teil, der im Geheimtext vorkommt, ist das ▭. Somit ist dieses Zeichen die Bezeichnung der ersten Zeile. Genauso schnell kann man die drei fehlenden Zeichen für die Spalten ermitteln.

Es ist aber nicht notwendig die Tabelle vollständig wiederherzustellen. Man kann die Buchstaben einsetzen, die man von der beschädigten Tabelle direkt ablesen kann. So erhält man den folgenden Lückentext:

I N _ O _ _ _ _ I _ I S _ _ O L L

Mit diesem Lückentext kann man alle Lösungen ausser A) ausschliessen: B) beginnt mit «MA», C) endet mit «EIM», D) endet mit «IER».

Ein anderer Lösungsansatz ist der, dass man erkennt, dass der Geheimtext mit zwei gleichen Zeichen endet. Somit kommen nur noch A) und B) in Frage. Das erste Zeichen kann man in der beschädigten Tabelle eindeutig als «I» identifizieren, womit klar ist, dass die richtige Lösung A) ist.

Dies ist Informatik!

Informationen geheim zu halten oder Daten zu schützen ist eine 4000 Jahre alte Aufgabe. Unzählige Geheimsprachen wurden zu diesem Zweck entwickelt und benutzt. Heute ist Datensicherheit eines der Kernthemen der Informatik. Eine der Methoden, Daten vor unbefugtem Lesen zu schützen, ist sie zu *chiffrieren*. Das Chiffrieren verwandelt einen *Klartext* in einen *Geheimtext*. Das Rekonstruieren des Klartextes aus dem Geheimtext nennt man *Dechiffrieren*. Die Lehre der Geheimschriften nennt man *Kryptologie*.

Die antiken Kulturen verwendeten meistens Geheimschriften, die durch Codierung von Buchstaben mit anderen Buchstaben oder ganz neuen Zeichen erzeugt worden sind. Die Geheimschrift hier ist speziell für den Informatik-Biber entwickelt worden, basiert aber auf einem Konzept aus dem antiken Palästina. Die damalige Sicherheitsregel war, dass nur Geheimschriften verwendet worden sind, die man leicht auswendig lernen kann. Eine schriftliche Beschreibung der Geheimschrift aufzubewahren, betrachtete man als zu grosses Risiko. Eine Tabelle, wie sie hier verwendet wird, kann man gut auswendig lernen. Die berühmte Geheimschrift der Freimaurer basiert auf diesem Prinzip.

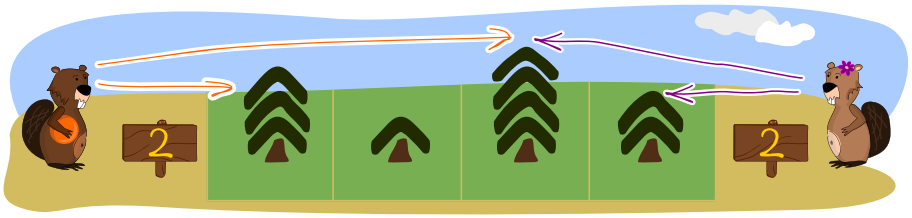
Stichwörter und Webseiten

- Kryptologie: `https://de.wikipedia.org/wiki/Kryptologie`
- Geheimschrift
- Chiffrieren
- Dechiffrieren

31. 4×4-Baum-Sudoku

Die Biber pflanzen sechzehn Bäume (vier Bäume der Höhe 4, vier Bäume der Höhe 3, vier Bäume der Höhe 2 und vier Bäume der Höhe 1) in ein Baumfeld der Grösse 4×4. Dabei beachten sie folgende Regeln:

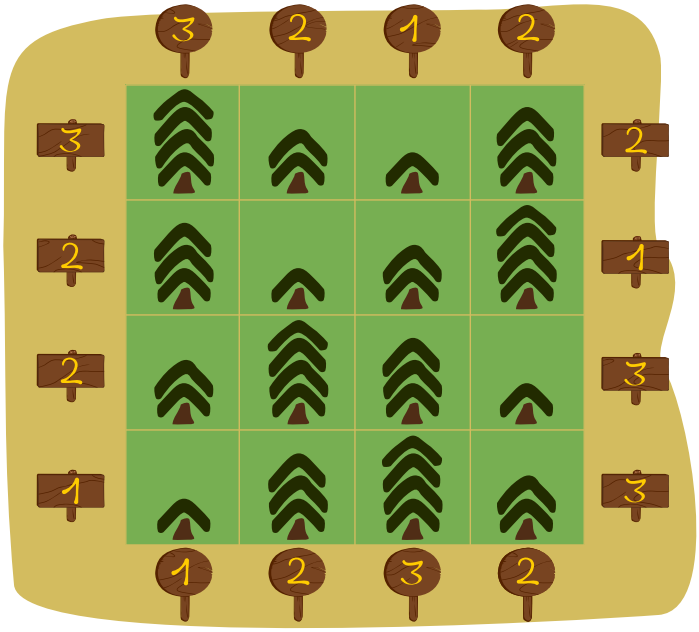
- In jeder Zeile (horizontalen Reihe) gibt es von jeder Höhe genau einen Baum.
- In jeder Spalte (vertikalen Reihe) gibt es von jeder Höhe genau einen Baum.



Wenn sich die Biber eine Tannenreihe von einem Ende her anschauen, dann können sie niedrigere Bäume, die hinter höheren Bäumen versteckt sind, **nicht** sehen. Am Ende jeder Baumreihe steht auf einem Schild, wie viele Bäume ein Biber von dieser Stelle sehen kann. Diese Schilder mit der Anzahl sichtbarer Bäume stehen rund um das Baumfeld.

Kubko versuchte die Beschreibung des Feldes auf ein Blatt Papier zu übertragen. Er hat die Zahlen der Schilder richtig übertragen, aber bei vier Bäumen hat er Fehler gemacht.

Kannst Du die vier Positionen mit falsch eingetragenen Bäumen finden und sie korrigieren?



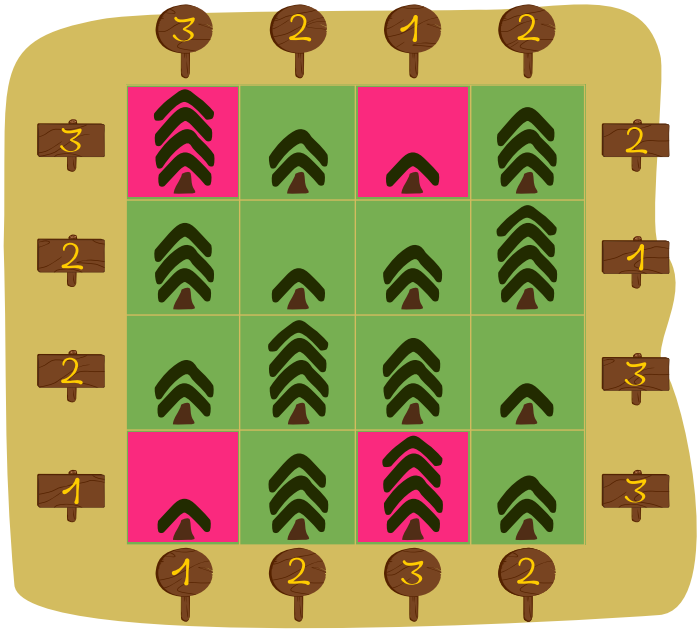
Lösung

Zuerst bemerkt man, dass beide «Sudoku»-Regeln eingehalten worden sind: In jeder Reihe gibt es genau einen Baum von jeder Höhe.

Danach kann man schauen, für welche Reihen die Zahlen auf den Schildern stimmen und für welche nicht. Dabei stellt man fest, dass für die Zeilen 2 und 3 und für die Spalten 2 und 4 die Zahlen stimmen. Für die anderen Reihen stimmen die Zahlen nicht, wir nennen diese Reihen *problematisch*.

Das genügt noch nicht. Man will wissen, welche Positionen die falschen Zahlen verursachen. Dazu bemerkt man, dass es genau vier Positionen gibt, die sich gleichzeitig in einer problematischen Zeile und problematischen Spalten befinden. Es sind die vier Positionen, wo sich die problematischen Zeilen (das sind 1 und 4) mit den problematischen Spalten (das sind 1 und 3) kreuzen.

Wenn man die Baumpaare an diesen vier problematischen Kreuzungen (unten rot markiert) innerhalb der Zeilen oder Spalten austauscht, erhält man die korrekte Lösung.



Dass dies tatsächlich auch die einzige mögliche Lösung ist, kann man wie folgt sehen: Es sind gemäss Aufgabenstellung genau vier Bäume falsch angegeben. Wenn an einer Position ein Baum geändert wird, müssen mindestens zwei weitere geändert werden, damit die Sudoku-Regel erfüllt bleibt, nämlich je ein weiterer in der betroffenen Zeile und Spalte. Somit hat man schon drei geänderte Bäume. Die letzten beiden Änderungen erzwingen wiederum je eine weitere Änderung in der neu betroffenen Zeile und Spalte. Weil total nur vier Änderungen gemacht werden dürfen, ist das nur möglich, wenn die letzten beiden Änderungen zusammenfallen. Das geht nur, wenn die vier Positionen mit Änderungen in einem Rechteck angeordnet sind. Weil in jeder problematischen Reihe mindestens eine Änderung vorgenommen werden muss, ergibt sich die obige Lösung als einzige Möglichkeit.

