INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Aufgaben und Lösungen 2020

Schuljahre 11/12/13

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

Herausgeber:

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

01010011010101100100100
10000001001011010101001 1
01010011010010010100010 1
00101101010100110101001 1
01001001010010010010000 1

SV!A

www.svia-ssie-ssii.ch
schweizerischerverein für informatikin
derausbildung//société suisse pour l'infor
matiquedans l'enseignement//società sviz
zera per l'informatica nell'insegnamento

Mitarbeit Informatik-Biber 2020

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

Projektleitung: Nora A. Escherle

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

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

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

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

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

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

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

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

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

HASLERSTIFTUNG

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

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

Vorwort

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

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

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

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

Der Informatik-Biber 2020 wurde in fünf Altersgruppen durchgeführt:

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

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

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

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

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

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

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

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

Für weitere Informationen:

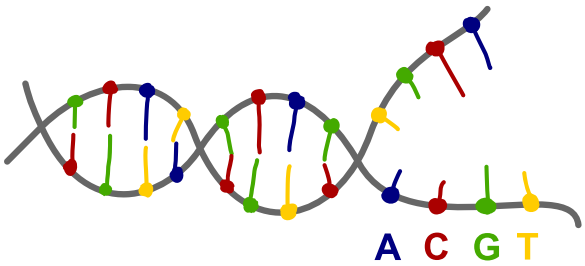
SVIA-SSIE-SSII Schweizerischer Verein für Informatik in der Ausbildung
Informatik-Biber
Nora A. Escherle

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

Inhaltsverzeichnis

1. 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.

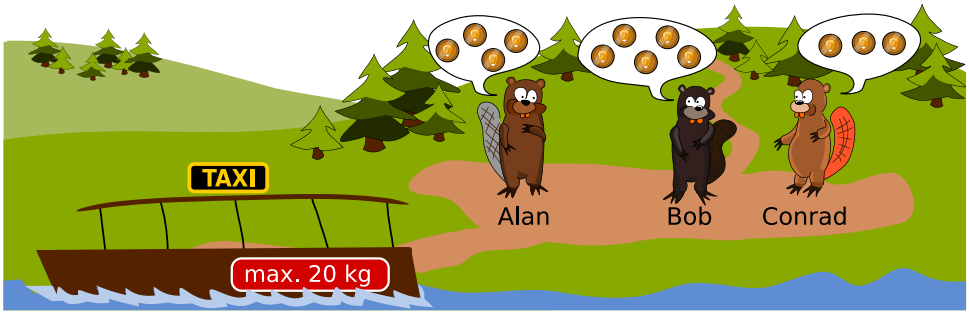
Der übliche Algorithmus zur Berechnung der Levenshtein-Distanz zweier Wörter basiert auf *dynamischer Programmierung*: Dabei schreibt die Levenshtein-Distanzen zwischen immer längeren Präfixen der beiden Wörter in eine Tabelle, bis man am Ende die beiden Präfixe den ganzen Wörtern entsprechen und man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

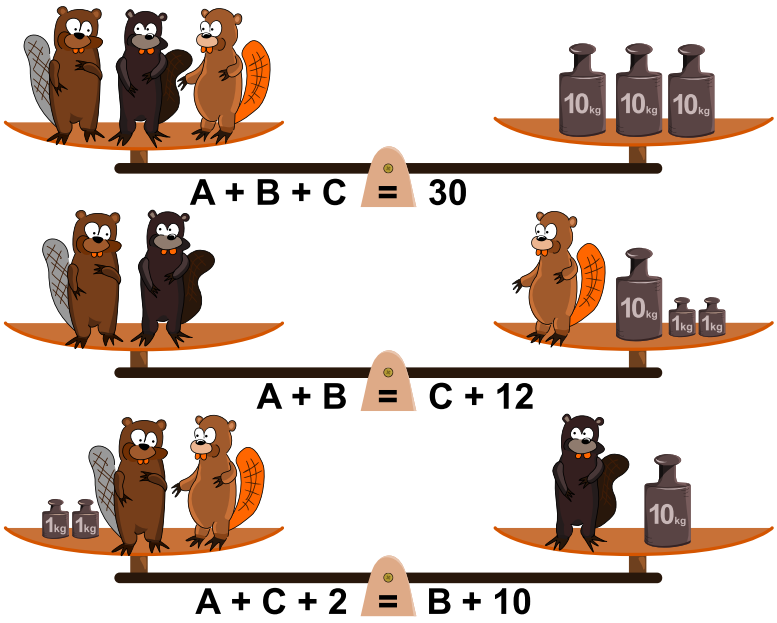
Stichwörter und Webseiten

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

2. 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

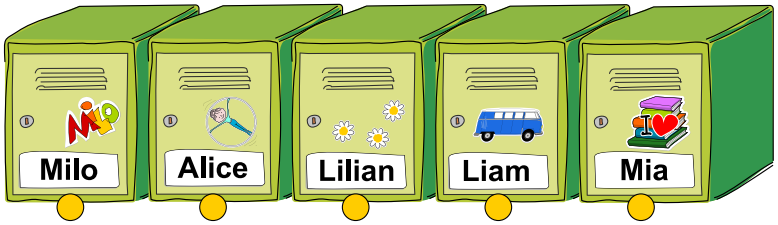
- Ganzzahlige lineare Optimierung: `https://de.wikipedia.org/wiki/Ganzzahlige_lineare_Optimierung`
- Nebenbedingung: `https://de.wikipedia.org/wiki/Nebenbedingung`
- Branch- and Boundverfahren: `https://de.wikipedia.org/wiki/Branch-and-Bound`
- Schnittebenenverfahren: `https://de.wikipedia.org/wiki/Schnittebenenverfahren`

