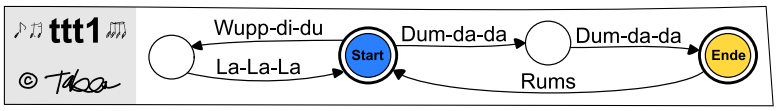
2. Tabeas taktvolle Texte

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» 

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» 

, darf das Lied zu Ende sein, kann aber auch weitergehen.

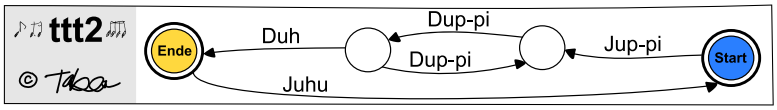
Mögliche Lieder sind zum Beispiel:

«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

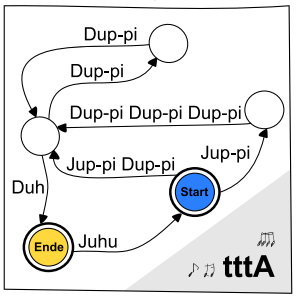
«Dum-da-da Dum-da-da Rums Wupp-di-du La-La-La
Dum-da-da Dum-da-da Rums Wupp-di-du La-La-La
Dum-da-da Dum-da-da Rums Dum-da-da Dum-da-da»

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

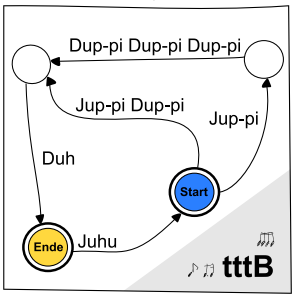


Mit welchem der folgenden Diagramme können genau dieselben Liedtexte wie mit ttt2 produziert werden?

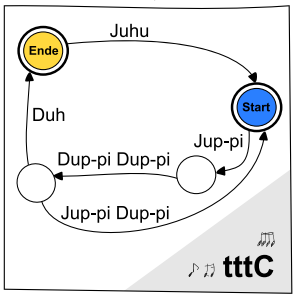
A)



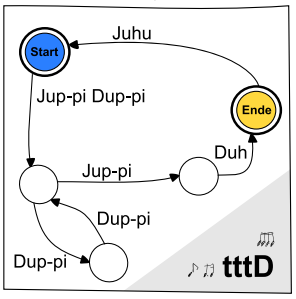
B)



C)

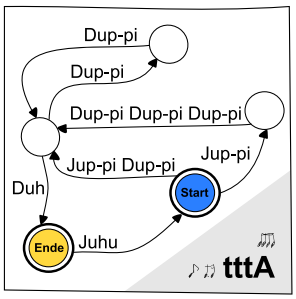


D)

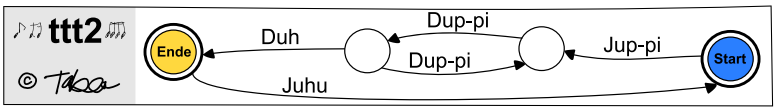


Lösung

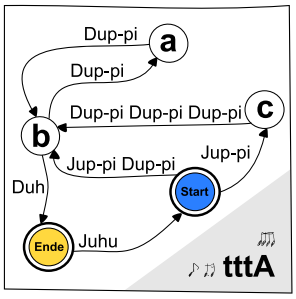
Die korrekte Antwort ist A) Diagramm tttA:



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.

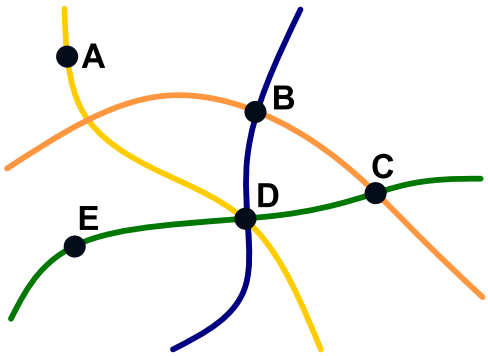
Mustererkennung ist aber auch in vielen anderen Bereichen wichtig, etwa der Verarbeitung natürlicher Sprache.

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`

4. Bahnnetz

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

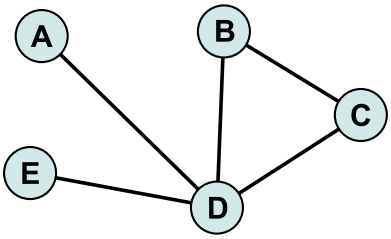


Ein Diagramm soll diese Karte so darstellen, dass:

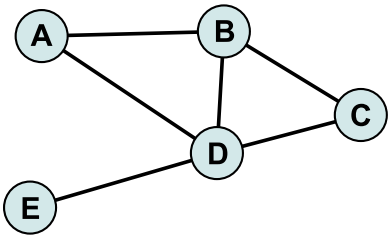
- die Städte durch Kreise dargestellt sind und
- zwei Städte genau dann durch eine Gerade verbunden sind, wenn sie an einem gemeinsamen Bahngleis liegen.

Welches Diagramm stellt die Karte richtig dar?

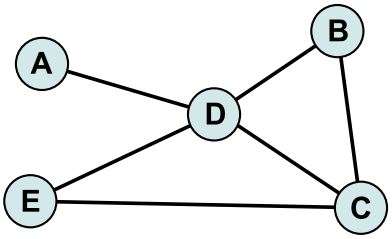
A)



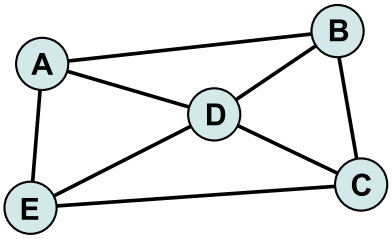
B)



C)

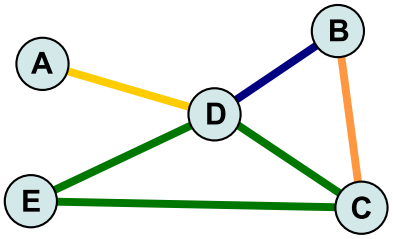


D)



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.