Dies ist Informatik!

Diese Aufgabe fokussiert auf drei grundlegende Kompetenzen von Informatikerinnen und Informatikern.

Zuerst geht es darum, eine Lösung zu finden, die die gegebenen Einschränkungen einhält, oder nach Bedarf einen Lösungsvorschlag zu korrigieren.

Zweitens geht es um die Fähigkeit, Objekte über ihre Darstellung aus Teilinformationen rekonstruieren zu können. Das hängt mit der Generierung von Objekten (*Objektdarstellungen*) aus eingeschränkten verfügbaren Informationen zusammen, wenn man die Gesetzmässigkeit des Objektes kennt. Solche Vorgehensweisen kann man auch bei der *Komprimierung* anwenden.

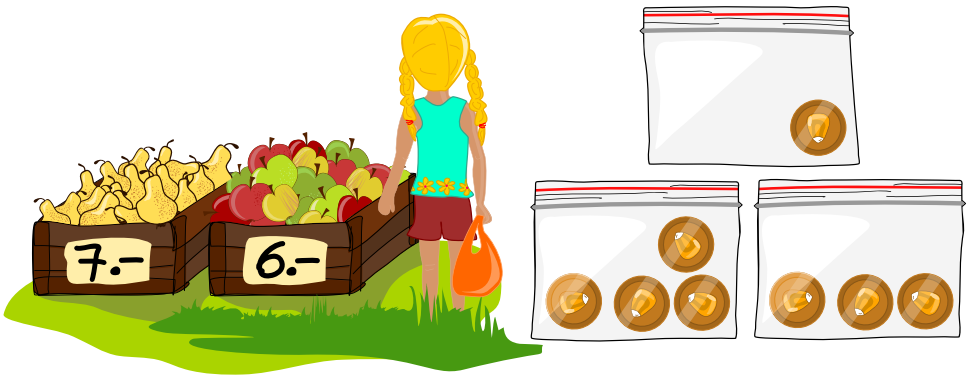
Drittens kann man solche Baumfelder mit Schildern zur Erzeugung von *selbst-verifizierenden Codierungen* einsetzen. Vorkommende Fehler beim Eintragen oder beim Informationstransport können dann automatisiert erkannt oder sogar korrigiert werden.

Stichwörter und Webseiten

- Sudoku: `https://de.wikipedia.org/wiki/Sudoku`
- Objektdarstellung
- Komprimierung: `https://de.wikipedia.org/wiki/Datenkompression`
- Fehlererkennung und Fehlerkorrektur:
  `https://de.wikipedia.org/wiki/Fehlerkorrekturverfahren`

32. Geldtransport

Bina geht gerne schwimmen. Dazu verpackt sie ihr Geld jeweils in wasserdichte Beutel, damit das Metall nicht zu rosten beginnt. Gestern hatte Bina drei Beutel mit 1, 3 und 4 Münzen dabei. Damit konnte sie zwar eine Birne passend (also ohne Rückgeld) mit verschlossenen Beuteln bezahlen, aber nicht einen Apfel.



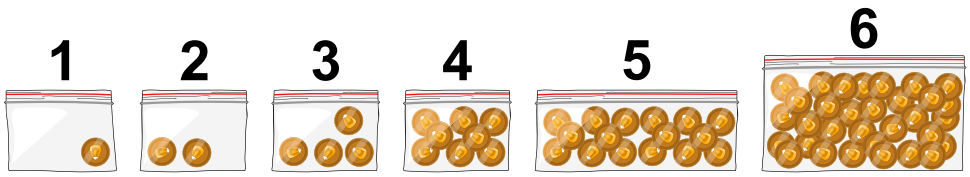
Heute hat Bina 63 identische Münzen dabei. Diese möchte sie so auf verschiedene Beutel aufteilen, dass sie jeden Betrag zwischen 1 und 63 Münzen mit verschlossenen Beuteln passend bezahlen kann.

Was ist die kleinste Anzahl Beutel, mit der Bina auskommt?

A) 4 Beutel
B) 5 Beutel
C) 6 Beutel
D) 7 Beutel
E) 8 Beutel
F) 15 Beutel
G) 16 Beutel
H) 31 Beutel
I) 32 oder mehr Beutel

Lösung

Die richtige Antwort ist C) 6 Beutel:



Bina kann die Münzen wie folgt auf die 6 Beutel aufteilen:

- Beutel 1: 1 Münze
- Beutel 2: 2 Münzen
- Beutel 3: 4 Münzen
- Beutel 4: 8 Münzen
- Beutel 5: 16 Münzen
- Beutel 6: 32 Münzen

Bina hat dann total $1 + 2 + 4 + 8 + 16 + 32 = 63$ Münzen in den Beuteln und kann jeden Gesamtbetrag von 1 bis zu 63 Münzen passend mit verschlossenen Beuteln bezahlen.

Um 13 Münzen zu bezahlen, kann sie beispielsweise mit den Beuteln 1, 3 und 4 bezahlen.

Die Tabelle unten zeigt, wie jeder Gesamtbetrag von mit der richtigen Auswahl von diesen 6 Beuteln passend bezahlt werden kann. Eine Zelle enthält eine 1, wenn Bina den entsprechenden Beutel zur Bezahlung benützt, und sonst 0.

Betrag	**32**	**16**	**8**	**4**	**2**	**1**	**Betrag**	**32**	**16**	**8**	**4**	**2**	**1**
0	0	0	0	0	0	0	**32**	1	0	0	0	0	0
1	0	0	0	0	0	1	**33**	1	0	0	0	0	1
2	0	0	0	0	1	0	**34**	1	0	0	0	1	0
3	0	0	0	0	1	1	**35**	1	0	0	0	1	1
4	0	0	0	1	0	0	**36**	1	0	0	1	0	0
5	0	0	0	1	0	1	**37**	1	0	0	1	0	1
6	0	0	0	1	1	0	**38**	1	0	0	1	1	0
7	0	0	0	1	1	1	**39**	1	0	0	1	1	1
8	0	0	1	0	0	0	**40**	1	0	1	0	0	0
9	0	0	1	0	0	1	**41**	1	0	1	0	0	1
10	0	0	1	0	1	0	**42**	1	0	1	0	1	0
11	0	0	1	0	1	1	**43**	1	0	1	0	1	1
12	0	0	1	1	0	0	**44**	1	0	1	1	0	0
13	0	0	1	1	0	1	**45**	1	0	1	1	0	1
14	0	0	1	1	1	0	**46**	1	0	1	1	1	0
15	0	0	1	1	1	1	**47**	1	0	1	1	1	1
16	0	1	0	0	0	0	**48**	1	1	0	0	0	0
17	0	1	0	0	0	1	**49**	1	1	0	0	0	1
18	0	1	0	0	1	0	**50**	1	1	0	0	1	0
19	0	1	0	0	1	1	**51**	1	1	0	0	1	1
20	0	1	0	1	0	0	**52**	1	1	0	1	0	0
21	0	1	0	1	0	1	**53**	1	1	0	1	0	1
22	0	1	0	1	1	0	**54**	1	1	0	1	1	0
23	0	1	0	1	1	1	**55**	1	1	0	1	1	1
24	0	1	1	0	0	0	**56**	1	1	1	0	0	0
25	0	1	1	0	0	1	**57**	1	1	1	0	0	1
26	0	1	1	0	1	0	**58**	1	1	1	0	1	0
27	0	1	1	0	1	1	**59**	1	1	1	0	1	1
28	0	1	1	1	0	0	**60**	1	1	1	1	0	0
29	0	1	1	1	0	1	**61**	1	1	1	1	0	1
30	0	1	1	1	1	0	**62**	1	1	1	1	1	0
31	0	1	1	1	1	1	**63**	1	1	1	1	1	1

Mit weniger als 6 Beuteln kann Bina ihr Ziel nicht erreichen. Jeden Beutel kann sie beim Bezahlen entweder benützen oder nicht, es gibt also genau zwei Möglichkeiten pro Beutel. Mit nur 5 oder noch weniger Beuteln hätte sie insgesamt also höchstens $2^5 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 32$ Kombinationsmöglichkeiten. Somit könnte sie höchstens 32 verschiedene Gesamtbeträge bezahlen passend bezahlen, was nicht für alle Gesamtbeträge bis 63 Münzen ausreicht.

Dies ist Informatik!

In dieser Aufgabe geht es um *Binärzahlen*. Binärzahlen werden in der Mathematik und Informatik auf verschiedene Weisen untersucht. Mathematik betrachtet vor allem ihre Eigenschaften, wohingegen sich die Informatik mehr mit ihren Anwendungen beschäftigt. Computer nützen die Binärzahlen um ganz unterschiedliche Arten von Informationen darzustellen: Dokumente, Bilder, Stimmen, Videos und Zahlen, sogar die Programme und Apps, die wir alle benützen, sind als Binärzahlen codiert. Die Einheit ist ein *Bit* (*B*inary dig*it* = Binärziffer), das entweder 0 oder 1 sein kann. Ein Bit kann alleine also nur zwei Möglichkeiten unterscheiden. Mit zwei Bits kann man hingegen schon vier Möglichkeiten unterscheiden: 00, 01, 10 und 11. In der vorliegenden Aufgabe benützt Bina 6 Bits (Beutel), um damit $2^6 = 64$ verschiedene Beträge darzustellen.

