INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Aufgaben und Lösungen 2020

Schuljahre 7/8

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

Herausgeber:

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

www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikind erausbildung//sociétésuissepourl'infor matiquedansl'enseignement//societàsviz zeraperl'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

Mitarbeit Informatik-Biber 2020 i

Vorwort iii

Inhaltsverzeichnis v

1. 3×3-Tannen-Sudoku 1

2. Nächster Halt, Bahnhof! 5

3. Farbiges Quartier 7

4. Summ, summ, summ... 11

5. Leiterspiel 15

6. Schwere Vergleiche 19

7. Armband 23

8. Haushaltsgeräte 27

9. Maximalausflug 31

10. Bahnnetz 35

11. DNA-Sequenz 39

12. Sturer Fred 41

13. Spinnenauto 45

14. Biberseeland 49

15. Hotspot-Bodenheizung 53

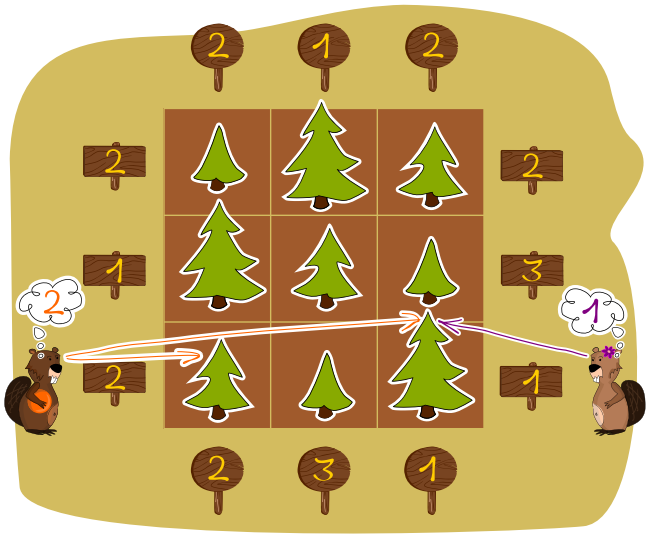
A. Aufgabenautoren 56

B. Sponsoring: Wettbewerb 2020 58

C. Weiterführende Angebote 61

1. 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.

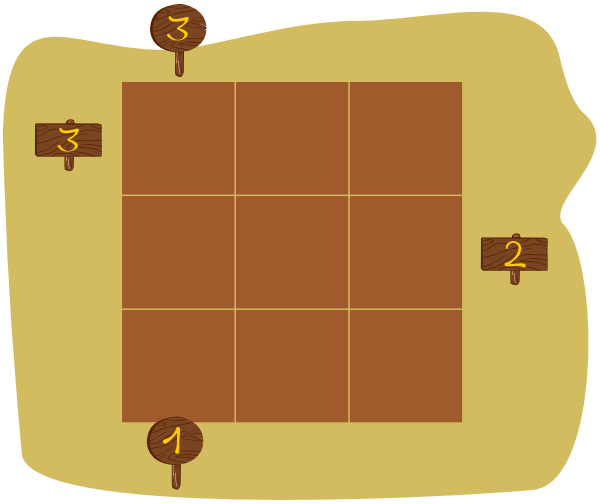
Am Ende jeder Tannenreihe steht auf einem Schild, wie viele Tannen ein Biber von dieser Stelle sehen kann.

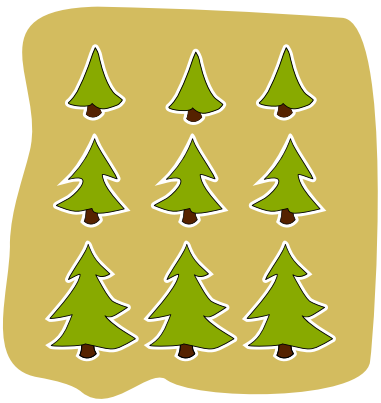
Nun pflanzen die Biber neun Tannen in ein 3×3-Feld, wie im Beispiel rechts.

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.

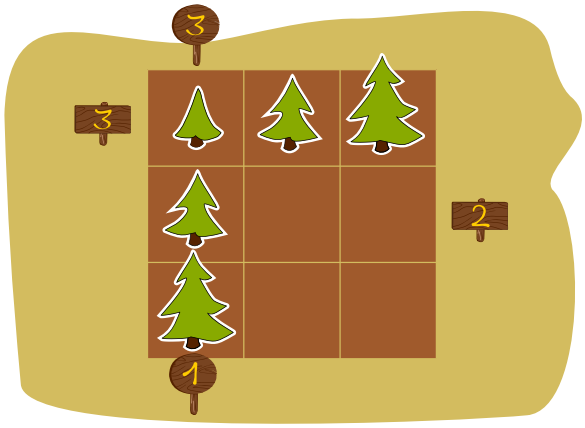
Verteile die Tannen auf die richtigen Felder.





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:

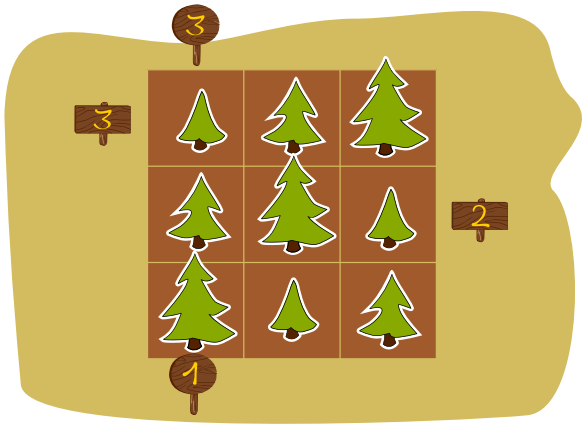


Das Schild rechts mit der 2 verlangt, dass von dort zwei Tannen sichtbar sind, also muss ganz in der Mitte eine Tanne der Höhe 3 sein und diese mittlere Zeile ist somit (), 3 (), 1 ().

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.

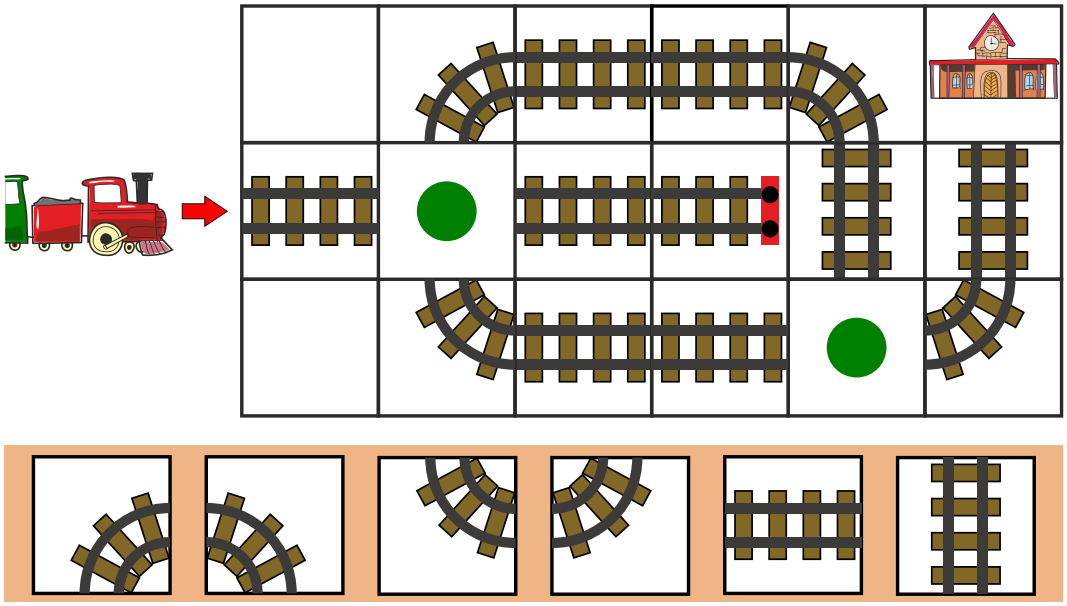
Drittens kann man solche Baumfelder mit Schildern zur Erzeugung von selbst-verifizierenden Codierungen einsetzen. Vorkommende Fehler beim Eintragen oder beim Informationstransport können dann automatisiert erkannt oder sogar korrigiert werden.