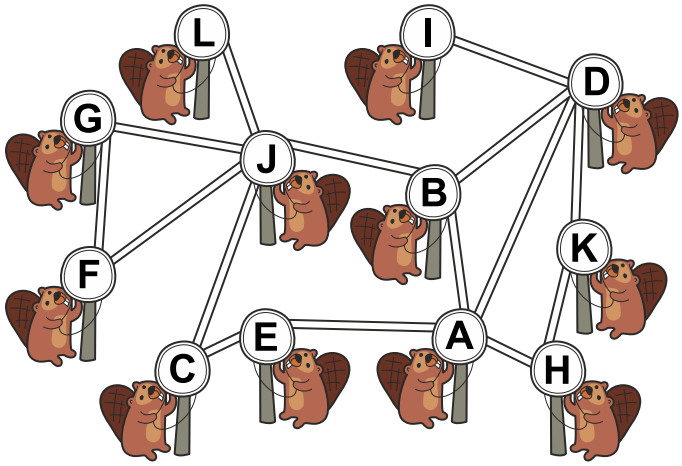
Stichwörter und Webseiten

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

5. 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.

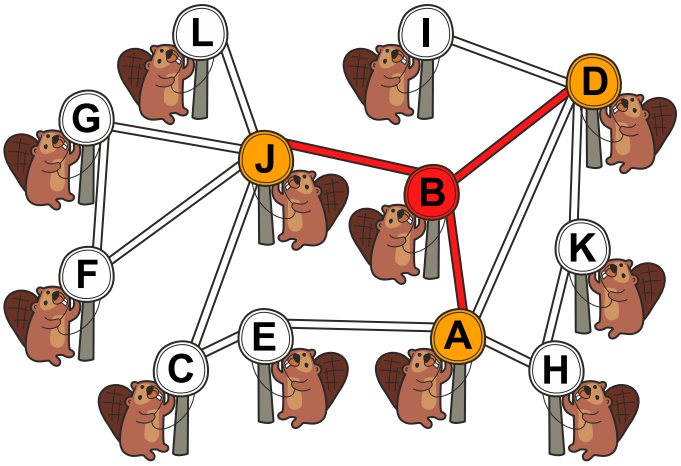
Von welchem Biber aus erreicht eine Nachricht am schnellsten, also in der kleinsten Anzahl Runden, alle anderen Biber?



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.

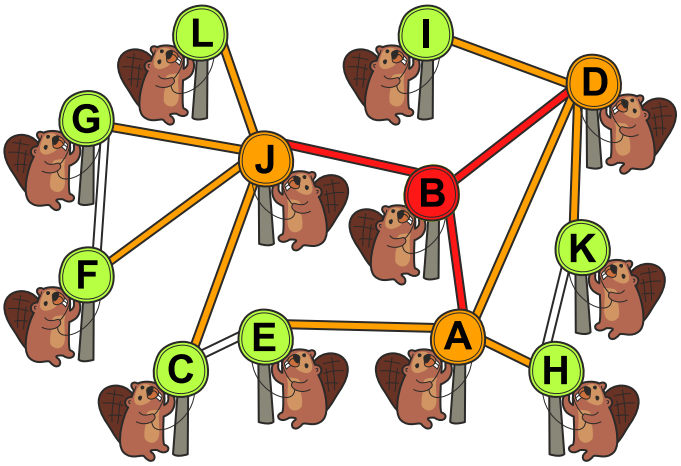


In der zweiten Runde versenden Biber A, D und J die Nachricht jeweils ihren Nachbarn:

- 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.

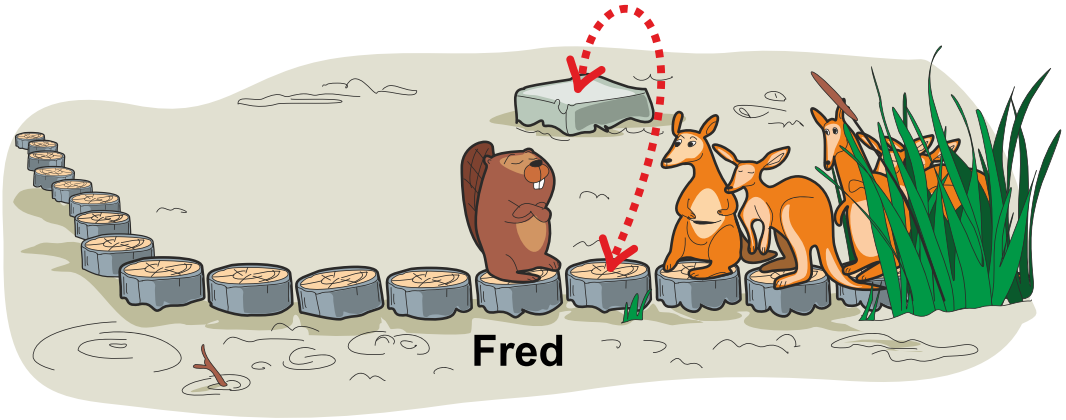
Stichwörter und Webseiten

- 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`

6. 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?

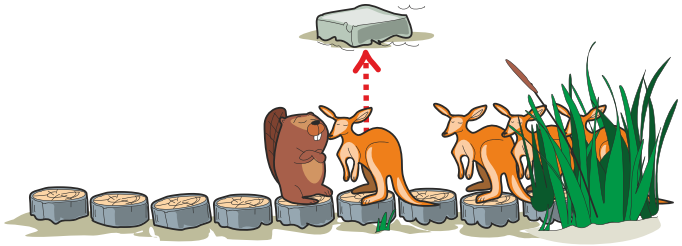
A) Mehr als 10 Kängurus.
B) Genau 10 Kängurus.
C) Genau 6 Kängurus.
D) Genau 4 Kängurus.
E) Weniger als 4 Kängurus.
F) Das kann man nicht genau sagen.

Lösung

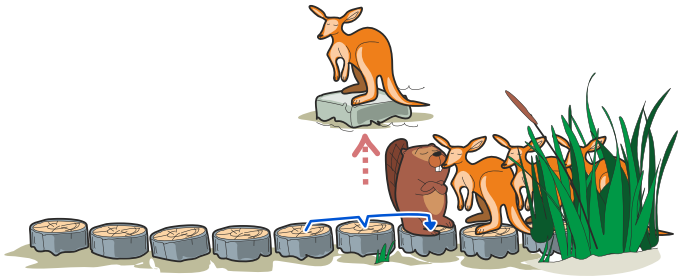
Fred kann maximal genau 6 Kängurus vorbeilassen.