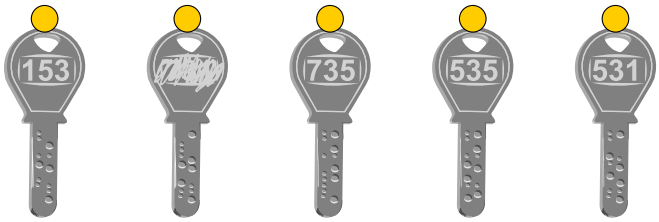
3. Schliessfächer

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».

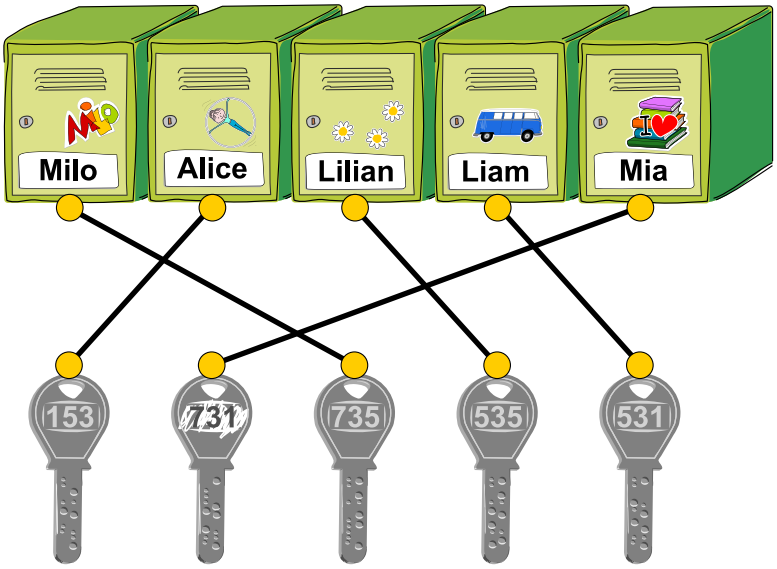
Ordne die Schlüssel den richtigen Schliessfächern zu. Zeichne dazu Linien zwischen den gelben Punkten.





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.

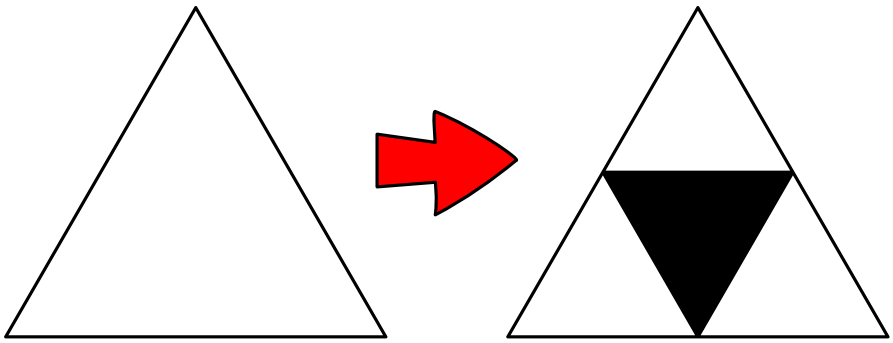
Wenn man nicht nur 10 Ziffern zur Codierung verwendet, sondern ein Symbol für jeden Buchstaben, dann kann man eine solche monoalphabetische Substitution auch als einfache Geheimschrift verwenden. Leider ist die monoalphabetische Verschlüsselungsmethode nicht sehr sicher, weil man die Zuordnung mit ein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

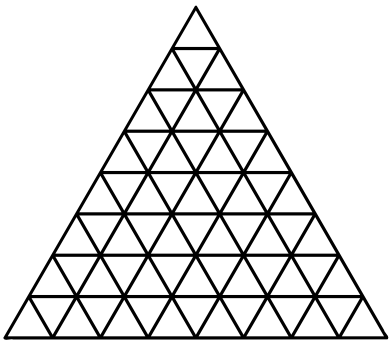
- Codierung, Monoalphabetische Substitution: `https://de.wikipedia.org/wiki/Monoalphabetische_Substitution`
- Kryptographie: `https://de.wikipedia.org/wiki/Kryptographie`

4. 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:

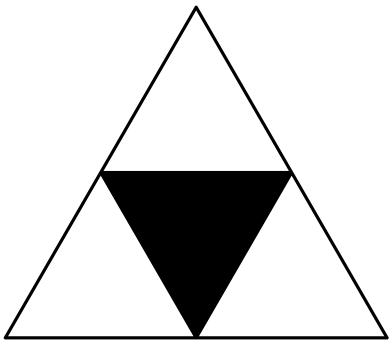


Zeichne die Figur, die nach drei Schritten entsteht. Male dazu die richtigen Teildreiecke an.