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

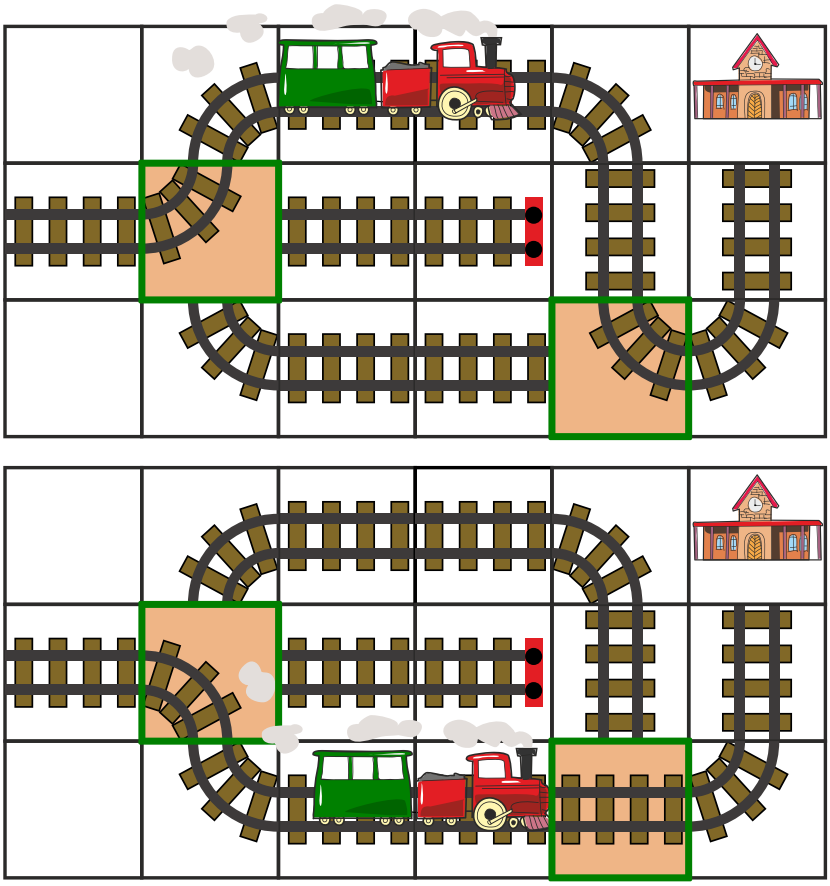
2. Nächster Halt, Bahnhof!

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`

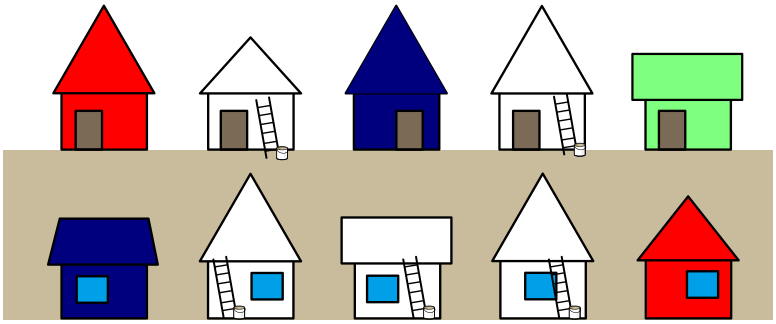
3. Farbiges Quartier

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.

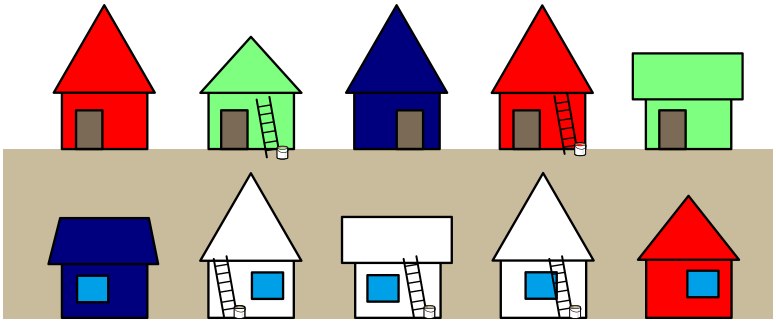
Finde für die Anwohner passende Farben.



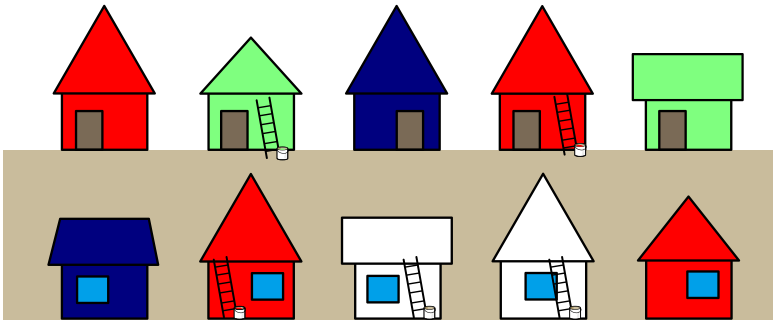
Lösung

Die Farben der Häuser lassen sich am einfachsten schrittweise herausfinden.

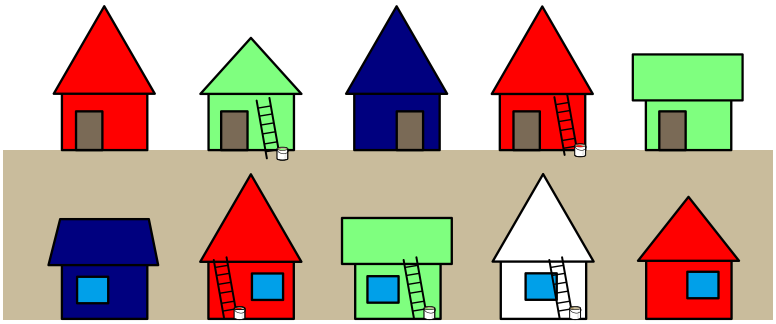
Die beiden weissen Häuser auf der oberen Strassenseite sind jeweils von zwei verschiedenfarbigen Häusern links und rechts umgeben. Deshalb können sie jeweils nur in einer ganz bestimmten Farbe angemalt werden, ohne die Regeln zu verletzen: das weisse Haus links oben in Hellgrün und das rechte weisse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



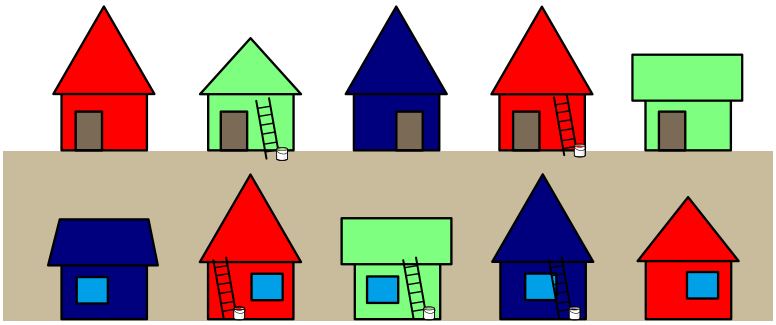
Als Nächstes kann man feststellen, dass das weisse Haus links unten rot angemalt werden muss, weil das Haus direkt links daneben dunkelblau und das Haus direkt gegenüber hellgrün ist:



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.