Ein Känguru kommt wie folgt an Fred vorbei:

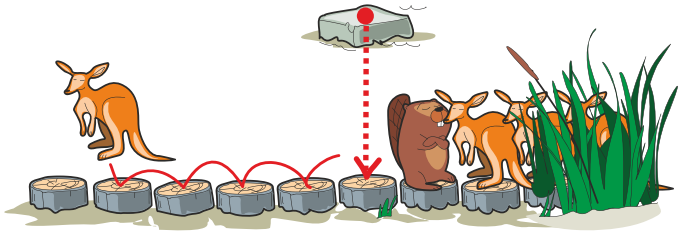
Das Känguru springt auf den Stein.



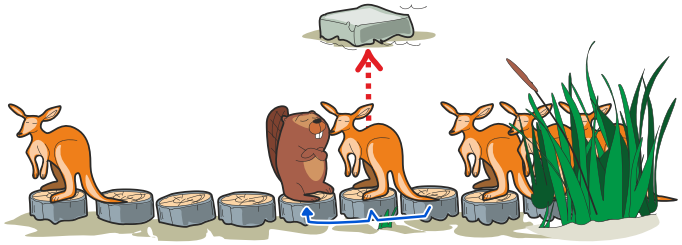
Fred geht zwei Baumstümpfe nach vorne.



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».

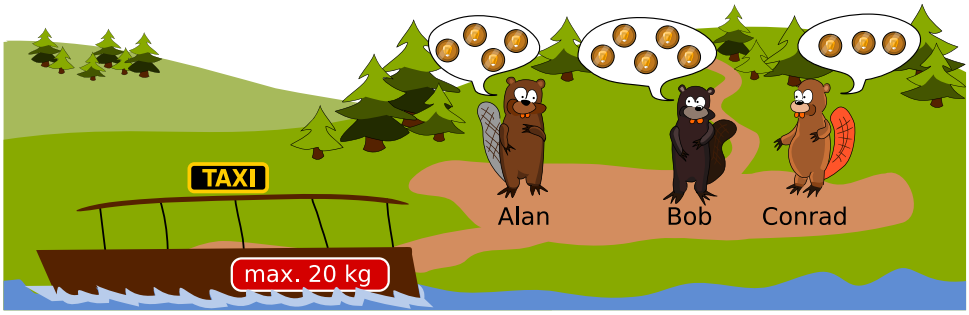
In einer sogenannten *Schleife* können Folgen von Anweisungen wiederholt werden. Auf diese Weise können gleichförmige Aufgaben zuverlässig mehrfach erledigt werden. Dabei ist es meistens von Vorteil, bei jedem Schleifendurchlauf die gleiche Situation herzustellen – die sogenannte *Schleifeninvariante*.

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

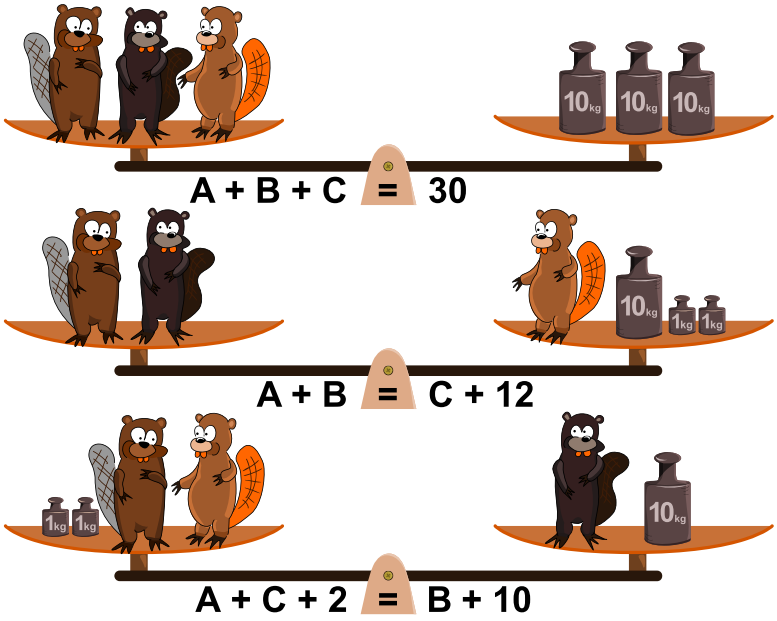
Stichwörter und Webseiten

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

7. 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

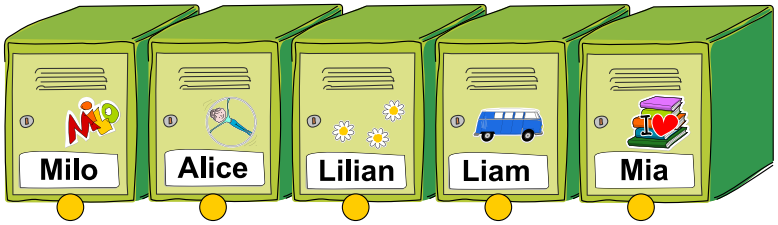
- Ganzzahlige lineare Optimierung: `https://de.wikipedia.org/wiki/Ganzzahlige_lineare_Optimierung`
- Nebenbedingung: `https://de.wikipedia.org/wiki/Nebenbedingung`
- Branch- and Boundverfahren: `https://de.wikipedia.org/wiki/Branch-and-Bound`
- Schnittebenenverfahren: `https://de.wikipedia.org/wiki/Schnittebenenverfahren`

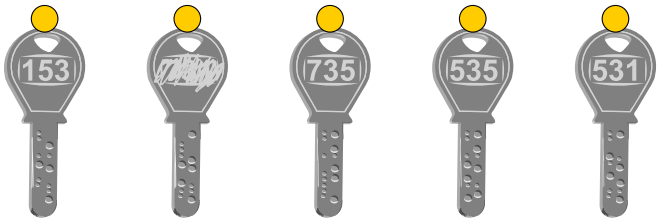
8. Schliessfächer

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».