In Computern werden Bits gewöhnlich in Achtergruppen zusammengefasst; eine solche Achtergruppe nennt man ein Byte. Ein Byte kann $2^8 = 256$ verschiedene Zahlen darstellen, 0 bis 255.

Stichwörter und Webseiten

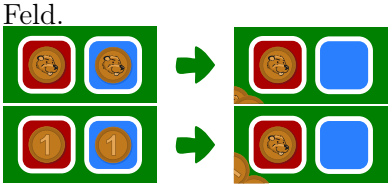
- Binärzahlen: `https://de.wikipedia.org/wiki/Binärcode`
- Datendarstellung
- Logik

33. Las Bebras

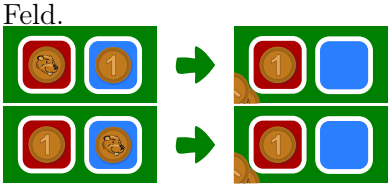
Im Casino «Las Bebras» kann Gloria bei John mit Münzen spielen. Gloria hat 4 Münzen mit Kopf, und auf der Rückseite mit Zahl 1. Gloria wirft die ersten 2 Münzen und legt eine auf das rote, die andere auf das blaue Feld.

John tauscht die beiden Münzen gegen eine neue Münze auf dem roten Feld.

- Sind die beiden Münzen gleich, legt John die neue Münze mit Kopf nach oben aufs rote Feld.

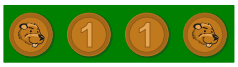


- Sind die Münzen unterschiedlich, legt John die neue Münze mit Zahl 1 nach oben aufs rote Feld.



Gloria wirft nun wieder eine Münze und legt sie auf das blaue Feld, John ersetzt sie wieder nach den Regeln oben und so weiter, bis Gloria alle 4 Münzen ausgespielt hat. Das Spiel ist zu Ende, wenn John die letzte Münze aufs rote Feld legt. Liegt sie mit der Zahl 1 nach oben, gewinnt Gloria!

Gloria spielt die 4 Münzen in der Reihenfolge von links nach rechts aus. Bei welcher Reihenfolge gewinnt Gloria?

A) 

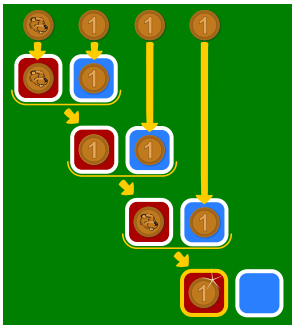
B) 

C) 

D) 

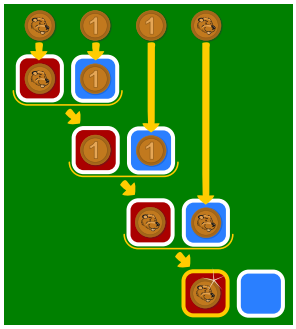
Lösung

Die richtige Antwort ist C). Nur bei Antwort C) liegt am Spielende die Münze mit der Zahl nach oben auf dem roten Feld.

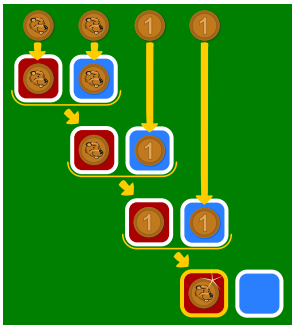


Bei allen anderen Reihenfolgen liegt am Schluss die Münze mit dem Kopf nach oben auf dem roten Feld.

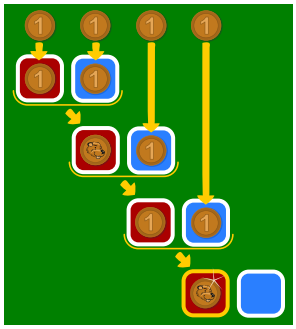
A)



B)



D)



Für jede der 4 Münzen, die Gloria ausspielt gibt es 2 Möglichkeiten sie zu legen (oder) also kann man mit 4 Münzen insgesamt $2^4 = 16$ Reihenfolgen ausspielen. Liegt eine gerade Anzahl Münzen mit Kopf (oder mit Zahl) oben in der Reihe, dann liegt am Spielende die Münze mit dem Kopf nach oben auf dem roten Feld. Liegt eine ungerade Anzahl Münzen mit Kopf (oder mit Zahl) nach oben in der Reihe, dann liegt am Spielende die Münze mit der Zahl nach oben auf dem roten Feld. Eine ungerade Anzahl Münzen mit Kopf (bzw. mit Zahl) nach oben sind also die «Gewinner-Reihenfolgen». Es gibt genau 8 Reihenfolgen mit einer ungeraden Anzahl und 8 Reihenfolgen mit einer geraden Anzahl.

Dies ist Informatik!

Da Computer elektronische Maschinen sind, wird Elektrizität zur Darstellung von Informationen verwendet. Zwei Zustände können einfach mit dem Vorhandensein oder der Abwesenheit eines elektrischen Stroms dargestellt werden. Informatiker stellen diese beiden Zustände normalerweise mit den beiden Zahlen 0 und 1 dar. Dies nennt man binäre Darstellung oder *binäre Repräsentation*. Eine Einheit der Information wird *bit* genannt.

Wir können Operationen an solchen Bits durchführen und sie kombinieren, ebenso wie zwei Münzpositionen (Kopf oder Zahl) in dieser Aufgabe zu einer neuen Münzposition führt.

Eine dieser Operationen wird *logisches XOR* (eXklusives OdeR) genannt. Eine solche Operation wird in dieser Aufgabe durchgeführt. Sie funktioniert so:

0 XOR 0 = 0
0 XOR 1 = 1
1 XOR 0 = 1
1 XOR 1 = 0

Solche Operationen begegnen uns auch im Alltag, z.B. im Treppenhaus: Auf beiden Seiten einer Treppe gibt es zwei Lichtschalter, die dasselbe Licht ein- oder ausschalten. Sind beide Lichtschalter oben, ist das Licht an und sind beide unten, ist das Licht auch an. Ist einer oben und der andere unten, ist das Licht aus.

Ein solches XOR-Gatter ist eine elektronische Umsetzung der XOR-Operation in Computern. Ein XOR-Gatter gibt an seinem Ausgang 1 aus, wenn genau einer seiner beiden Eingänge 1 ist. Sind beide Eingänge gleich, dann gibt der Ausgang 0 aus.

In der Informatik hat die XOR-Operation mehrere Anwendungen, z.B.:

- Es sagt uns, ob zwei Bits gleich oder ungleich sind.
- Es sagt, ob die Anzahl von 1-Bits gerade oder ungerade ist (das XOR einer Folge von Bits ist «wahr» genau dann, wenn eine ungerade Anzahl von Bits «wahr» sind).
- In der Kryptographie wird die XOR-Operation bei der symmetrischen Verschlüsselung mit sogenannten one-time pads verwendet.

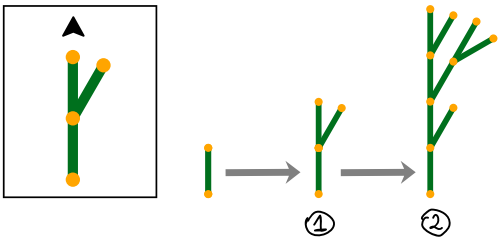
Stichwörter und Webseiten

- Binäre Operation: `https://de.wikipedia.org/wiki/Zweistellige_Verknüpfung`
- XOR: `https://de.wikipedia.org/wiki/Exklusiv-Oder-Gatter`
- Logisches Gatter: `https://de.wikipedia.org/wiki/Logikgatter`
- `https://de.wikipedia.org/wiki/Kontravalenz`

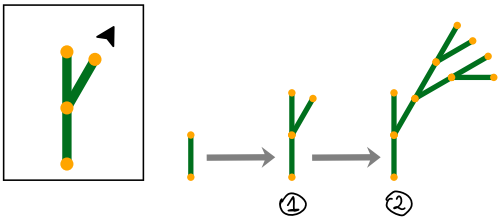
34. Digitale Bäume

Ein digitaler Baum wächst aus folgendem einzelnen Baumstück:  .