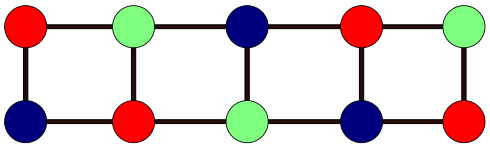
Zuletzt kann man auch für das rechte weisse Haus in der unteren Strassenseite die Farbe bestimmen: Das Haus direkt rechts daneben und das Haus direkt gegenüber sind zwar beide rot, da aber das Haus direkt links daneben jetzt hellgrün ist, bleibt nur noch die Möglichkeit, das Haus dunkelblau anzumalen:



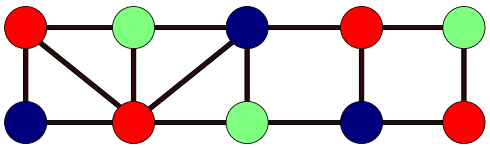
Dies ist Informatik!

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.

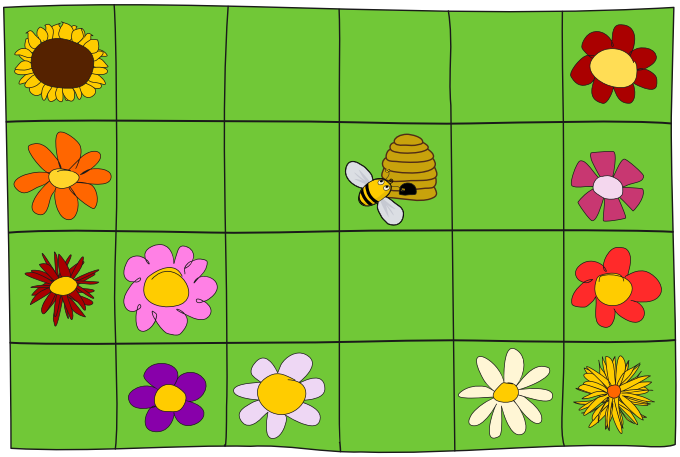
Stichwörter und Webseiten

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

4. Summ, summ, summ...

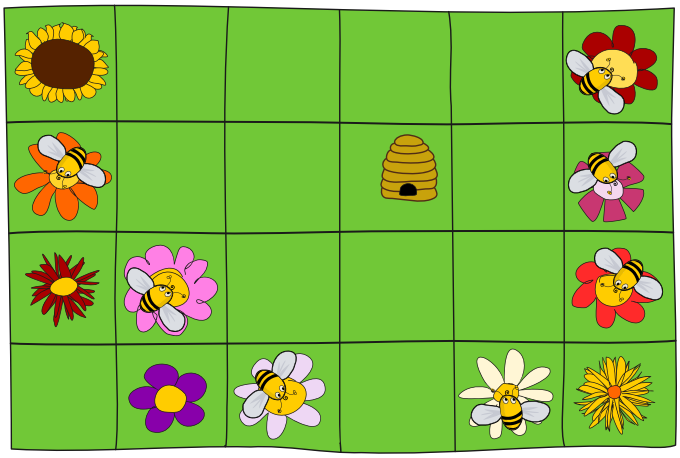
Eine Biene fliegt in 10 Minuten ein Feld nach oben, unten, links oder rechts. Sie fliegt vom Bienenstock aus höchstens 30 Minuten, bevor sie umkehrt.

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.

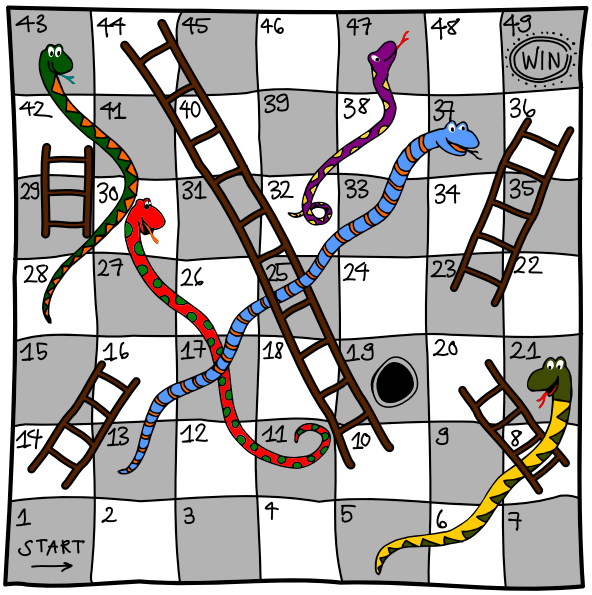
Die übliche Distanzmessung, die zu einem Kreis führt, heisst *Euklidische Distanz*. Die Distanzmessung aus der Aufgabe, bei der man sich nur so horizontal oder vertikal durch Quadrate bewegen darf, heisst *Manhattan-Metrik*. (Der Name kommt von den gitterförmigen Strassennetzen in modernen Städten wie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

5. 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.

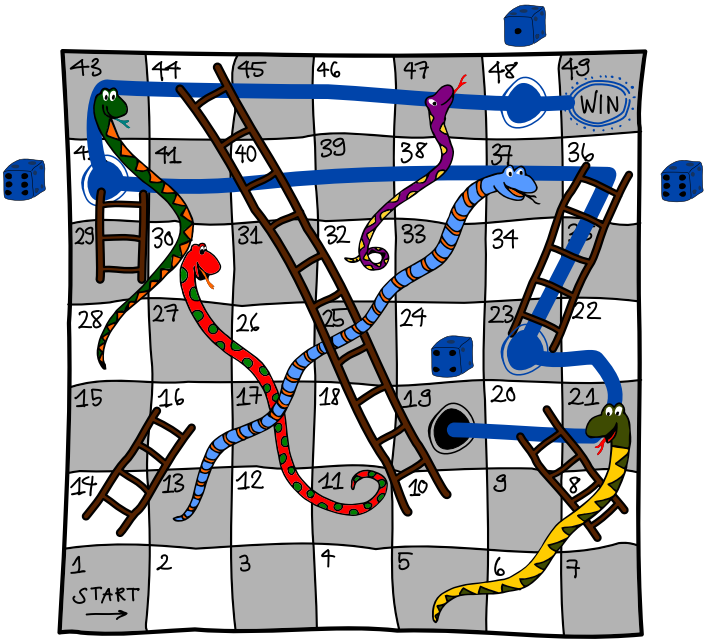
Deine Figur steht auf dem Feld 19. Wie viele Runden brauchst Du mindestens noch, um das Feld 49 zu erreichen?

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

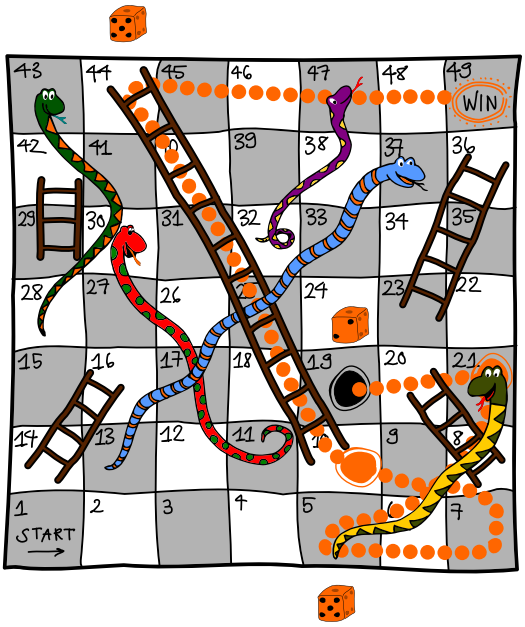
Lösung

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.