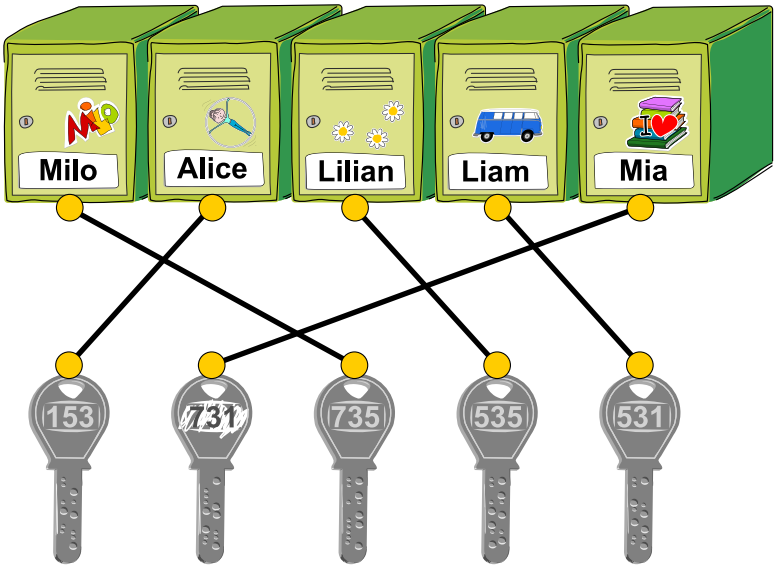
Ordne die Schlüssel den richtigen Schliessfächern zu. Zeichne dazu Linien zwischen den gelben Punkten.





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.

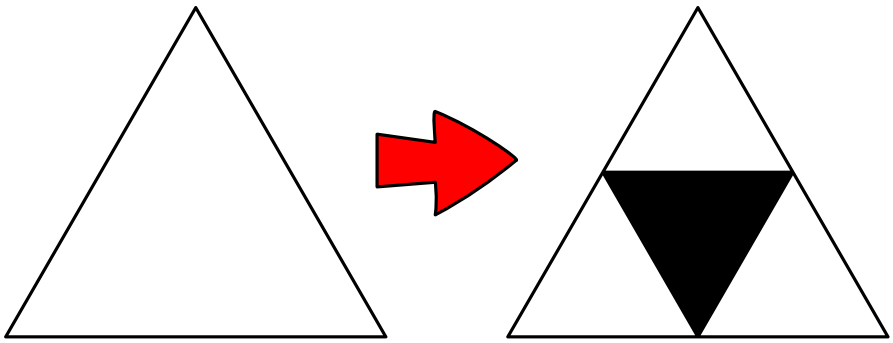
Wenn man nicht nur 10 Ziffern zur Codierung verwendet, sondern ein Symbol für jeden Buchstaben, dann kann man eine solche monoalphabetische Substitution auch als einfache Geheimschrift verwenden. Leider ist die monoalphabetische Verschlüsselungsmethode nicht sehr sicher, weil man die Zuordnung mit ein 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

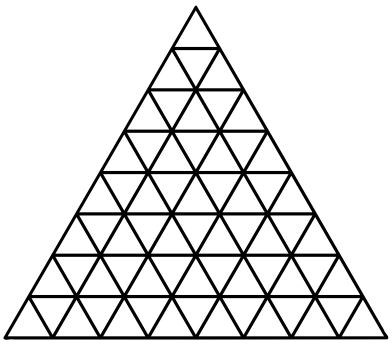
- Codierung, Monoalphabetische Substitution: `https://de.wikipedia.org/wiki/Monoalphabetische_Substitution`
- Kryptographie: `https://de.wikipedia.org/wiki/Kryptographie`

9. 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:

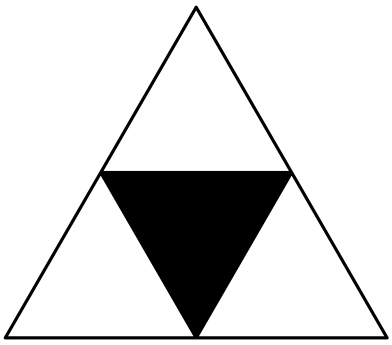


Zeichne die Figur, die nach drei Schritten entsteht. Male dazu die richtigen Teildreiecke an.

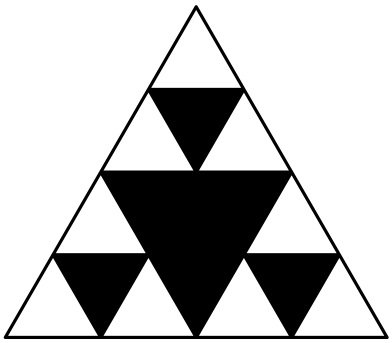


Lösung

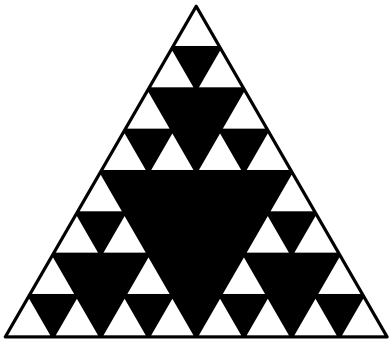
Nach dem ersten Schritt ist das mittlere Teildreieck schwarz und es es bleiben drei weisse Teildreiecke:



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

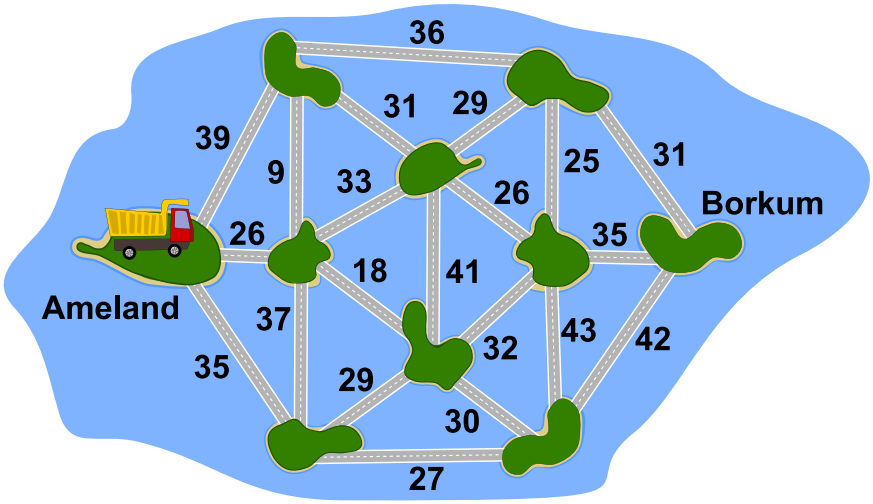
- Sierpiński-Dreieck: `https://de.wikipedia.org/wiki/Sierpinski-Dreieck`
- Rekursion: `https://de.wikipedia.org/wiki/Rekursion`
- Fraktal: `https://de.wikipedia.org/wiki/Fraktal`
- `https://de.wikipedia.org/wiki/Wacław_Sierpiński`
- `https://de.wikipedia.org/wiki/Türme_von_Hanoi#Rekursiver_Algorithmus`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

10. 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.