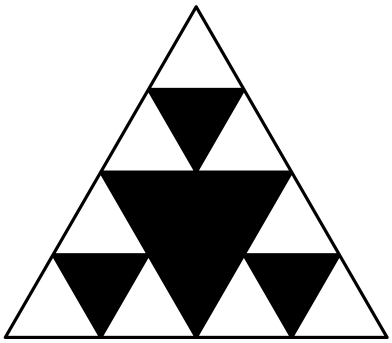


Lösung

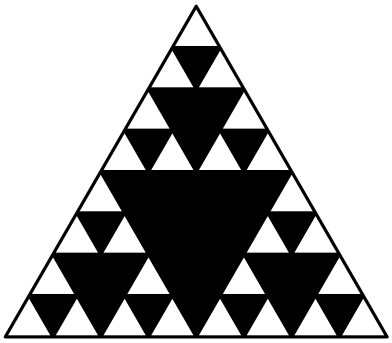
Nach dem ersten Schritt ist das mittlere Teildreieck schwarz und es es bleiben drei weisse Teildreiecke:



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.

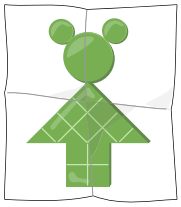
Das Konzept der *Rekursion* wird in der Informatik breit eingesetzt. Denn viele komplexe Objekte – zum Beispiel Fraktale – können mit Rekursion kompakt beschrieben werden und viele komplizierte Aufgaben – zum Beispiel das Problem der *Türme von Hanoi* – können mit sehr einfachen rekursiven Algorithmen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

Stichwörter und Webseiten

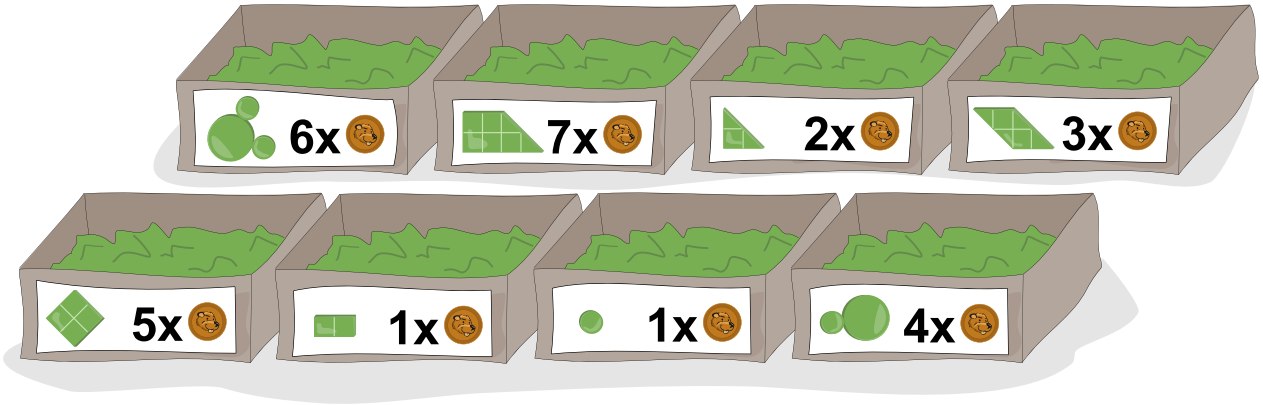
- Sierpiński-Dreieck: `https://de.wikipedia.org/wiki/Sierpinski-Dreieck`
- Rekursion: `https://de.wikipedia.org/wiki/Rekursion`
- Fraktal: `https://de.wikipedia.org/wiki/Fraktal`
- `https://de.wikipedia.org/wiki/Wacław_Sierpiński`
- `https://de.wikipedia.org/wiki/Türme_von_Hanoi#Rekursiver_Algorithmus`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

5. Legespiel

Giulia will Plättchen kaufen, um damit diese Figur legen:



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



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

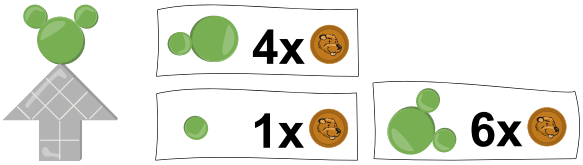
Wie viele Münzen muss Giulia ausgeben, wenn sie die günstigste Option wählt?

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

Lösung

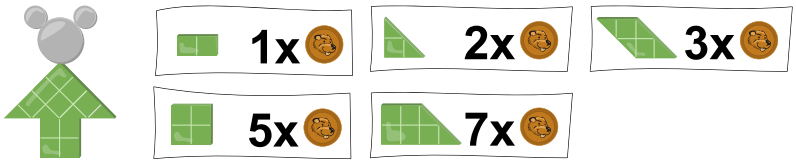
Die richtige Antwort lautet 13 Münzen.