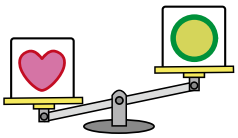
Stichwörter und Webseiten

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

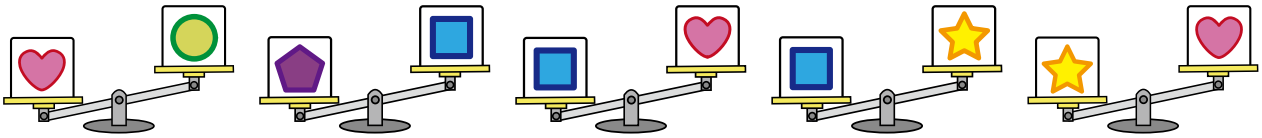
6. Schwere Vergleiche

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:



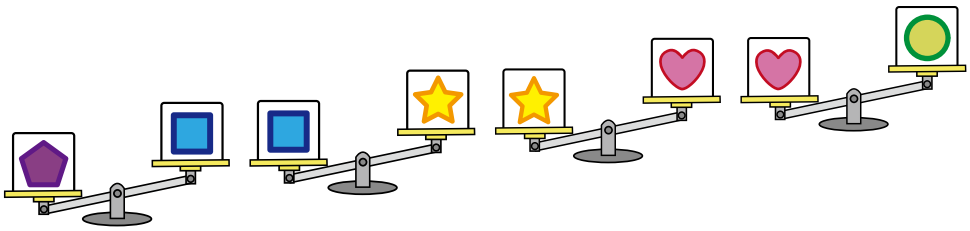
Welche Kiste ist am schwersten?

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.



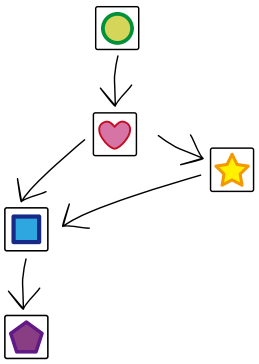
Damit sieht man sofort: Die Kiste mit dem Pentagon ist schwerer als die Kiste mit dem Quadrat . Die Kiste mit dem Quadrat ist schwerer als die Kiste mit dem Stern . Die Kiste mit dem Stern ist schwerer als die Kiste mit dem Herz . Und die Kiste mit dem Herz ist schwerer als die Kiste mit dem Kreis .

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

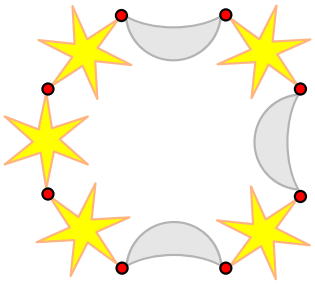
eine *topologische Sortierung*. Eine topologische Sortierung erhält man sehr einfach, indem man immer wieder einen Knoten aus dem Graphen herausnimmt, auf den kein Pfeil zeigt, und die herausgenommenen Knoten in dieser Reihenfolge hintereinander stellt.

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`

7. Armband



Marie möchte das Armband rechts.

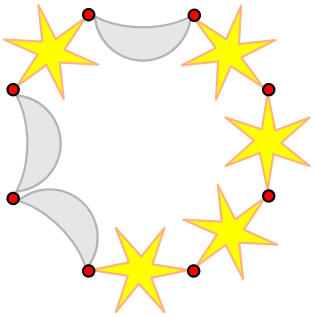
Daher gibt sie Jonas folgende Anweisungen:

- 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.

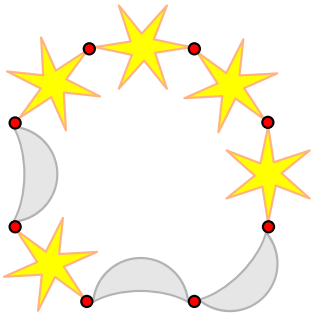
Jonas hat kein Bild des gewünschten Armbands. Es kann sein, dass ein ganz anders aussehendes Armband herauskommt, obwohl sich Jonas exakt an Maries Anweisungen hält.

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

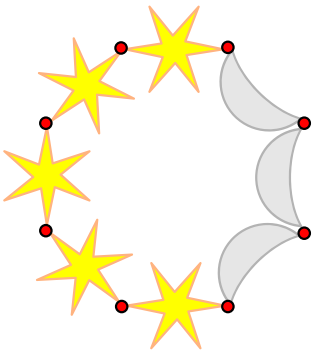
A)



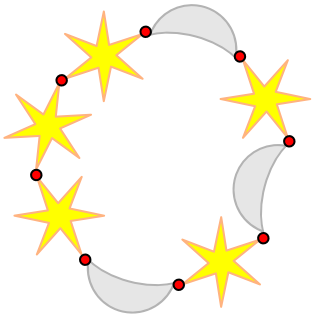
B)



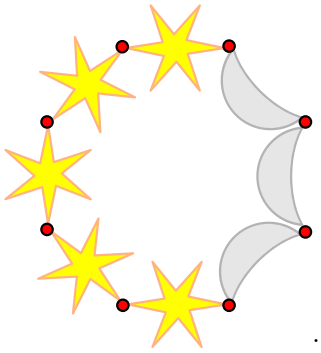
C)



D)

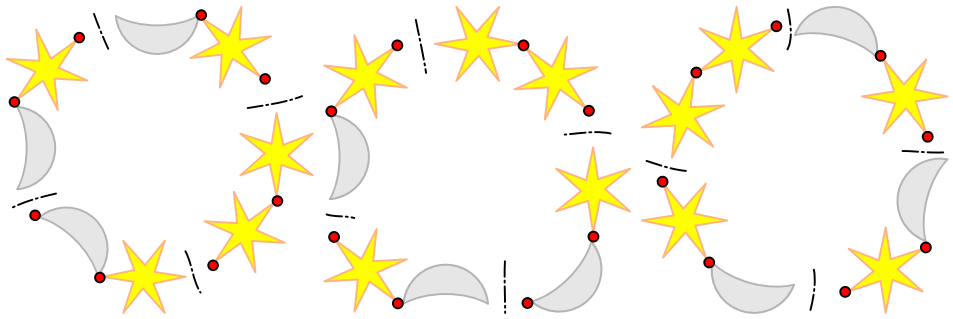


Lösung

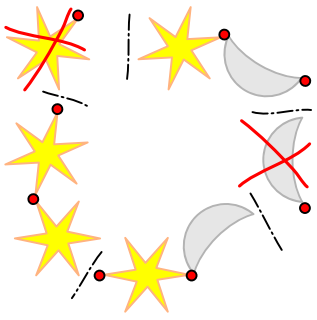
Die richtige Antwort ist C)  .

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:

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

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:

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

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

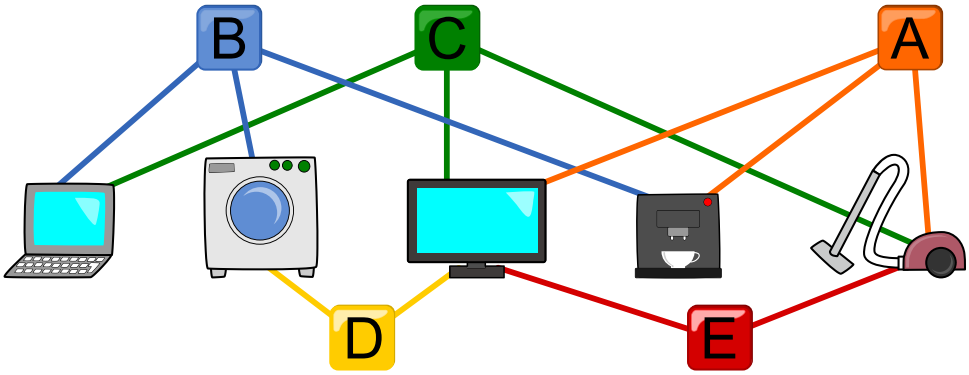
- Formale Grammatik: `https://de.wikipedia.org/wiki/Formale_Grammatik`

8. 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!