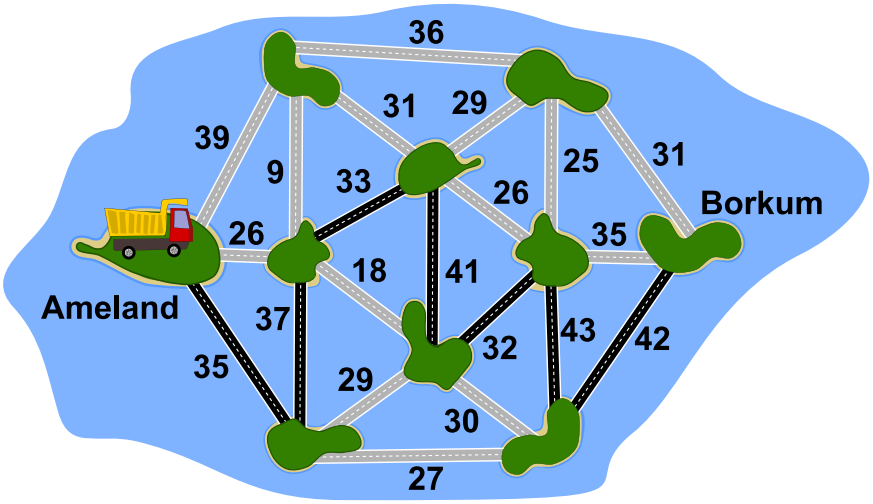
Biber Knuth möchte auf der Insel Borkum einen Strand aufschütten. Mit einer Fahrt will er daher möglichst viel Sand von der Insel Ameland zur Insel Borkum transportieren. Dabei ist ihm die Länge der Fahrtstrecke egal, er will aber über keine Brücke zweimal fahren.

Welchen Weg nach Borkum sollte er mit seinem Lastwagen nehmen?

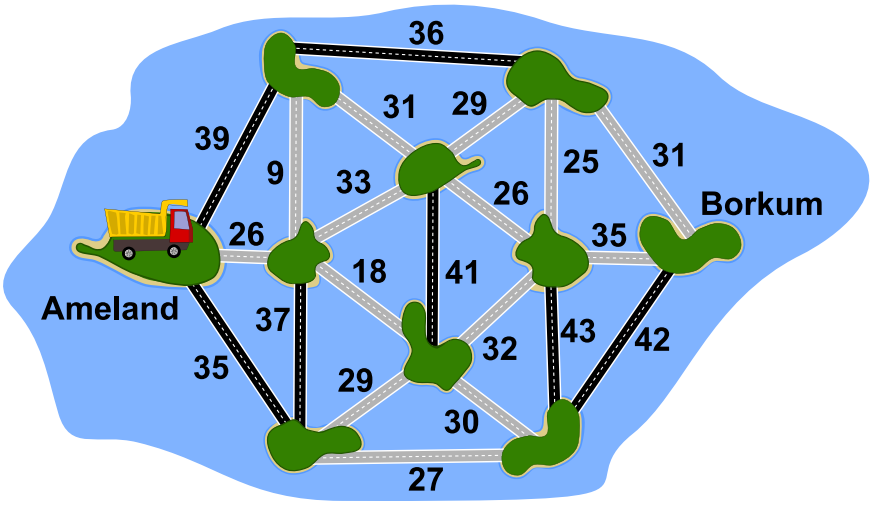


Lösung

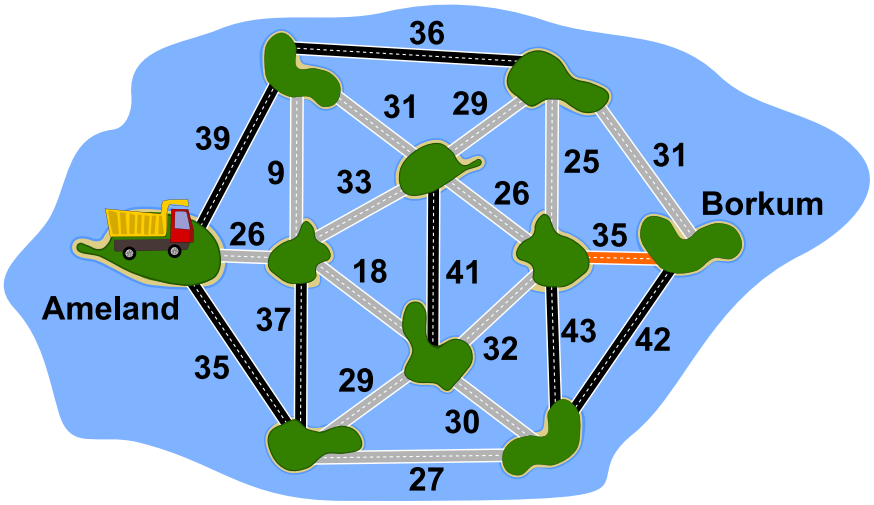
Für die Fahrt beträgt das maximale Gesamtgewicht eines Lastwagens 32 Tonnen. Er nimmt zum Beispiel den folgenden Weg:



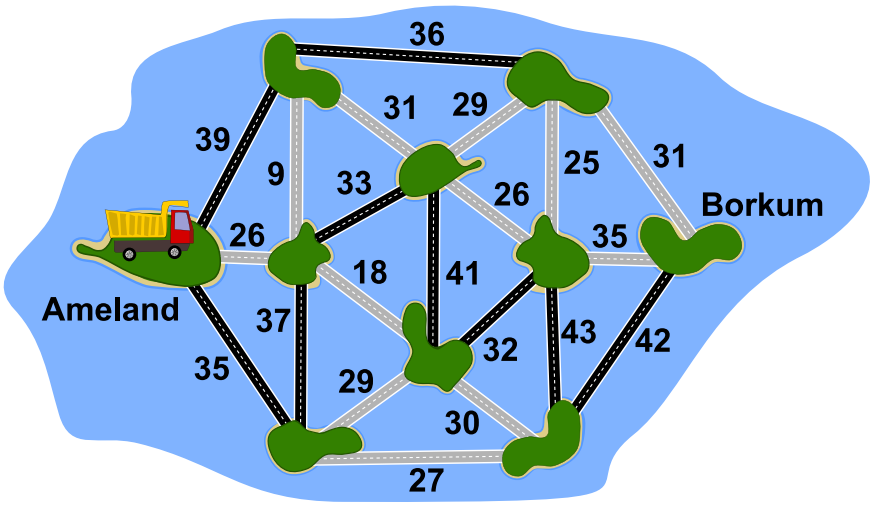
Um diesen zu bestimmen, können wir zum Beispiel virtuell zunächst alle Brücken aus der Karte entnehmen. Alle Brücken werden nach ihrer Belastbarkeit sortiert. Wir fangen mit denjenigen mit der grössten Belastbarkeit an und fügen diese der Karte zu. Danach kommen diejenigen mit der nächstgrössten Belastbarkeit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

11. 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.

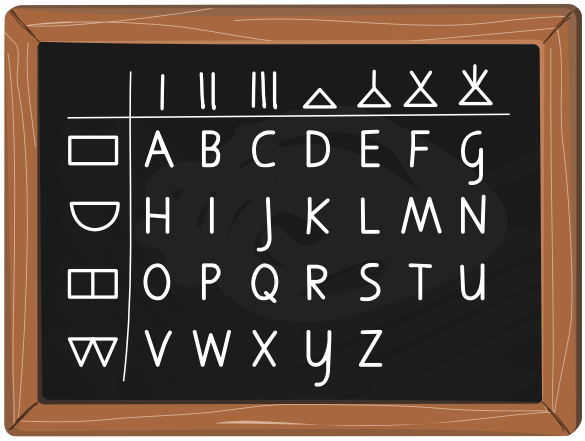
Rekonstruiere den ursprünglichen Text aus dem vorliegenden Geheimtext (dechiffriere den Geheimtext). Welcher der 4 Lösungsvorschläge stimmt?

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.

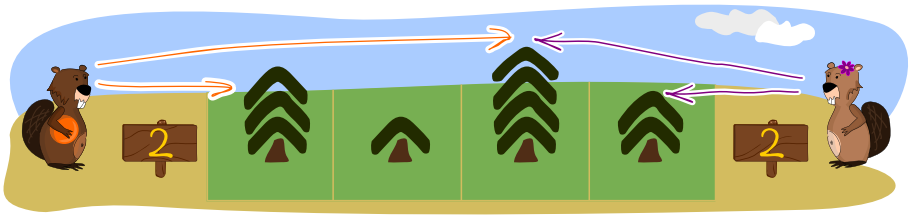
Stichwörter und Webseiten

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

12. 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:

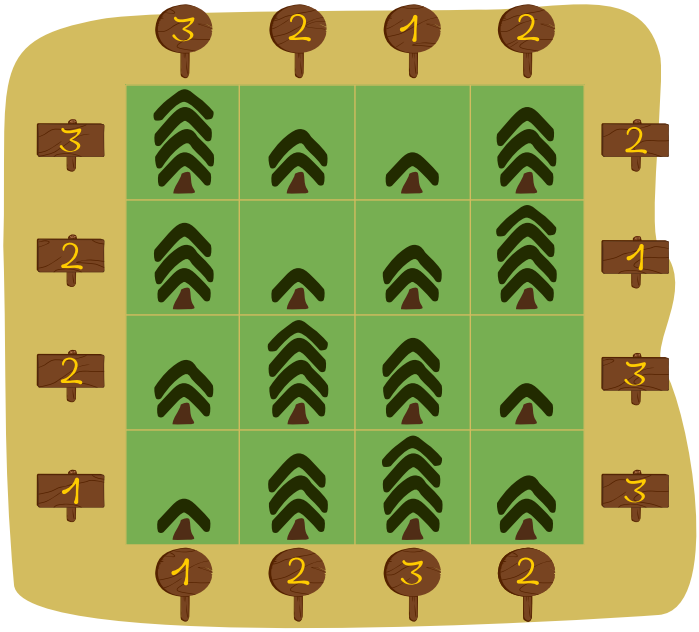
- In jeder Zeile (horizontalen Reihe) gibt es von jeder Höhe genau einen Baum.
- In jeder Spalte (vertikalen Reihe) gibt es von jeder Höhe genau einen Baum.



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?



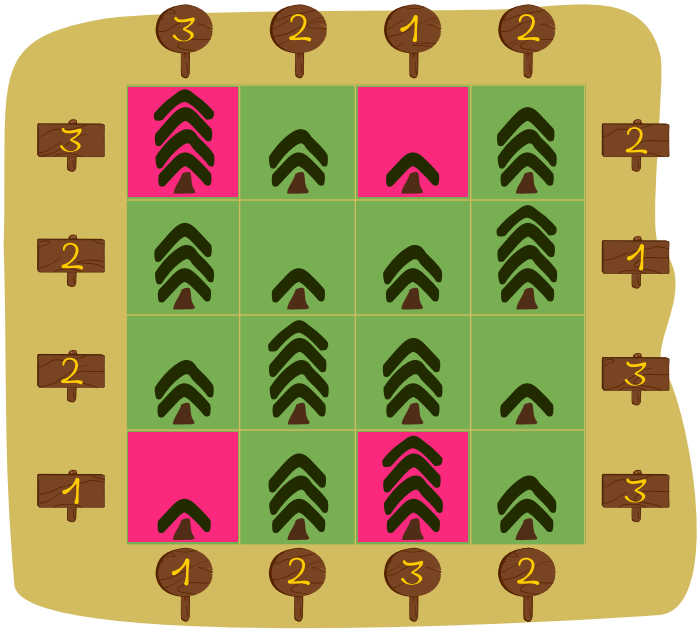
Lösung

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.