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

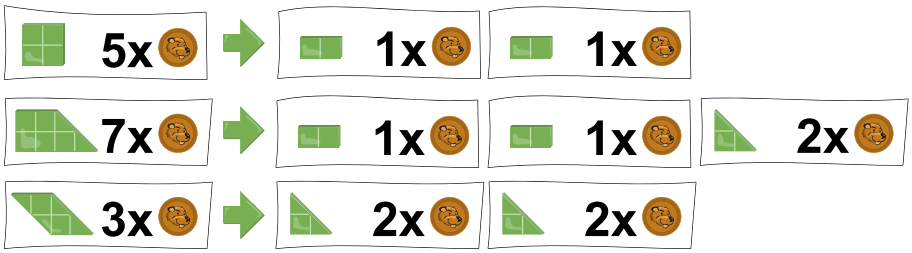


Für den Kopf gibt es nur zwei Möglichkeiten: Entweder benutzt man direkt das passende Plättchen für 6 Münzen oder aber man setzt es aus den anderen beiden runden Plättchen zusammen, die zusammen $4 + 1 = 5$ Münzen kosten. Die zweite Option ist günstiger, wir benutzen also diese.

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



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



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

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

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

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

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

Version A

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

Version B

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

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

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

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

Dies ist Informatik!

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

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

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

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

Stichwörter und Webseiten

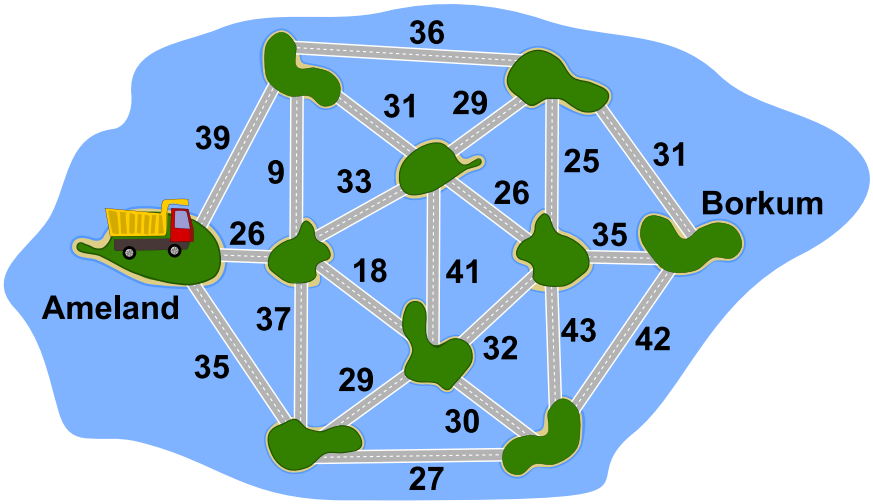
- Optimierungsproblem: `https://de.wikipedia.org/wiki/Optimierungsproblem`
- Teile und Herrsche: `https://de.wikipedia.org/wiki/Teile-und-herrsche-Verfahren`
- Modularität: `https://de.wikipedia.org/wiki/Modularität`
- Tangram: `https://de.wikipedia.org/wiki/Tangram`

6. 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.

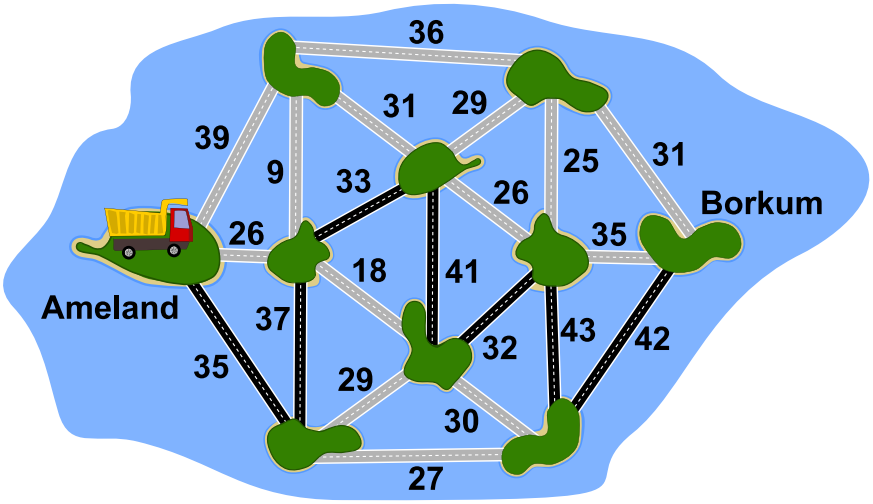
Biber Knuth möchte auf der Insel Borkum einen Strand aufschütten. Mit einer Fahrt will er daher möglichst viel Sand von der Insel Ameland zur Insel Borkum transportieren. Dabei ist ihm die Länge der Fahrtstrecke egal, er will aber über keine Brücke zweimal fahren.

Welchen Weg nach Borkum sollte er mit seinem Lastwagen nehmen?