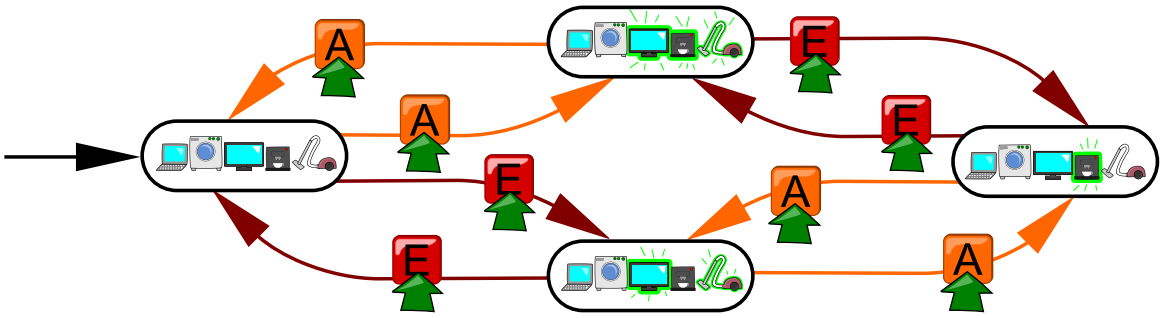
Das System der Geräte und der Knöpfe zum Ein- und Ausschalten kann als sogenannter *endlicher Automat* modelliert werden. Das geht wie folgt.

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.

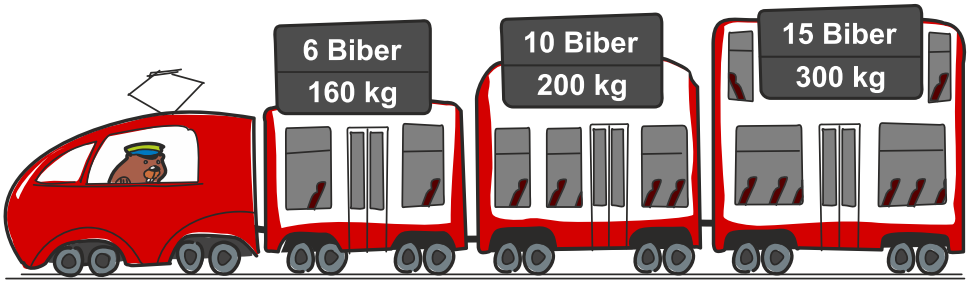
Stichwörter und Webseiten

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

9. 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



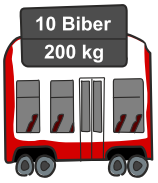
Das Bild zeigt für jeden Waggon, wie viele Biber und wie viel Gepäck in ihm höchstens transportiert werden dürfen. Zudem müssen Familien mit ihrem Gepäck komplett in einem Waggon fahren und dürfen sich nicht aufteilen.

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*.

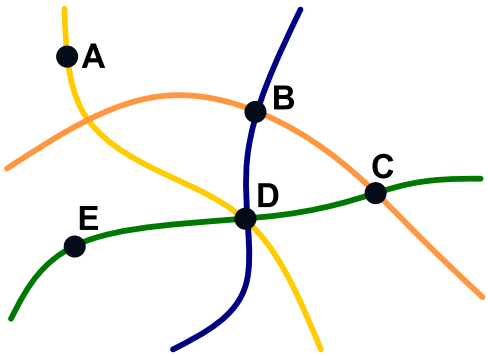
Manchmal lassen sich solche Probleme so reduzieren, dass sie mit Hilfe *Dynamischer Programmierung* gelöst werden können, also indem zunächst mögliche Teillösungen identifiziert werden, die sich dann immer weiter zu einer Gesamtlösung ausbauen lassen. In vielen Fällen gehören die Aufgaben allerdings 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`

10. Bahnnetz

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

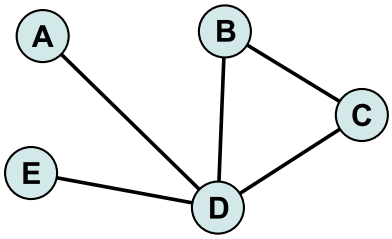


Ein Diagramm soll diese Karte so darstellen, dass:

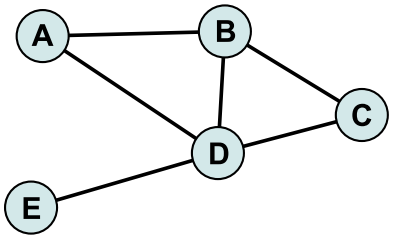
- die Städte durch Kreise dargestellt sind und
- zwei Städte genau dann durch eine Gerade verbunden sind, wenn sie an einem gemeinsamen Bahngleis liegen.

Welches Diagramm stellt die Karte richtig dar?

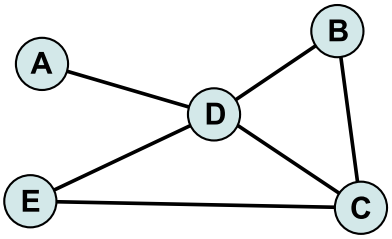
A)



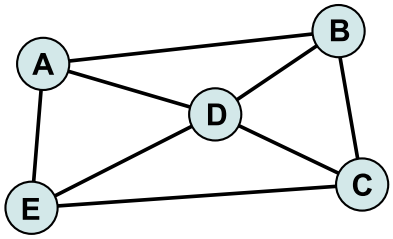
B)



C)

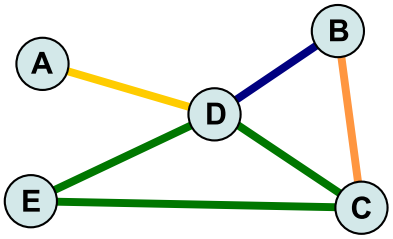


D)



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

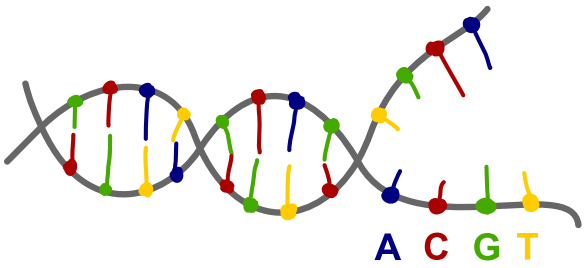
einem Bahngleis von C nach E kommt. Gegenüber der Karte verliert man aber die Information, dass man auf der Fahrt auf diesem Bahngleis von Stadt C nach Stadt E an der Stadt D vorbeikommt.

Stichwörter und Webseiten

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

11. 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.

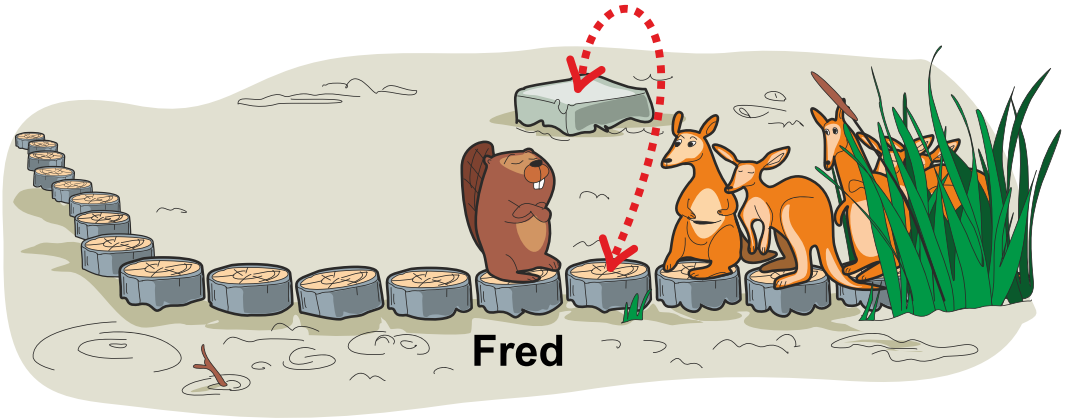
Stichwörter und Webseiten

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

12. 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?