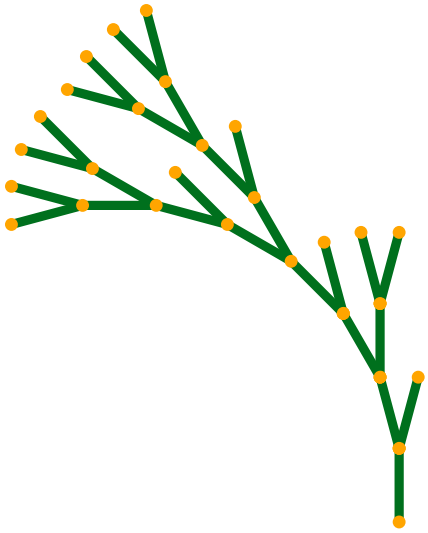
Er wächst schrittweise nach einer vorgegebenen Wachstumsregel.
Die Wachstumsregel gibt an, wie ein Baumstück durch eine Struktur von neuen Baumstücken ersetzt werden kann. In jedem Schritt wird jedes Baumstück auf diese Weise ersetzt. Eine Pfeilspitze gibt an, wo und in welche Richtung die Baumstücke dabei zusammengesetzt werden.



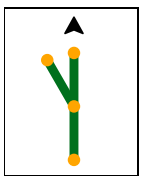
Rechts sind zwei Beispiele für eine Wachstumsregel und jeweils die zugehörigen ersten beiden Wachstumsschritte.

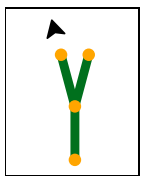


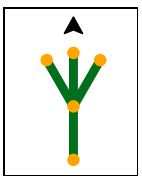
Der folgende digitale Baum ist in 3 Schritten gewachsen:

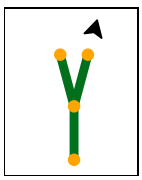


Nach welcher Wachstumsregel ist der digitale Baum gewachsen?

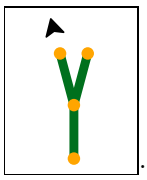
A) 

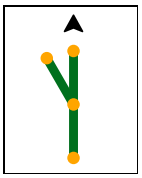
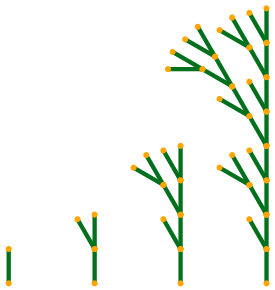
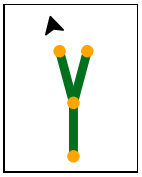
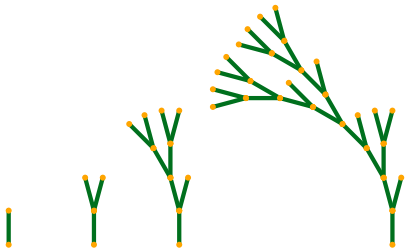
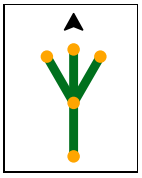
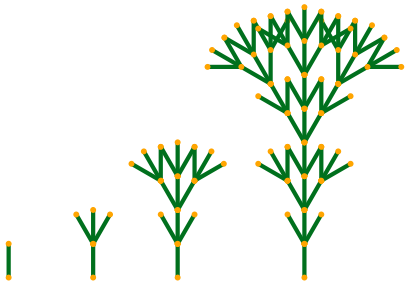
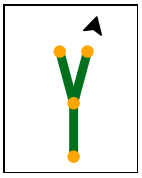
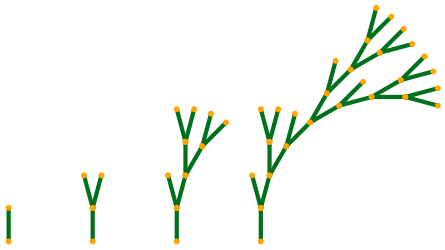
B) 

C) 

D) 

Lösung

Die richtige Antwort ist B)  .

Wachstumsregel	**3 Wachstumsschritte**	**Beschreibung**
		Der Rest des Baumes wird stets am nach oben zeigenden Zweig angefügt, in gerader Richtung. Er bildet dadurch einen geraden Stamm mit Ästen, die nur nach links zeigen.
		Der Rest des Baumes wird stets am linken oberen Zweig angefügt. Der Baum neigt sich deshalb nach links.
		Der Rest des Baumes wird stets in der Mitte angefügt, in gerader Richtung. Durch die beiden Abzweigungen nach links und rechts bildet sich insgesamt eine gleichmässige, symmetrische Struktur.
		Der Rest des Baumes wird stets am rechten oberen Zweig angefügt. Der Baum neigt sich deshalb nach rechts.

Dies ist Informatik!

In der Aufgabe sieht man, wie durch das wiederholte Anwenden einer sehr einfachen Erzeugungsregel komplizierte Figuren entstehen können. Solche Figuren, die aus Teilen bestehen, die der Gesamtfigur ähnlich sind, nennt man auch *Fraktale*. Fraktale werden sehr oft auf Computern eingesetzt, um beispielsweise Landschaften zu erzeugen oder Spezialeffekte für Filme.

In der Biologie werden sogenannte *Lindenmayer-Systeme* (benannt nach dem Biologen Aristid Lindenmayer) verwendet, um das Wachstum von Pflanzen zu simulieren. Dabei entstehen auch Fraktale. In der Aufgabe haben wir vier sehr einfache Beispiele für ein Lindenmayer-System gesehen.

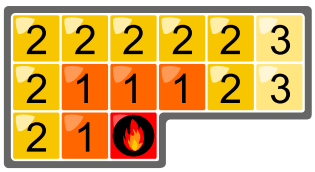
Die Bäume in der Aufgabe entstehen dadurch, dass man eine Regel auf jedes Baumstück anwendet, und auf die dabei entstehenden Baumstücke dann wieder und so weiter. Solche Vorgänge nennt man *rekursiv*. Das Konzept der *Rekursion* ist in der Informatik wichtig. Mit Rekursion ist es möglich, viele komplizierte Dinge sehr einfach zu beschreiben.

Stichwörter und Webseiten

- Fraktal: `https://de.wikipedia.org/wiki/Fraktal`
- Lindenmayer System: `https://de.wikipedia.org/wiki/Lindenmayer-System`, `http://paulbourke.net/fractals/lsys/`
- Rekursion: `https://de.wikipedia.org/wiki/Rekursion`

35. Hotspot-Bodenheizung

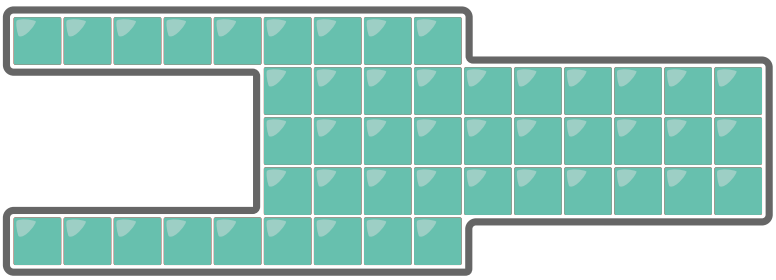
Luis mag es nicht, sich morgens im kalten Badezimmer umzuziehen, deswegen möchte er im neuen Haus eine Bodenheizung einbauen lassen. Der Heizungsmonteur empfiehlt ihm die innovative Hotspot-Bodenheizung: Ein Hotspot wird direkt unter einer Fliese montiert. Schaltet man den Hotspot ein, wird diese Fliese sofort warm.



In einer Minute breitet sich die Wärme auf alle benachbarten Fliesen aus, also auf alle Fliesen, die an einer Kante oder einer Ecke die bereits erwärmte Fliese berühren. Die Zahlen auf jeder Fliese geben an, nach wie vielen Minuten sie warm ist.

Luis will in seinem neuen Badezimmer 4 Hotspots so montieren lassen, dass beim Einschalten alle Fliesen möglichst schnell warm werden.

Unter welchen 4 Fliesen muss der Heizungsmonteur die 4 Hotspots montieren?

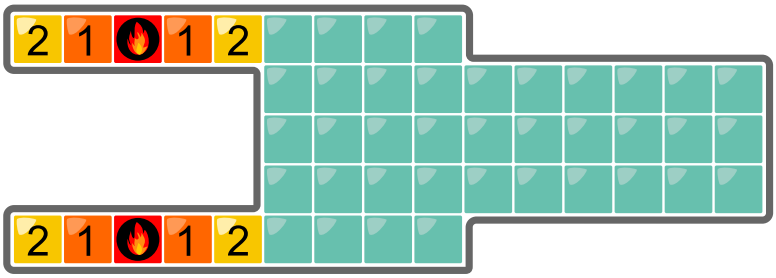


Lösung

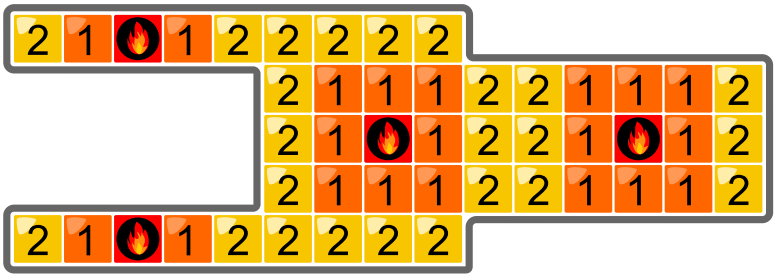
Wenn die 4 Hotspots wie im Bild ganz unten montiert werden, erwärmen sich alle Fliesen des Badezimmers nach dem Einschalten innerhalb 2 Minuten.