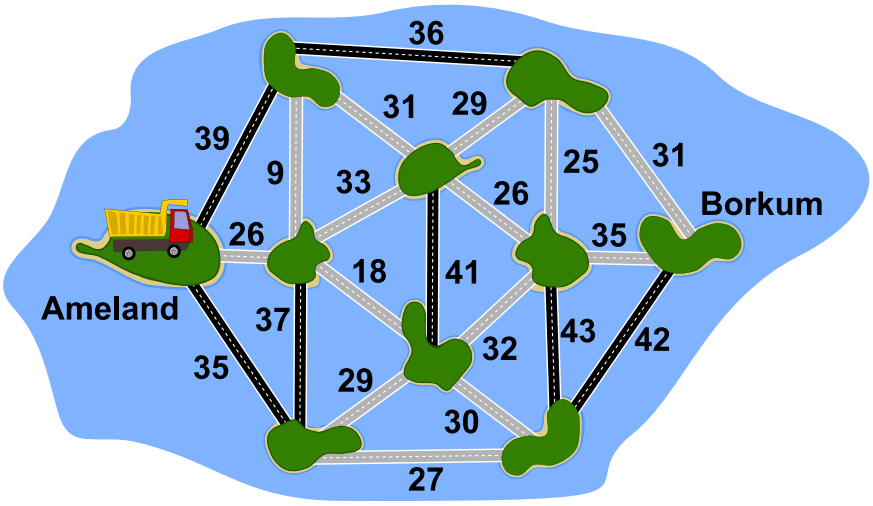


Lösung

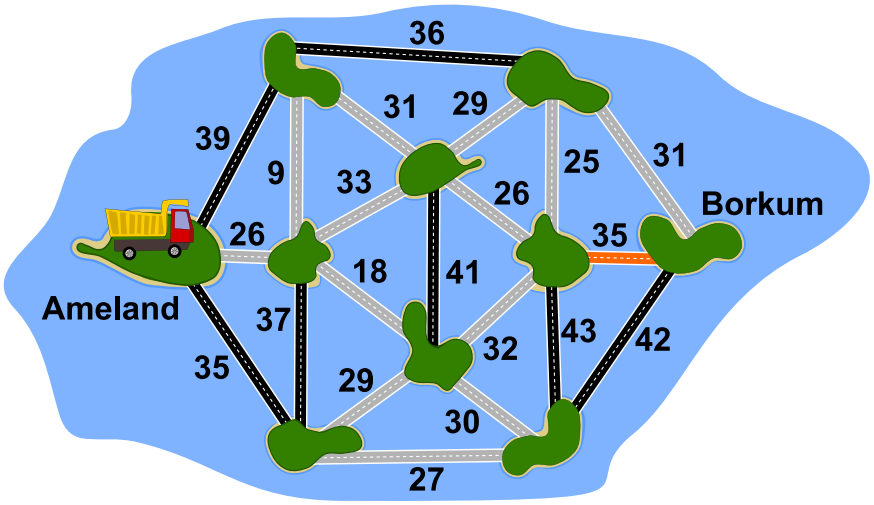
Für die Fahrt beträgt das maximale Gesamtgewicht eines Lastwagens 32 Tonnen. Er nimmt zum Beispiel den folgenden Weg:



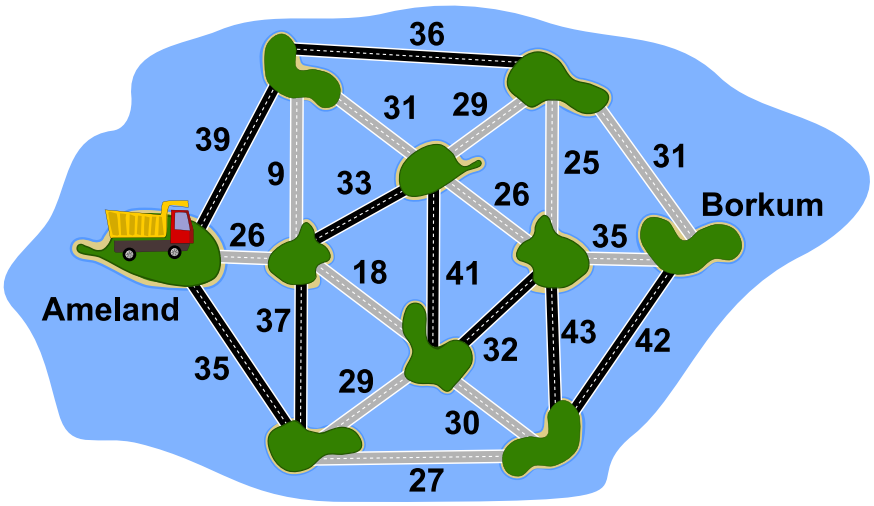
Um diesen zu bestimmen, können wir zum Beispiel virtuell zunächst alle Brücken aus der Karte entnehmen. Alle Brücken werden nach ihrer Belastbarkeit sortiert. Wir fangen mit denjenigen mit der grössten Belastbarkeit an und fügen diese der Karte zu. Danach kommen diejenigen mit der nächstgrössten Belastbarkeit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

7. 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.