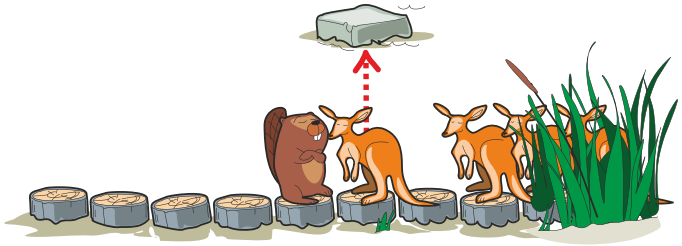
A) Mehr als 10 Kängurus.
B) Genau 10 Kängurus.
C) Genau 6 Kängurus.
D) Genau 4 Kängurus.
E) Weniger als 4 Kängurus.
F) Das kann man nicht genau sagen.

Lösung

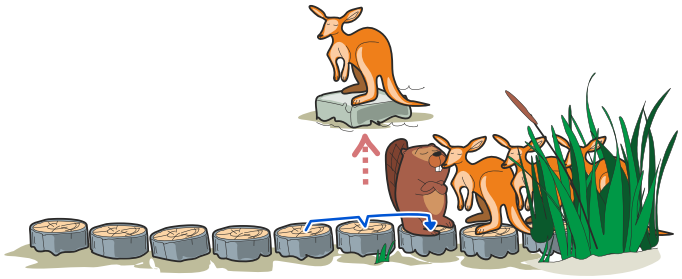
Fred kann maximal genau 6 Kängurus vorbeilassen.

Ein Känguru kommt wie folgt an Fred vorbei:

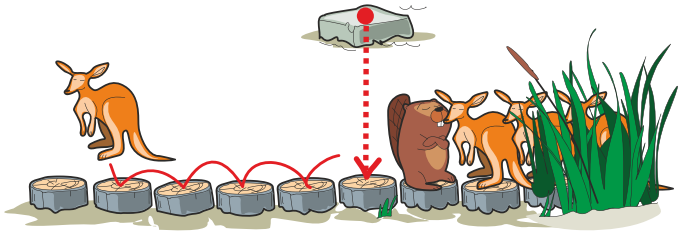
Das Känguru springt auf den Stein.



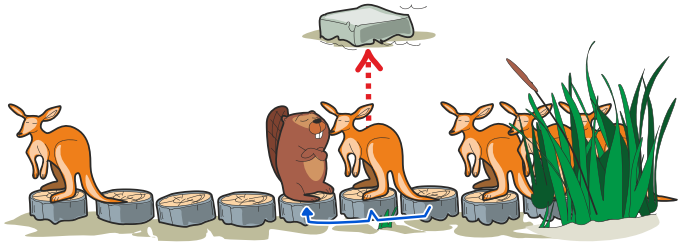
Fred geht zwei Baumstümpfe nach vorne.



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.

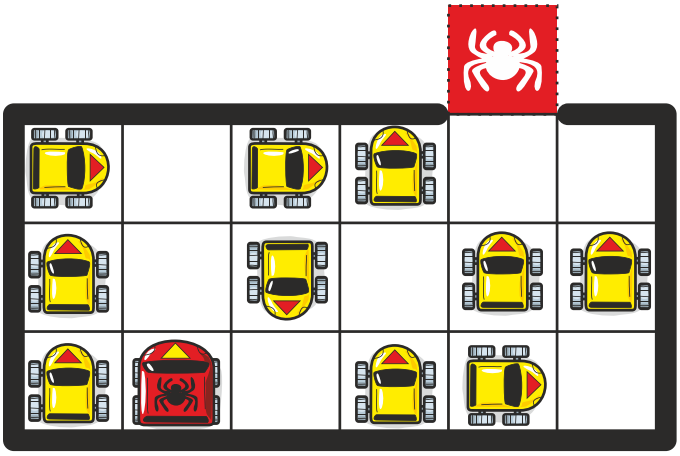
Stichwörter und Webseiten

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

13. 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.

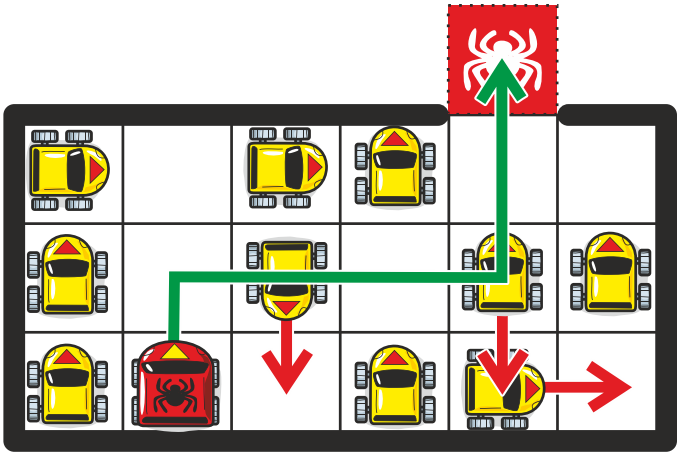
Wie viele Bewegungen der Autos sind insgesamt nötig, um das rote Spinnenauto zum roten Spinnenfeld zu bringen?

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

Lösung

Die richtige Antwort ist: B) 11 Bewegungen.

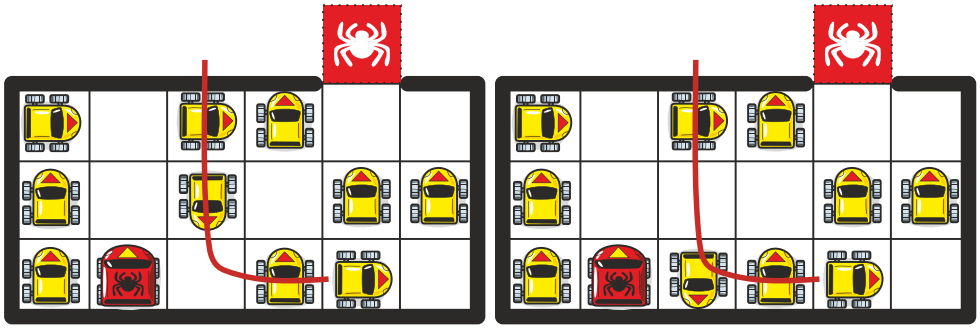
Das Bild zeigt die 11 Bewegungen, um das Spinnenauto zum roten Spinnenfeld zu bringen:



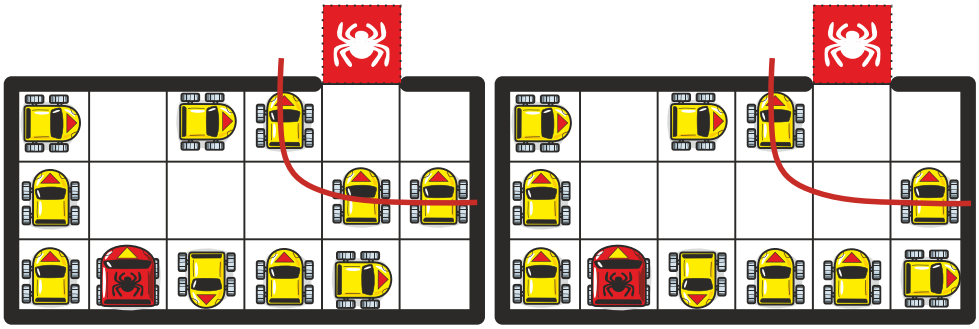
Es muss noch gezeigt werden, dass 11 die minimale Anzahl von Bewegungen ist, die benötigt wird.