Dies ist optimal, denn es ist unmöglich, mit 4 Hotspots alle Fliesen in nur 1 Minute zu erwärmen. Das kann man wie folgt sehen. Jeder Hotspot kann in der ersten Minute höchstens 9 Fliesen erwärmen, nämlich die eigene und bis zu 8 Fliesen rund herum. Somit erwärmen 4 Hotspots zusammen in der ersten Minute höchstens $4 \cdot 9 = 36$ Fliesen. Das Badezimmer hat insgesamt aber 48 Fliesen. Somit reicht 1 Minute sicher nicht. Mit 2 Minuten könnte es hingegen funktionieren, dann könnten theoretisch bis zu $4 \cdot 25 = 100$ Fliesen erwärmt werden.

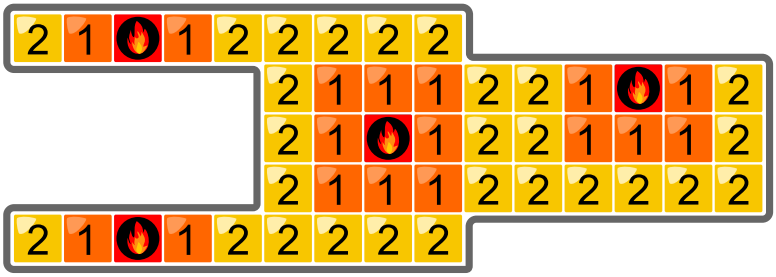
Es bietet sich jetzt an, beim Verteilen der Hotspots mit den beiden Gängen links zu beginnen. Mit je einem Hotspot in der Mitte der beiden Gänge werden gerade alle Fliesen des Ganges innerhalb von 2 Minuten erwärmt:

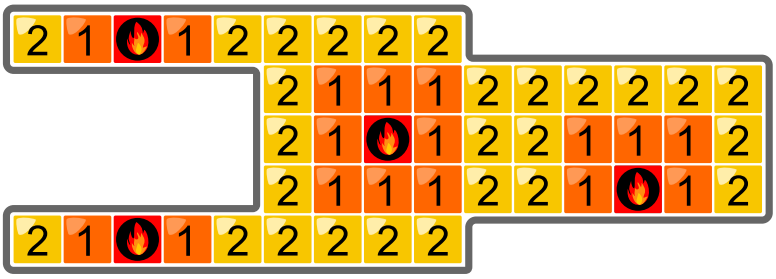


Die anderen zwei Hotspots können wir dann so platzieren:



Die folgenden beiden Platzierungen sind ebenfalls möglich:





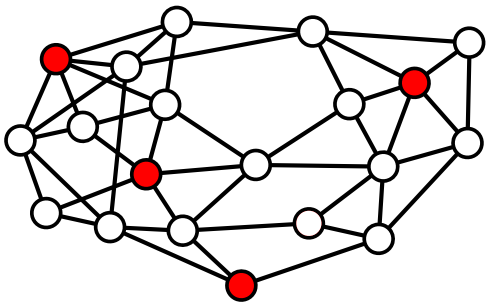
Wenn das Badezimmer eine andere Form hätte, könnten bei gleicher Fläche auch schon 2 Hotspots ausreichen, um das gesamte Badezimmer in 2 Minuten zu erwärmen.

Dies ist Informatik!

Das in dieser Aufgabe gelöste Problem ist mit einem sehr bekannten Optimierungsproblem verwandt: Hier wird eine kleine Menge von *Knoten* in einem *Graphen* gesucht, die man *Dominating Set* nennt.

Eine Dominating Set ist wie folgt definiert: Jeder Knoten des Graphen muss im Dominating Set enthalten sein oder einen Nachbarn haben, der im Dominating Set enthalten ist. Die Fliesen im Badezimmer können als Knoten interpretiert werden. Die Knoten sind mit Kanten verbunden, wenn nach einer Minute die benachbarten Fliese erwärmt wird. Ein Dominating Set des entstehenden Graphen gibt dann die Stellen an, in welchen Hotspots gestellt werden können, um das Badezimmer in 2 Minuten zu erwärmen.

Im Allgemeinen ist es sehr schwer ein minimales Dominating Set zu finden. Für spezielle Graphen gibt es effiziente Algorithmen. Die folgende Zeichnung zeigt ein Beispiel. Wie man sehen kann, ist jeder weisse Knoten Nachbar mindestens eines roten Knotens. Also sind die roten Knoten ein Dominating Set.



Eine typische Anwendung ist die Platzierung von WiFi-Hotspots in einem grossen Gebäude. Die Knoten des Graphen sind die einzelnen Zimmer. Zwei von ihnen sind im Graphen benachbart, wenn beide Zimmer innerhalb der Reichweite eines Hotspots liegen. Zimmer, die ein minimales Dominating Set bilden, sind geeignete Standorte für die Hotspots.

Stichwörter und Webseiten

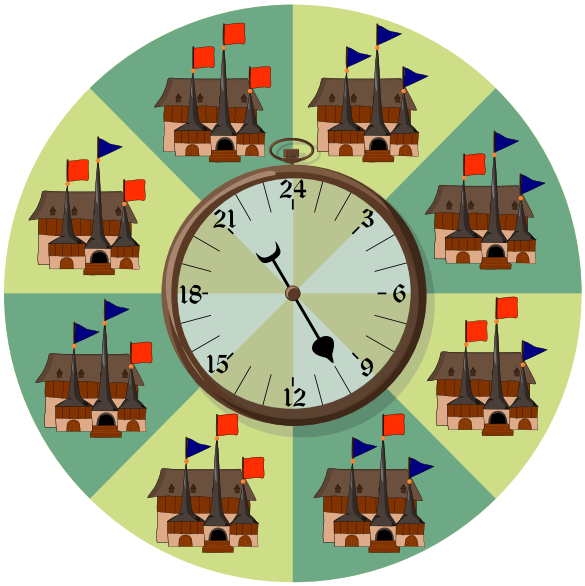
- Dominating set: `https://en.wikipedia.org/wiki/Dominating_set`

36. Bequeme Biber

In einem idyllischen Dorf sind die Biber zeitlich sehr entspannt. Sie teilen den Tag in nur 8 Zeitabschnitte zu je 3 Stunden ein. Der aktuelle Zeitabschnitt wird am Rathaus durch drei Flaggen angezeigt, wie im Bild unten dargestellt. Es werden 2 verschiedene Flaggentypen verwendet, ein rotes Quadrat und ein blaues Dreieck.

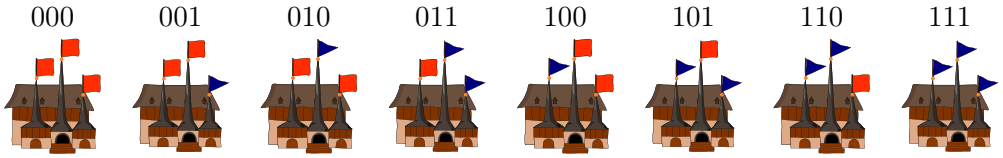
Die momentane Anordnung erfordert bei fast jedem Übergang nur einen Flaggenwechsel. Nur um Mitternacht müssen drei Flaggen gleichzeitig gewechselt werden. Die Biber wünschen sich eine bequeme Anordnung, bei der immer nur eine Flagge gewechselt werden muss.

Finde eine solche bequeme Anordnung.



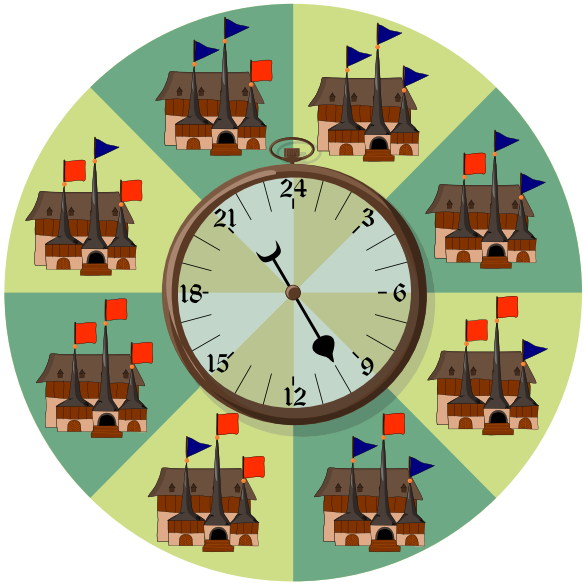
Lösung

Die 8 Muster kann man auch mit dreistelligen Binärzahlen darstellen: 0 steht für ein rotes Quadrat und 1 für ein blaues Dreieck.



Die 8 Muster sind also 000, 001, 010, 011, 100, 101 110, 111. Wir müssen diese Zahlen jetzt so anordnen, dass sich alle benachbarte Zahlen nur in einer Stelle unterscheiden und ebenso die erste und letzte Zahl.