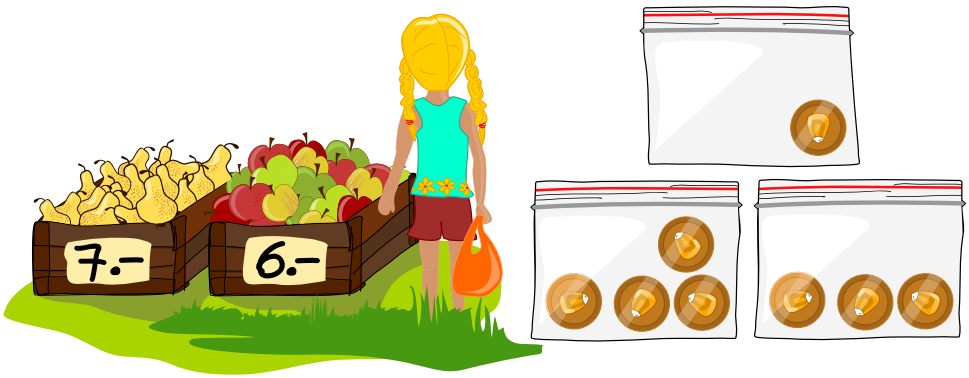
Drittens kann man solche Baumfelder mit Schildern zur Erzeugung von *selbst-verifizierenden Codierungen* einsetzen. Vorkommende Fehler beim Eintragen oder beim Informationstransport können dann automatisiert erkannt oder sogar korrigiert werden.

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`

13. 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.



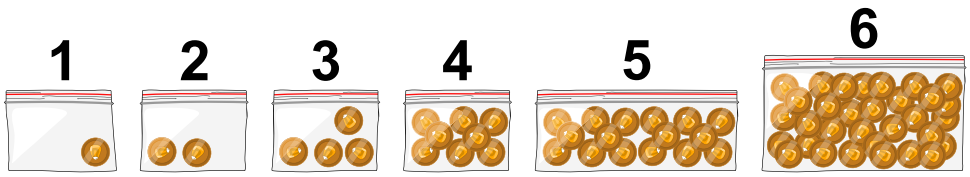
Heute hat Bina 63 identische Münzen dabei. Diese möchte sie so auf verschiedene Beutel aufteilen, dass sie jeden Betrag zwischen 1 und 63 Münzen mit verschlossenen Beuteln passend bezahlen kann.

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.

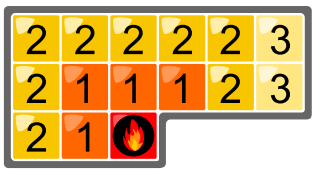
In Computern werden Bits gewöhnlich in Achtergruppen zusammengefasst; eine solche Achtergruppe nennt man ein Byte. Ein Byte kann $2^8 = 256$ verschiedene Zahlen darstellen, 0 bis 255.

Stichwörter und Webseiten

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

14. Hotspot-Bodenheizung

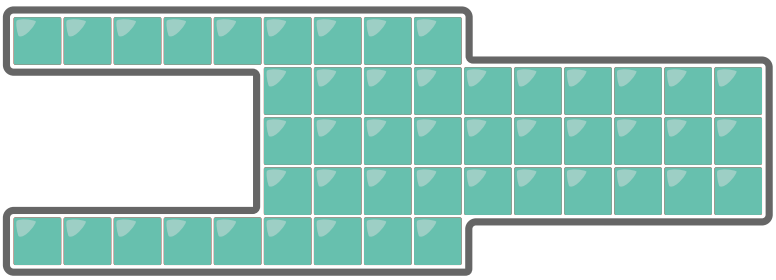
Luis mag es nicht, sich morgens im kalten Badezimmer umzuziehen, deswegen möchte er im neuen Haus eine Bodenheizung einbauen lassen. Der Heizungsmonteur empfiehlt ihm die innovative Hotspot-Bodenheizung: Ein Hotspot wird direkt unter einer Fliese montiert. Schaltet man den Hotspot ein, wird diese Fliese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

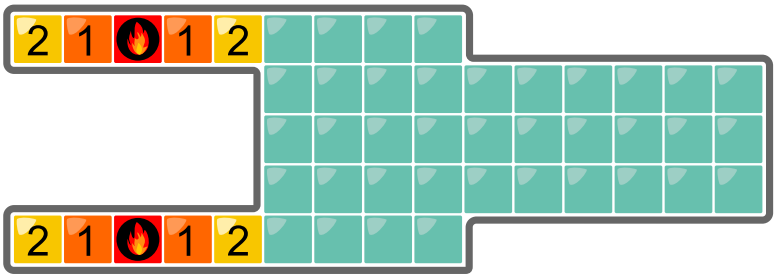


Lösung

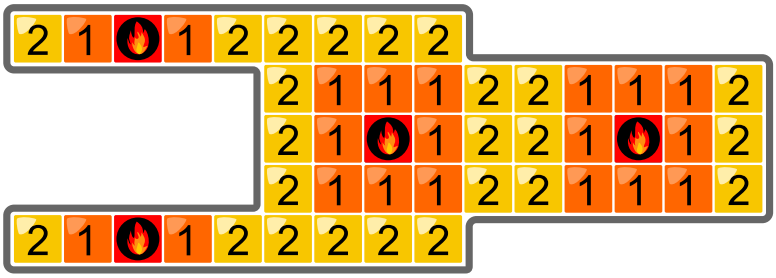
Wenn die 4 Hotspots wie im Bild ganz unten montiert werden, erwärmen sich alle Fliesen des Badezimmers nach dem Einschalten innerhalb 2 Minuten.

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.