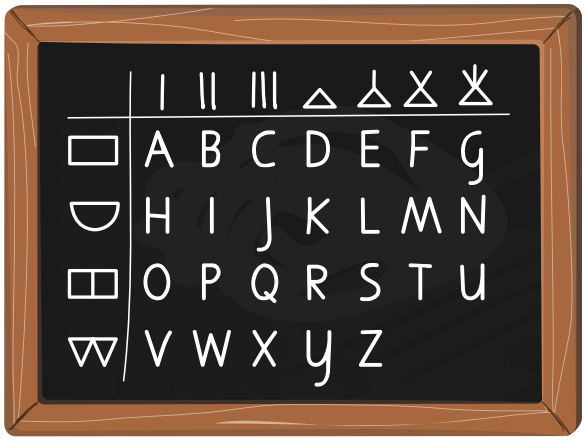
Rekonstruiere den ursprünglichen Text aus dem vorliegenden Geheimtext (dechiffriere den Geheimtext). Welcher der 4 Lösungsvorschläge stimmt?

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.

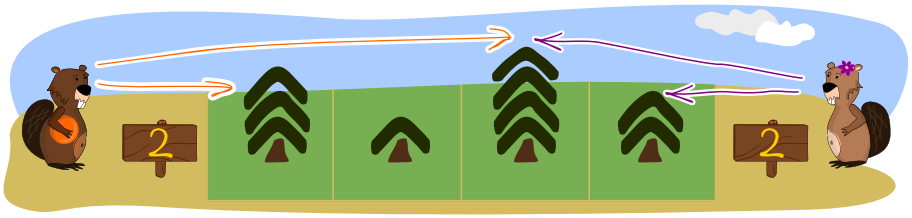
Stichwörter und Webseiten

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

8. 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:

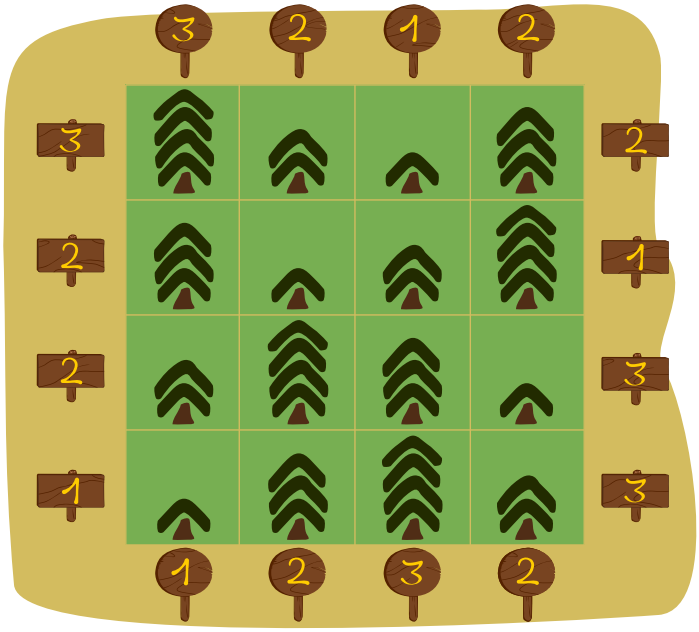
- In jeder Zeile (horizontalen Reihe) gibt es von jeder Höhe genau einen Baum.
- In jeder Spalte (vertikalen Reihe) gibt es von jeder Höhe genau einen Baum.



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?



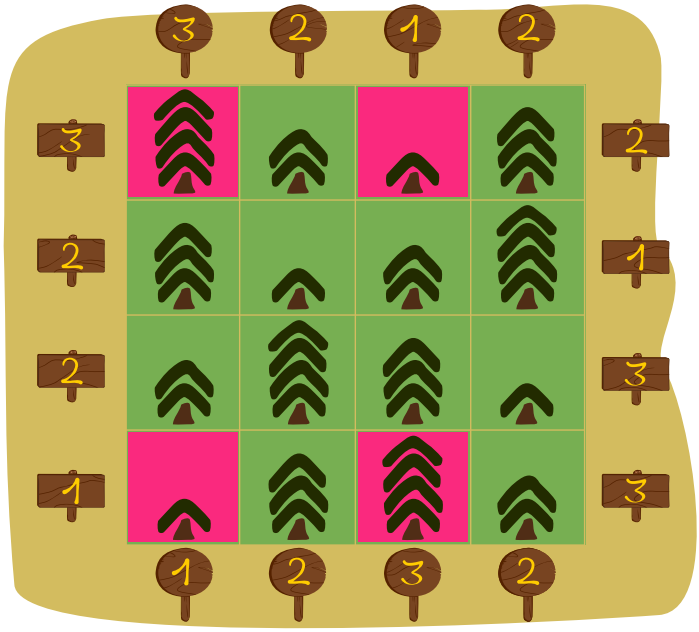
Lösung

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.