Das kann man durch geschicktes Ausprobieren schaffen. Eine mögliche Lösung ist 111, 011, 001, 101, 100, 000, 010, 110. Hier ist die zugehörige Uhr:



Systematisch findet man eine Lösung mit der folgenden Methode:

Wir betrachten zuerst nur die Zahlen, die mit zwei Nullen beginnen, also 000 und 001. Hier gibt es zwei möglich Anordnungen, beide erfüllen die oben beschriebene Bedingung. Wir wählen 000, 001.

Jetzt schreiben wir dahinter diese beiden Zahlen nochmals in umgekehrter Reihenfolge (also 001, 000), ändern aber die zweite Stelle von 0 zu 1 (also 011, 010). So erhalten wir die Zahlenfolge 000, 001, 011, 010. Sie erfüllt auch wieder die Bedingung.

Diese neue Zahlenfolge schreiben wir jetzt gleich nochmals rückwärts, ändern aber überall die erste Stelle von 0 zu 1. So erhalten wir 000, 001, 011, 010, 110, 111, 101, 100, was wieder die Bedingung erfüllt. Wir haben also die gewünschte Lösung.

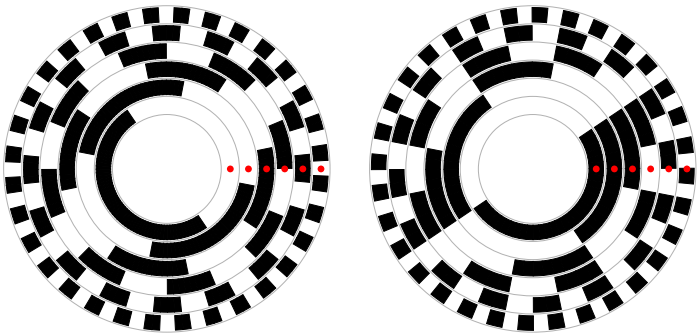
Diese Methode (Spiegeln der bestehenden Zahlenfolge und Ändern der nächsthöheren Stelle von 0 zu 1) kann man beliebig lange fortsetzen, um solche Anordnungen für beliebig viele statt nur drei

Flaggen zu erhalten. Man kann sich überlegen, weshalb diese Methode immer funktioniert und dass immer alle möglichen Muster verwendet werden.

Dies ist Informatik!

Ein solche Anordnung von Binärzahlen heisst *Gray-Code* und hat viele Anwendungen. Dass sich zwischen benachbarten Zahlen jeweils nur ein Bit ändert, kann beispielsweise beim Energiesparen helfen. Mehrere Bits zu ändern, erfordert auf jeden Fall mehr Energie und bei der normalen, aufsteigenden Aufzählung der Binärzahlen ändern sich sehr oft viele Bits gleichzeitig.

Eine berühmte Anwendung des Gray-Code im Ingenieurwesen ist das Messen von Winkeln einer Drehscheibe. Wir zeichnen den Gray-Code so auf die Scheibe wie im Bild links unten gezeigt, weiss für 0 und schwarz für 1. Die roten Punkte sind Sensoren, die auf einer Linie angebracht sind und zwischen schwarz und weiss unterscheiden können. Die Sensoren können also eine Binärzahl (ein Code-Wort) ablesen, die den aktuellen Drehwinkel der Scheibe codiert.



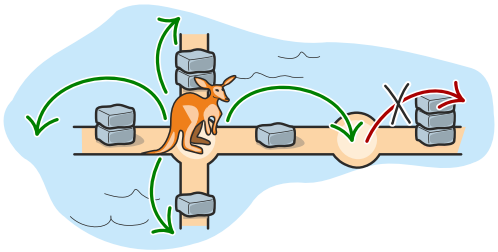
Im linken Bild sehen wir, dass sich der vierte Sensor genau auf der Grenze zwischen Schwarz und Weiss befindet. Der Sensor liest also entweder 001010 oder 001110. Beide Optionen sind akzeptabel, da sich der echte Winkel ja genau in der Mitte befindet. Wenn wir keinen Gray-Code haben, sieht das ganze viel schlechter aus. Betrachten wir den normal Binärcode im rechten Bild. Hier folgen die Code-Wörter 111010 und 111001 aufeinander. Wenn die Sensoren genau dazwischen stehen, können sich die letzten beiden Sensoren beide nicht zwischen Schwarz und Weiss entscheiden, es könnte also auch die Zahl 111011 gelesen werden, die schon einiges weiter weg liegt. 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 kann jeder Sensor willkürlich zwischen 0 and 1 wechseln, was die Winkelmessung völlig unbrauchbar macht.

Stichwörter und Webseiten

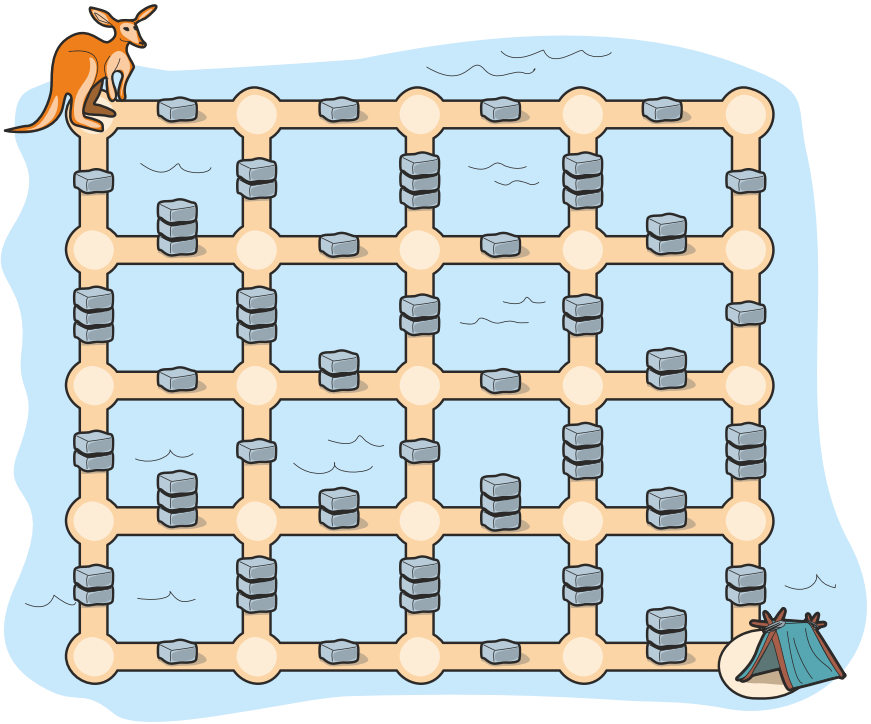
- Gray-Code: `https://de.wikipedia.org/wiki/Gray-Code`

37. Hüpfendes Känguru

Ein Känguru hüpft nach Hause. Es kann nur dem Weg entlang hüpfen und erreicht die nächste Kreuzung in einem grossen Sprung. Bei einer Kreuzung hüpft es entweder nach rechts, links, oben oder unten. Über einen Stapel mit 3 Steinen kann es nicht hüpfen.



Das Känguru möchte auf dem kürzesten Weg nach Hause kommen.

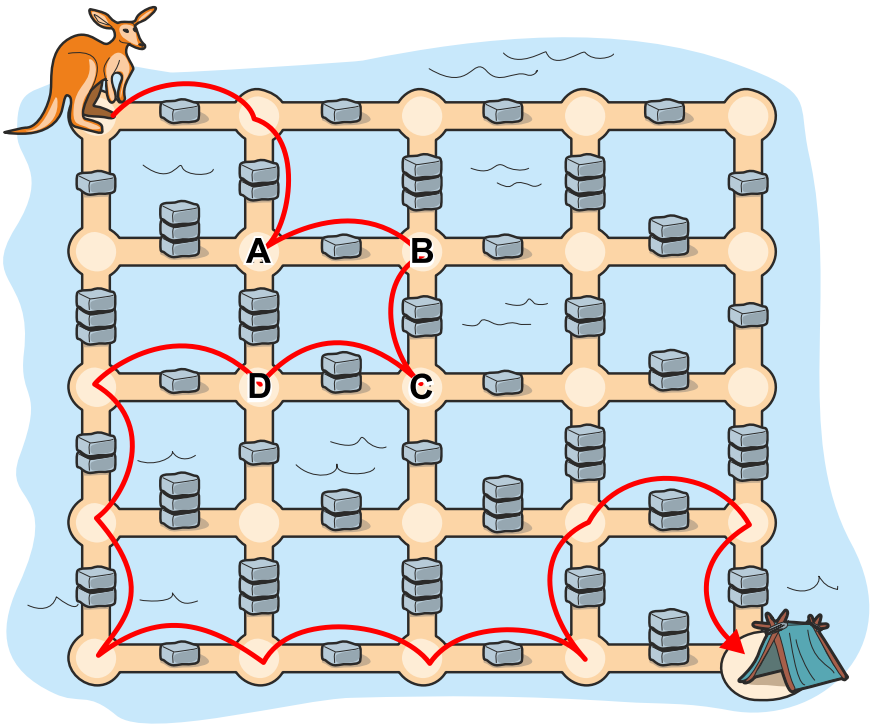


Wie viele Sprünge muss das Känguru machen, um auf dem kürzesten Weg nach Hause zu kommen?