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

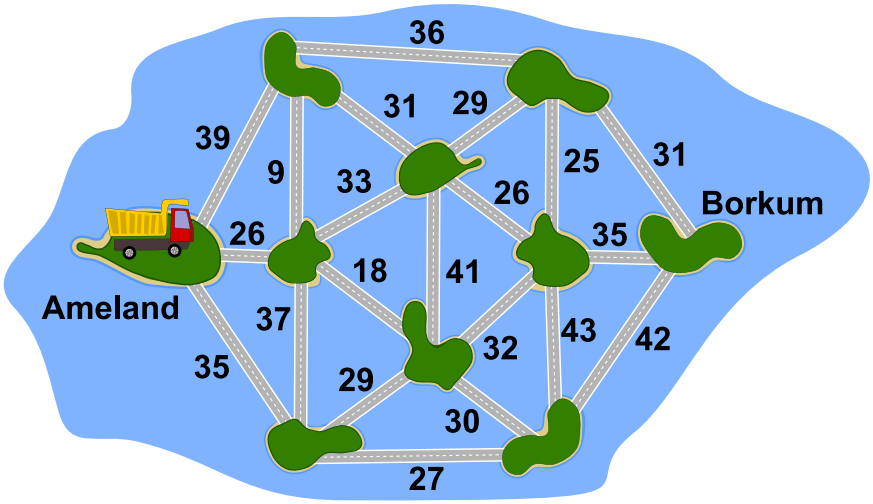
- Brute Force: `https://de.wikipedia.org/wiki/Brute-Force-Methode`
- Branch and Bound: `https://de.wikipedia.org/wiki/Branch-and-Bound`
- Greedy-Algorithmen: `https://de.wikipedia.org/wiki/Greedy-Algorithmus`
- Rush Hour: `https://de.wikipedia.org/wiki/Rush_Hour_(Spiel)`

14. 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.

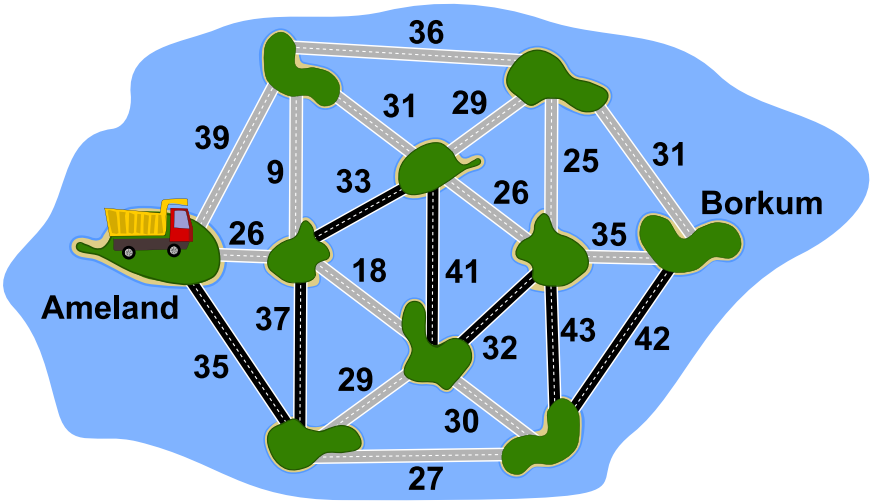
Biber Knuth möchte auf der Insel Borkum einen Strand aufschütten. Mit einer Fahrt will er daher möglichst viel Sand von der Insel Ameland zur Insel Borkum transportieren. Dabei ist ihm die Länge der Fahrtstrecke egal, er will aber über keine Brücke zweimal fahren.

Welchen Weg nach Borkum sollte er mit seinem Lastwagen nehmen?

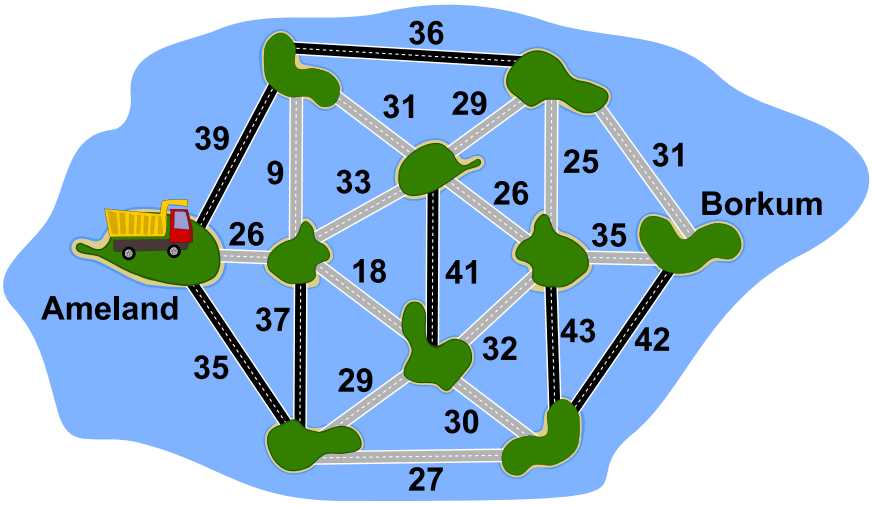


Lösung

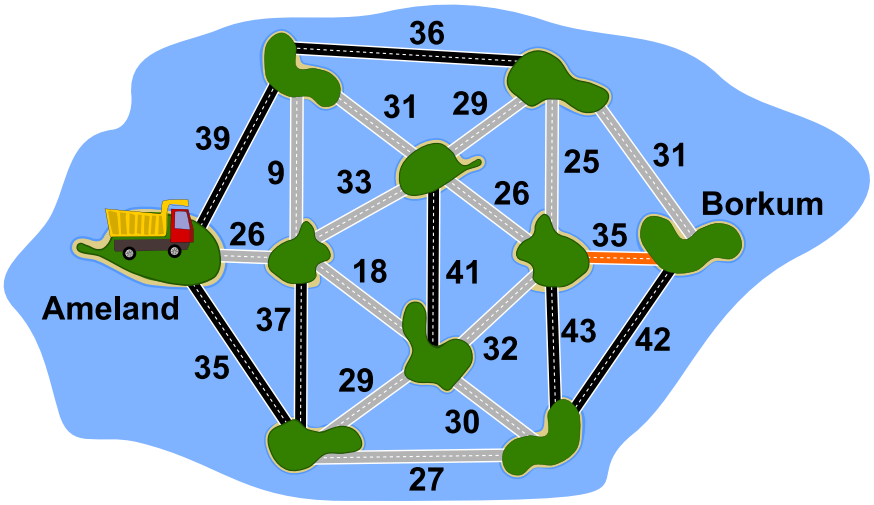
Für die Fahrt beträgt das maximale Gesamtgewicht eines Lastwagens 32 Tonnen. Er nimmt zum Beispiel den folgenden Weg:



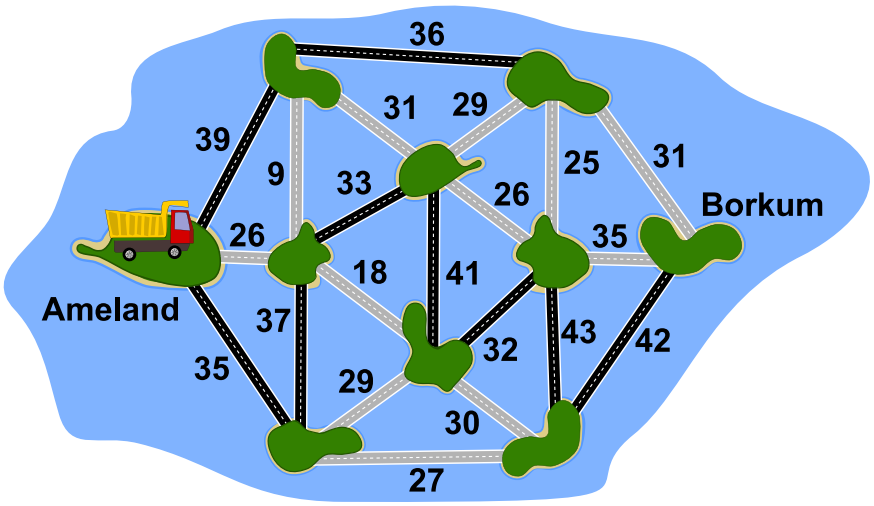
Um diesen zu bestimmen, können wir zum Beispiel virtuell zunächst alle Brücken aus der Karte entnehmen. Alle Brücken werden nach ihrer Belastbarkeit sortiert. Wir fangen mit denjenigen mit der grössten Belastbarkeit an und fügen diese der Karte zu. Danach kommen diejenigen mit der nächstgrössten Belastbarkeit 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.