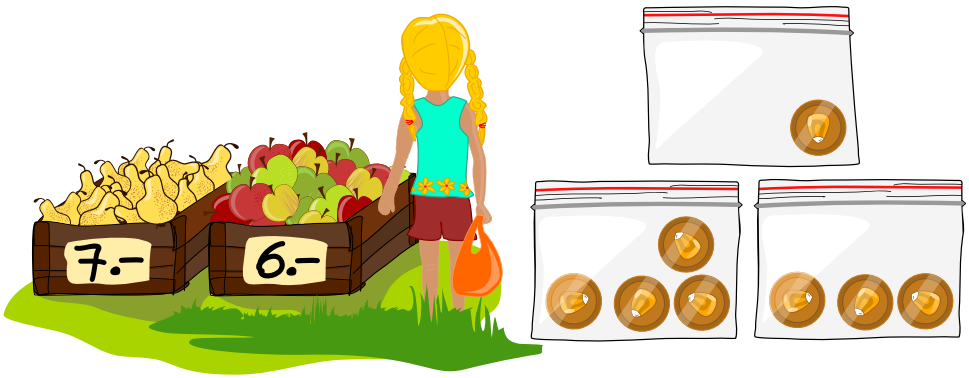
Drittens kann man solche Baumfelder mit Schildern zur Erzeugung von *selbst-verifizierenden Codierungen* einsetzen. Vorkommende Fehler beim Eintragen oder beim Informationstransport können dann automatisiert erkannt oder sogar korrigiert werden.

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`

9. 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.



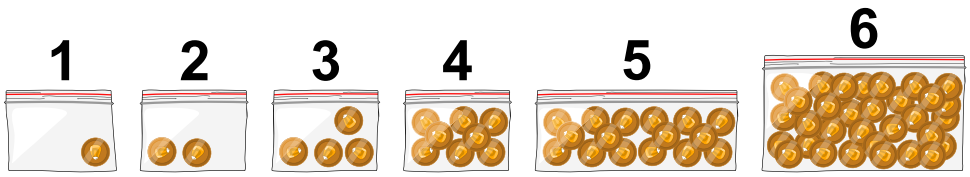
Heute hat Bina 63 identische Münzen dabei. Diese möchte sie so auf verschiedene Beutel aufteilen, dass sie jeden Betrag zwischen 1 und 63 Münzen mit verschlossenen Beuteln passend bezahlen kann.

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

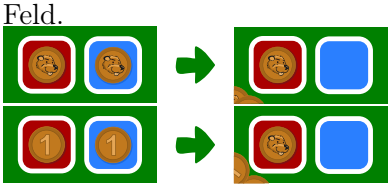
- Binärzahlen: `https://de.wikipedia.org/wiki/Binärcode`
- Datendarstellung
- Logik

10. Las Bebras

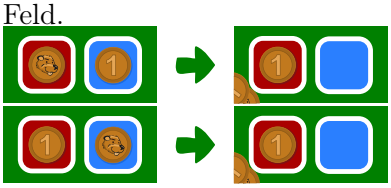
Im Casino «Las Bebras» kann Gloria bei John mit Münzen spielen. Gloria hat 4 Münzen mit Kopf, und auf der Rückseite mit Zahl 1. Gloria wirft die ersten 2 Münzen und legt eine auf das rote, die andere auf das blaue Feld.

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

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

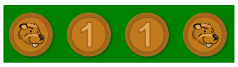


- Sind die Münzen unterschiedlich, legt John die neue Münze mit Zahl 1 nach oben aufs rote Feld.



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

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

A) 

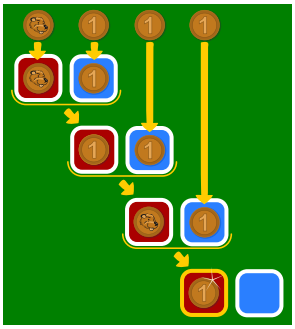
B) 

C) 

D) 

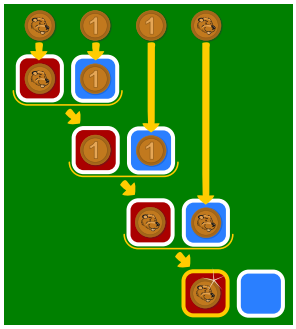
Lösung

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

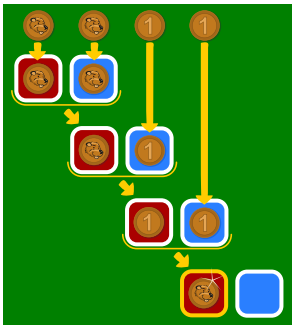


Bei allen anderen Reihenfolgen liegt am Schluss die Münze mit dem Kopf nach oben auf dem roten Feld.

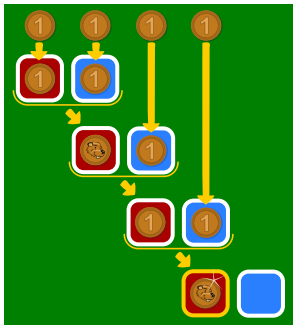
A)



B)



D)



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

Dies ist Informatik!

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

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

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

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

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

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

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

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

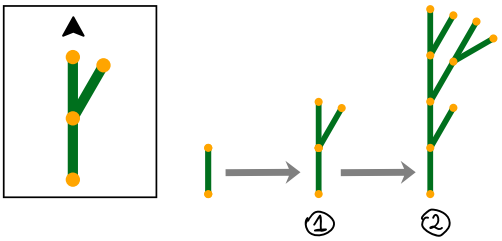
Stichwörter und Webseiten

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

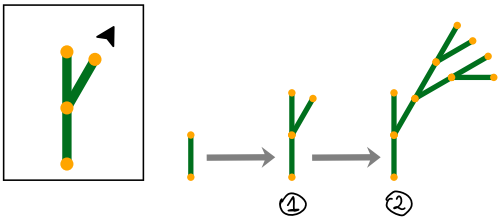
11. Digitale Bäume

Ein digitaler Baum wächst aus folgendem einzelnen Baumstück:  .