A) 10 Sprünge
B) 11 Sprünge
C) 12 Sprünge
D) 13 Sprünge
E) 14 Sprünge
F) 15 Sprünge
G) 16 Sprünge
H) 17 Sprünge
I) 18 Sprünge
J) 19 Sprünge
K) 20 Sprünge

Lösung

Die richtige Antwort ist E) 14 Sprünge:



Am einfachsten ist es, mit der Suche von hinten zu beginnen. Man sieht schnell, dass es vom Ziel her sehr lange nur einen möglichen Weg gibt, nämlich 9 Sprünge bis zum Punkt D. Jetzt muss man nur noch den kürzesten Weg vom Start zum Punkt Punkt D finden. Mit zwei Schritten gelangt das Känguru zum Punkt A, einem Nachbarpunkt von Punkt D. Direkt kann es jedoch nicht von A nach D hüpfen, da dazwischen ein Stapel mit 3 Steinen liegt. Der kürzeste Umweg von A nach D geht über B und C, dazu braucht es 3 Sprünge. Insgesamt braucht das Känguru so $2 + 3 + 9 = 14$ Sprünge und alle anderen Wege sind länger.

Dies ist Informatik!

Um irgendeinen Weg zu finden, kann man wie folgt vorgehen: Man geht Schritt für Schritt einem beliebigen Weg entlang. Sobald man in eine Sackgasse gelangt, bei der alle Richtungen entweder versperrt sind oder zu einem bereits besuchten Punkt des Weges führen, geht man so lange den abgelaufenen Weg zurück, bis es eine alternative Richtungswahl gibt, und probiert es dann damit weiter.

Dieser Lösungsansatz ist in der Informatik als *backtracking* (Englisch für *Zurückgehen*) bekannt. Er wird in der Informatik vielseitig in verschiedenen Algorithmen eingesetzt. Er kann benutzt werden, um Lösungen von Puzzlespielen, Sudokus oder anderen kombinatorischen Optimierungsproblemen zu finden.

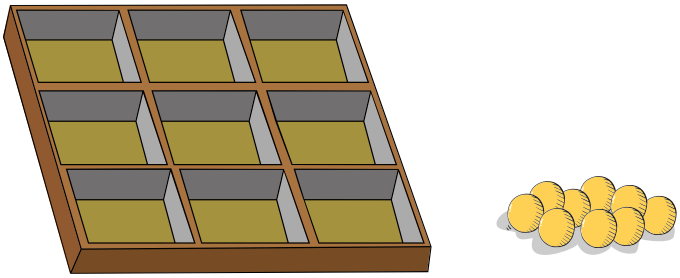
Die Aufgabe zeigt, dass es manchmal effizienter ist, von hinten nach einer Lösung zu suchen. Man spricht dann von einer *Rückwärtssuche*. Im vorliegenden Fall braucht es dadurch weniger Backtracking, weil es am Ende gar keine Optionen mehr gibt. Man kann nicht allgemein sagen, ob eine *Vorwärtssuche* oder *Rückwärtssuche* besser ist, es hängt vom konkreten Problem ab.

Stichwörter und Webseiten

- Backtracking: `https://de.wikipedia.org/wiki/Backtracking`

38. Fächer und Murmeln

Hira hat eine Schachtel, die in 9 Fächer unterteilt ist, und beliebig viele Murmeln:



Hira legt Murmeln in die Fächer der Schachtel. Dabei beachtet sie folgende Regeln:

- In jedes Fach legt sie höchstens eine Murmel.
- In jeder Zeile und jeder Spalte ist die Anzahl Murmeln am Ende gerade.

Wie viele unterschiedliche Muster kann Hira so erzeugen?

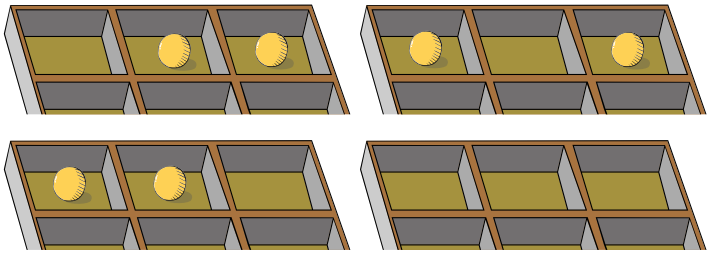
(Die Schachtel kann nicht rotiert werden. Das Muster mit nur einer Murmel links oben ist beispielsweise unterschieden vom Muster mit nur einer Murmel rechts oben.)

A) 12
B) 16
C) 64
D) 512

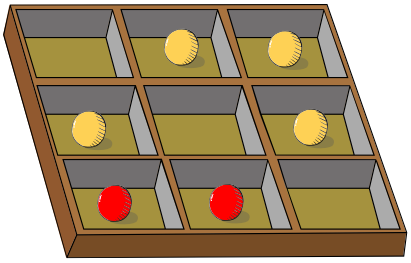
Lösung

Die richtige Antwort ist: B) 16.

Auf wie viele Arten kann Hira die erste Zeile füllen? In der ersten Zeile muss eine gerade Anzahl Murmeln liegen, also 0 oder 2. Daher gibt es 4 Möglichkeiten, die erste Zeile zu füllen:



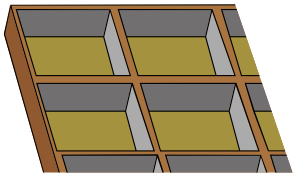
Genauso hat Hira 4 Möglichkeiten, die zweite Zeile zu füllen. Danach kann sie jedoch nichts mehr wählen, denn in den drei Spalten muss ja auch eine gerade Anzahl Murmeln liegen. Liegen in den zwei oberen Zeilen eine ungerade Anzahl Murmeln (also genau eine), so muss Hira in die dritte Zeile dieser Spalte eine Murmel legen, wie dies in den ersten zwei Spalten im folgenden Beispiel der Fall ist (rote Murmeln):



Liegen in den ersten zwei Zeilen einer Spalte eine gerade Anzahl Murmeln (also 0 oder 2), so darf sie keine Kugel in die dritte Zeile dieser Spalte legen, wie dies in der dritten Spalte im Beispiel oben der Fall ist.

Weil die Wahl für die erste Zeile völlig unabhängig von der Wahl für die zweite Zeile ist, hat Hira 4 Möglichkeiten für die erste Zeile und für jede dieser Möglichkeit hat sie wieder 4 Möglichkeiten für die zweite Zeile. Daher hat sie insgesamt $4 \cdot 4 = 16$ Möglichkeiten.

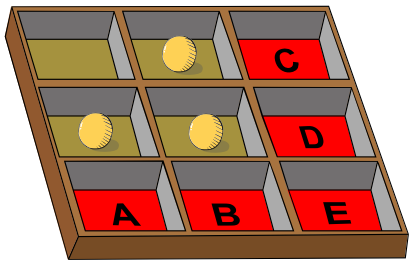
Eine zweite Option für das Zählen der Möglichkeiten ist die folgende: Man betrachtet dazu erst einen 2×2-Teil der Schachtel.



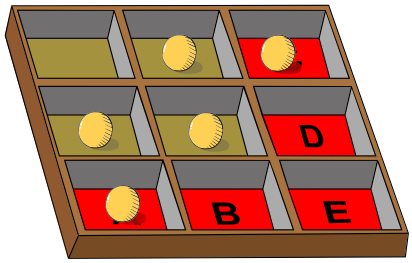
In diesem Teil gibt es 4 Fächer und jede kann entweder eine oder keine Murmel beinhalten. Daher gibt es $2^4 = 16$ verschiedene Möglichkeiten diesen Teil mit Murmeln zu füllen.

Eine wichtige Beobachtung ist die folgende: Nachdem in diesem Teil der Schachtel die Murmeln platziert wurden, hat Hira keine Wahl mehr, die restlichen Fächer zu füllen. Für jedes Fach am rechten Rande oder in der unteren Zeile muss Hira zwingend entweder eine Murmel platzieren oder diese weglassen, damit die Anzahl gerade ist.

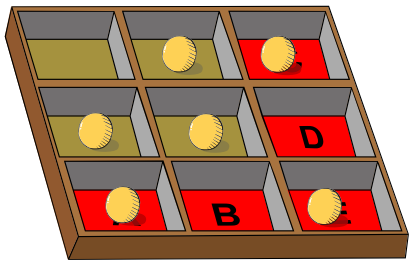
Zum Beispiel könnte Hira den betrachteten 2×2-Teil wie folgt füllen:



Da die erste Spalte nur eine Murmel enthält, muss Hira eine Murmel in das Fach A legen, damit die Anzahl Murmeln in der ersten Spalte gerade ist. In der zweiten Spalte liegt bereits ein gerade Anzahl Murmeln und daher darf Hira keine Murmel in das Fach B legen. Mit ähnlichen Argumenten sieht man, dass das Fach D leer bleiben muss und Hira in C eine Murmel legen muss.



Die Anzahl Murmeln in $A + B$ ist daher genau dann gerade, wenn die Anzahl Murmeln im 2×2-Teil gerade ist. Genau dasselbe gilt für die Summe $C + D$. Falls diese beiden Summen gerade sind, kann und muss das Fach E leer bleiben; falls beide ungerade sind, kann und muss Hira ein Murmel in das Fach E legen.