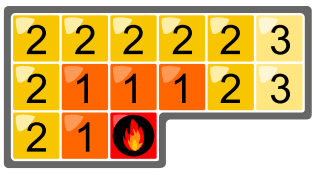
Für eine Lösung kann man wie hier präsentiert das Netzwerk zunächst modellieren, also vereinfachen. In unserem Fall wird durch den *Kruskal-Algorithmus* ein *maximaler Spannbaum* erstellt, in dem der Flaschenhals direkt ersichtlich ist.

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`

15. Hotspot-Bodenheizung

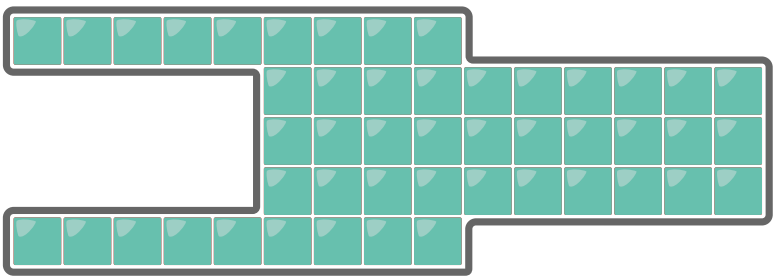
Luis mag es nicht, sich morgens im kalten Badezimmer umzuziehen, deswegen möchte er im neuen Haus eine Bodenheizung einbauen lassen. Der Heizungsmonteur empfiehlt ihm die innovative Hotspot-Bodenheizung: Ein Hotspot wird direkt unter einer Fliese montiert. Schaltet man den Hotspot ein, wird diese Fliese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

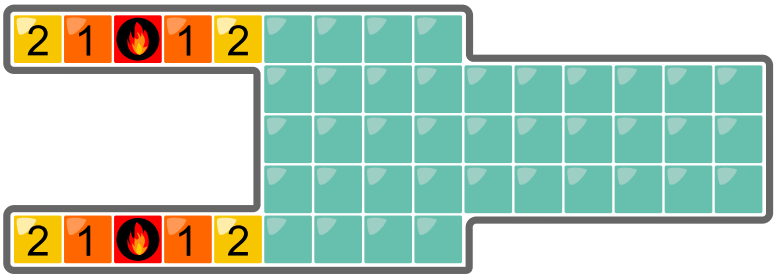


Lösung

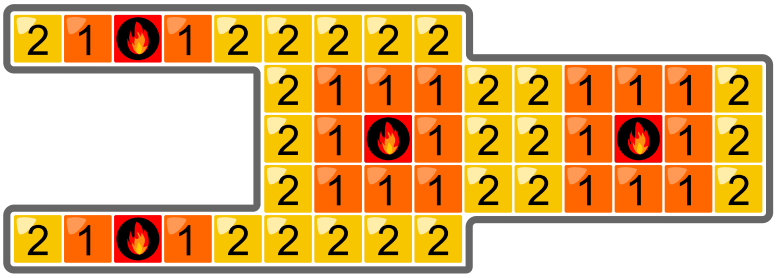
Wenn die 4 Hotspots wie im Bild ganz unten montiert werden, erwärmen sich alle Fliesen des Badezimmers nach dem Einschalten innerhalb 2 Minuten.

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.

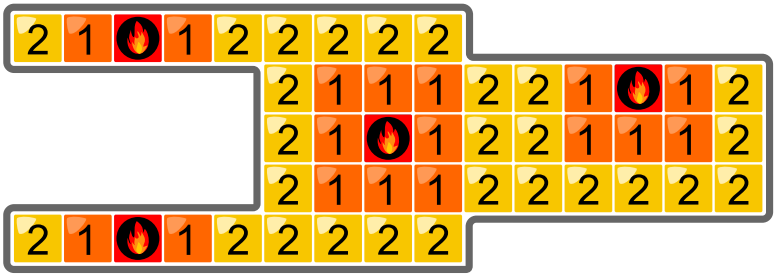
Es bietet sich jetzt an, beim Verteilen der Hotspots mit den beiden Gängen links zu beginnen. Mit je einem Hotspot in der Mitte der beiden Gänge werden gerade alle Fliesen des Ganges innerhalb von 2 Minuten erwärmt:

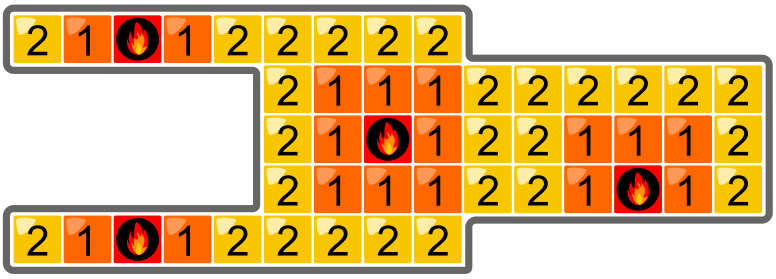


Die anderen zwei Hotspots können wir dann so platzieren:



Die folgenden beiden Platzierungen sind ebenfalls möglich:





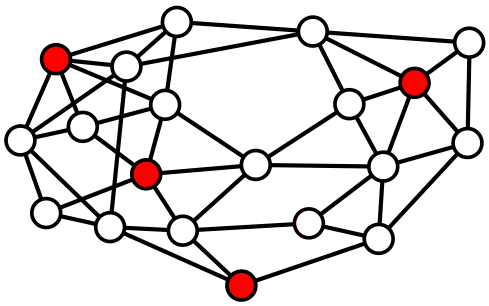
Wenn das Badezimmer eine andere Form hätte, könnten bei gleicher Fläche auch schon 2 Hotspots ausreichen, um das gesamte Badezimmer in 2 Minuten zu erwärmen.

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`

A. Aufgabenautoren

Faisal Al-Sudani
Michael Barot
Carlo Bellettini
Linda Björk Bergsveinsdóttir
Maksim Bolonkin
Andrey Brodnik
Lucia Budinská
Špela Cerar
Sarah Chan
Marios O. Choudary
Kris Coolsaet
Valentina Dagienė
Christian Datzko
Susanne Datzko
Hanspeter Erni
Fabian Frei
Gerald Futschek
Jens Gallenbacher
Yasemin Gulbahar
Mathias Hiron
Juraj Hromkovič
Tiberiu Iorgulescu
Takeharu Ishizuka
Mile Jovanov
Ungyeol Jung
Vaidotas Kinčius
Ritambhra Korpal
Regula Lacher
Marielle Léonard
Hiroki Manabe
Pedro Marcelino
Kwangsik Moon
Anna Morpurgo
Xavier Muñoz
Hiroyuki Nagataki
Vania Natali
Rana R. Natawigena
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

 Maiko Shimabuku

 Timur Sitdikov

 Emil Stankov

 Preethi Sudharsha

 Maciej M. Sysło

 Peter Tomcsányi

 Monika Tomcsányiová

 Troy Vasiga

 Michael Weigend

 Khairul Anwar Mohamad Zaki

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

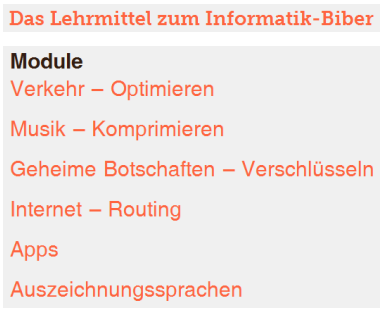
http://www.phlu.ch/
Pädagogische Hochschule Luzern

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.

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.