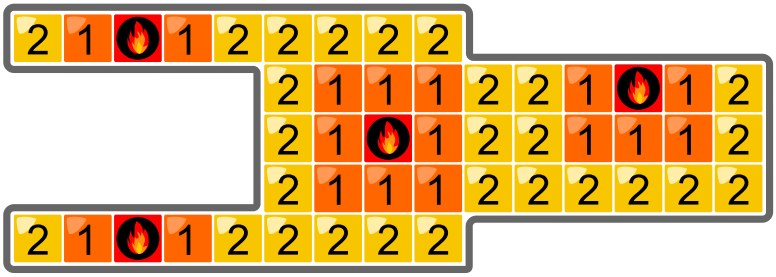
Es bietet sich jetzt an, beim Verteilen der Hotspots mit den beiden Gängen links zu beginnen. Mit je einem Hotspot in der Mitte der beiden Gänge werden gerade alle Fliesen des Ganges innerhalb von 2 Minuten erwärmt:

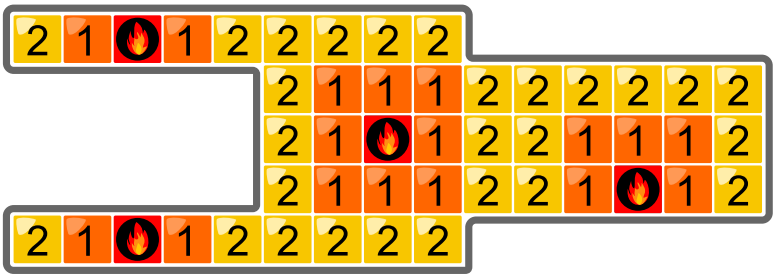


Die anderen zwei Hotspots können wir dann so platzieren:



Die folgenden beiden Platzierungen sind ebenfalls möglich:





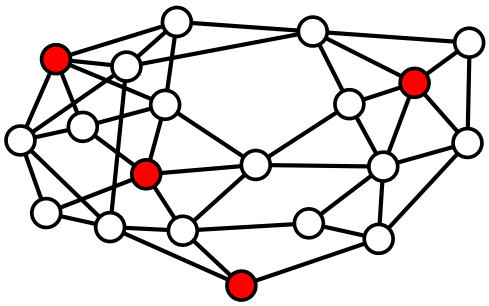
Wenn das Badezimmer eine andere Form hätte, könnten bei gleicher Fläche auch schon 2 Hotspots ausreichen, um das gesamte Badezimmer in 2 Minuten zu erwärmen.

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

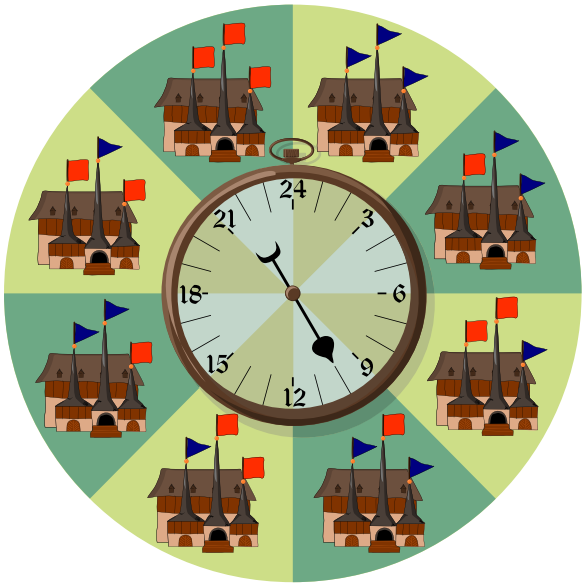
- Dominating set: `https://en.wikipedia.org/wiki/Dominating_set`

15. 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.

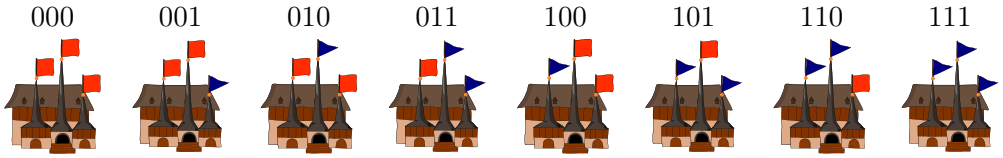
Die momentane Anordnung erfordert bei fast jedem Übergang nur einen Flaggenwechsel. Nur um Mitternacht müssen drei Flaggen gleichzeitig gewechselt werden. Die Biber wünschen sich eine bequeme Anordnung, bei der immer nur eine Flagge gewechselt werden muss.

Finde eine solche bequeme Anordnung.



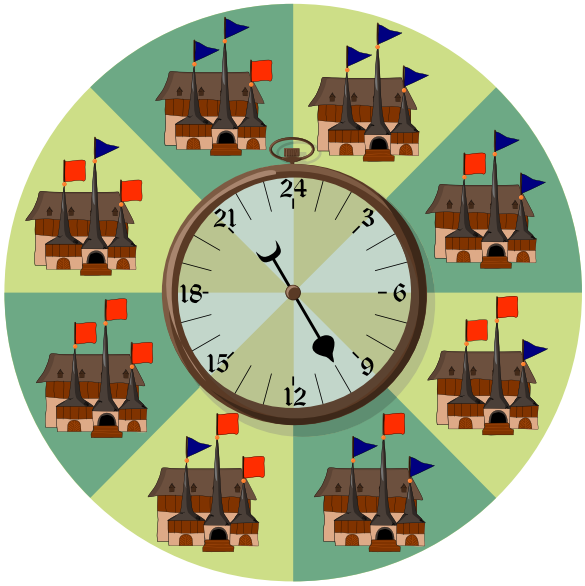
Lösung

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.

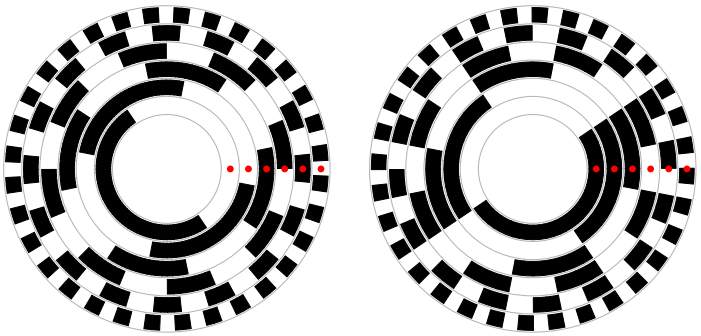
Diese Methode (Spiegeln der bestehenden Zahlenfolge und Ändern der nächsthöheren Stelle von 0 zu 1) kann man beliebig lange fortsetzen, um solche Anordnungen für beliebig viele statt nur drei

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`