Dies zeigt, dass Hira Murmeln auf 16 verschieden Arten in die Fächer der Schachtel legen kann.

Dies ist Informatik!

Eine wichtige Aufgabe der Informatik ist es, Daten sicher zu übertragen. Eine Art, die Datenübertragung gegenüber Übertragungsfehlern abzusichern, besteht darin, eine *Paritätsüberprüfung* vorzunehmen.

Ein *Paritätsbit* wird am Ende der Nachricht auf Grund der zu übermittelnden Nachricht berechnet und der Nachricht angefügt. Beim Erhalt der Nachricht, kann das Paritätsbit erneut berechnet werden. Stimmt dieses nicht mit dem übermittelten Paritätsbit überein, so weiss man, dass ein Übertragungsfehler aufgetreten ist.

In dieser Aufgabe dienen die Fächer der letzten Zeile und der letzten Spalte als Paritätsbits. Falls die Zahlen der Murmeln in den Fächern als Nachricht übermittelt wurde, kann der Empfänger die Zeilensummen und Spaltensummen berechnen. Sind diese nicht gerade, so kann der Empfänger Hira melden, dass ein Übertragungsfehler aufgetreten ist.

Eine andere Kompetenz der Informatik ist die Fähigkeit, alle Lösungen mit vorgegebenen Eigenschaften aufzuzählen und somit auch ihre Anzahl zu bestimmen.

Stichwörter und Webseiten

- Paritätsbit: `https://de.wikipedia.org/wiki/Paritätsbit`

A. Aufgabenautoren

Serge Adam
Faisal Al-Sudani
Tony René Andersen
Michael Barot
Wilfried Baumann
Carlo Bellettini
Linda Björk Bergsveinsdóttir
Daniela Bezáková
Maksim Bolonkin
Andrey Brodnik
Lucia Budinská
Špela Cerar
Sarah Chan
Marios O. Choudary
Kris Coolsaet
Valentina Dagienė
Tolmantas Dagys
Christian Datzko
Susanne Datzko
Amirmohammad Djazbi
Hanspeter Erni
Nora A. Escherle
Lidia Feklistova
Fabian Frei
Gerald Futschek
Jens Gallenbacher
Tom Grubb
Yasemin Gulbahar
Husnul Hakim
Mathias Hiron
Juraj Hromkovič
Alisher Ikramov
Thomas Ioannou
Tiberiu Iorgulescu
Takeharu Ishizuka
Mile Jovanov
Ungyeol Jung
Vaidotas Kinčius
Sophie Koh
Dennis Komm
Ritambhra Korpal
Chia-Yi Ku
Regula Lacher
Taina Lehtimäki
Marielle Léonard
Judith Lin
Lynn Liu
Matija Lokar
Vu Van Luan
Hiroki Manabe
Pedro Marcelino
Hamed Mohebbi

Kwangsik Moon
Anna Morpurgo
Xavier Muñoz
Hiroyuki Nagataki
Vania Natali
Rana R. Natawigena
Tom Naughton
Ágnes Erdősné Németh
Andrei Nicolicioiu
Dejan Ozebek
Gabriel Parriaux
Jean-Philippe Pellet
Melinda Phelps
Margot Phillipps
Hannah Piper
Wolfgang Pohl
Prathyush Ponnekanti
Raymond Chandra Putra
Susannah Quidilla
Pedro Ribeiro
Chris Roffey
Peter Rossmanith
Eljakim Schrijvers
Vipul Shah
Fei Shang
Wenpan Sheng
Maiko Shimabuku
Timur Sitdikov
Emil Stankov
Preethi Sudharsha
Maciej M. Sysło
Congyu Tian
Peter Tomcsányi
Monika Tomcsányiová
Meng-ting Tsai
Jiří Vaníček
Troy Vasiga
Fan Wang
Michael Weigend
Jonas Winckler
Michal Winczer
Yang Xing
Khairul Anwar Mohamad Zaki
Binru Zhi

B. Sponsoring: Wettbewerb 2020

http://www.haslerstiftung.ch/
Stiftungszweck der Hasler Stiftung ist die Förderung der Informations- und Kommunikationstechnologie (IKT) zum Wohl und Nutzen des Denk- und Werkplatzes Schweiz. Die Stiftung will aktiv dazu beitragen, dass die Schweiz in Wissenschaft und Technologie auch in Zukunft eine führende Stellung innehat.

http://www.baerli-biber.ch/
Schon in der vierten Generation stellt die Familie Bischofberger ihre Appenzeller Köstlichkeiten her. Und die Devise der Bischofbergers ist dabei stets dieselbe geblieben: «Hausgemacht schmeckt's am besten». Es werden nur hochwertige Rohstoffe verwendet: reiner Bienenhonig und Mandeln allererster Güte. Darum ist der Informatik-Biber ein «echtes Biberli».

http://www.verkehrshaus.ch/

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

i-factory (Verkehrshaus Luzern)
Die i-factory bietet ein anschauliches und interaktives Erproben von vier Grundtechniken der Informatik und ermöglicht damit einen Erstkontakt mit Informatik als Kulturtechnik. Im optischen Zentrum der i-factory stehen Anwendungsbeispiele zur Informatik aus dem Alltag und insbesondere aus der Verkehrswelt in Form von authentischen Bildern, Filmbeiträgen und Computer-Animationen. Diese Beispiele schlagen die Brücke zwischen der spielerischen Auseinandersetzung in der i-factory und der realen Welt.

http://www.ubs.com/
Wealth Management IT and UBS Switzerland IT

http://www.oxocard.ch/
OXOcard: Spielend programmieren lernen
OXON

https://educatec.ch/
educaTEC
Wir sind MINT-Experten. Seit unserer Gründung 2004 verfolgen wir das Ziel, Technik und ingenieurwissenschaftliches Denken in öffentlichen und privaten Schulen der Schweiz zu fördern. In Kombination mit kompetenter Beratung und Unterstützung offerieren wir Lehrkräften innovative Lehrmaterialien von weltweit führenden Herstellern sowie Lernkonzepte für den MINT-Bereich und verwandte Fächer.

http://senarclens.com/
Senarclens Leu & Partner

http://www.abz.inf.ethz.ch/
Ausbildungs- und Beratungszentrum für Informatikunterricht der ETH Zürich.

http://www.hepl.ch/
Haute école pédagogique du canton de Vaud

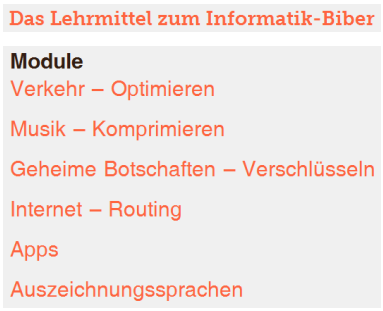
http://www.phlu.ch/
Pädagogische Hochschule Luzern

https://www.fhnw.ch/de/die-fhnw/hochschulen/ph
Pädagogische Hochschule FHNW

http://www.supsi.ch/home/supsi.html
La Scuola universitaria professionale della Svizzera italiana (SUPSI)

https://www.zhdk.ch/
Zürcher Hochschule der Künste

C. Weiterführende Angebote



`http://informatik-biber.ch/einleitung/`
Das Lehrmittel zum Biber-Wettbewerb ist ein vom SVIA, dem schweizerischen Verein für Informatik in der Ausbildung, initiiertes Projekt und hat die Förderung der Informatik in der Sekundarstufe I zum Ziel.
Das Lehrmittel bringt Jugendlichen auf niederschwellige Weise Konzepte der Informatik näher und zeigt dadurch auf, dass die Informatikbranche vielseitige und spannende Berufsperspektiven bietet.
Lehrpersonen der Sekundarstufe I und weiteren interessierten Lehrkräften steht das Lehrmittel als Ressource zur Vor- und Nachbereitung des Wettbewerbs kostenlos zur Verfügung.
Die sechs Unterrichtseinheiten des Lehrmittels wurden seit Juni 2012 von der LerNetz AG in Zusammenarbeit mit dem Fachdidaktiker und Dozenten Dr. Martin Guggisberg der PH FHNW entwickelt. Das Angebot wurde zweisprachig (Deutsch und Französisch) entwickelt.

iLearnIT.ch

I learn it: `http://ilearnit.ch/`
In thematischen Modulen können Kinder und Jugendliche auf dieser Website einen Aspekt der Informatik auf deutsch und französisch selbständig entdecken und damit experimentieren. Derzeit sind sechs Module verfügbar.

Werden Sie SVIA Mitglied – `http://svia-ssie-ssii.ch/svia/mitgliedschaft` und unterstützten Sie damit den Informatik-Biber.
Ordentliches Mitglied des SVIA kann werden, wer an einer schweizerischen Primarschule, Sekundarschule, Mittelschule, Berufsschule, Hochschule oder in der übrigen beruflichen Aus- und Weiterbildung unterrichtet.
Als Kollektivmitglieder können Schulen, Vereine oder andere Organisationen aufgenommen werden.