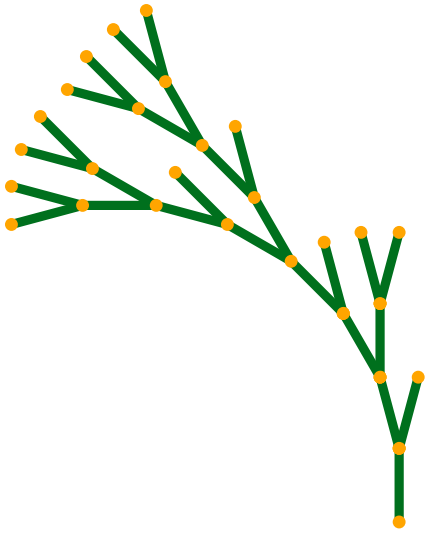
Er wächst schrittweise nach einer vorgegebenen Wachstumsregel.
Die Wachstumsregel gibt an, wie ein Baumstück durch eine Struktur von neuen Baumstücken ersetzt werden kann. In jedem Schritt wird jedes Baumstück auf diese Weise ersetzt. Eine Pfeilspitze gibt an, wo und in welche Richtung die Baumstücke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



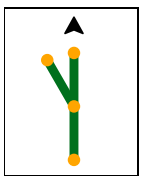
Rechts sind zwei Beispiele für eine Wachstumsregel und jeweils die zugehörigen ersten beiden Wachstumsschritte.

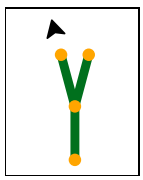


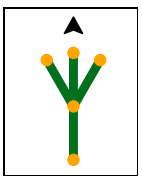
Der folgende digitale Baum ist in 3 Schritten gewachsen:

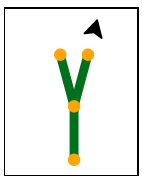


Nach welcher Wachstumsregel ist der digitale Baum gewachsen?

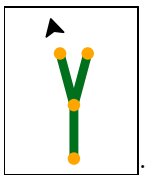
A) 

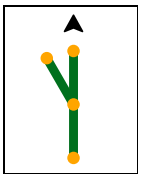
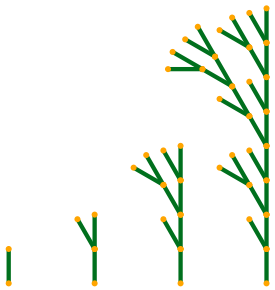
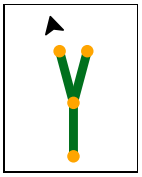
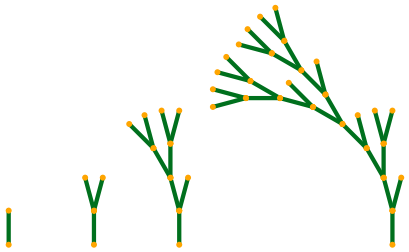
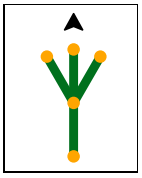
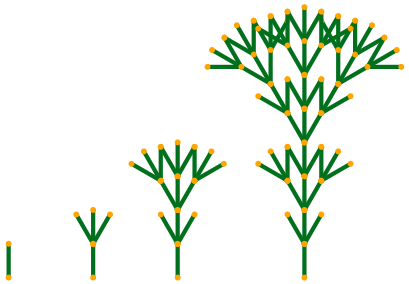
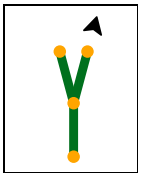
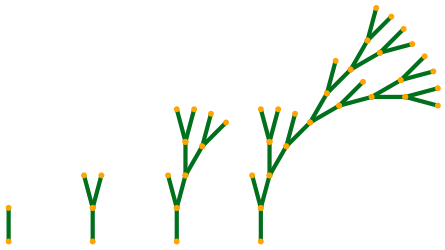
B) 

C) 

D) 

Lösung

Die richtige Antwort ist B)  .

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

Dies ist Informatik!

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

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

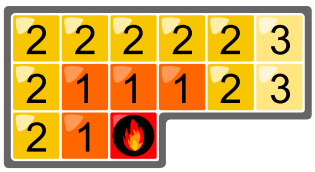
Die Bäume in der Aufgabe entstehen dadurch, dass man eine Regel auf jedes Baumstück anwendet, und auf die dabei entstehenden Baumstücke dann wieder und so weiter. Solche Vorgänge nennt man *rekursiv*. Das Konzept der *Rekursion* ist in der Informatik wichtig. Mit Rekursion ist es möglich, viele komplizierte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

Stichwörter und Webseiten

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

12. Hotspot-Bodenheizung

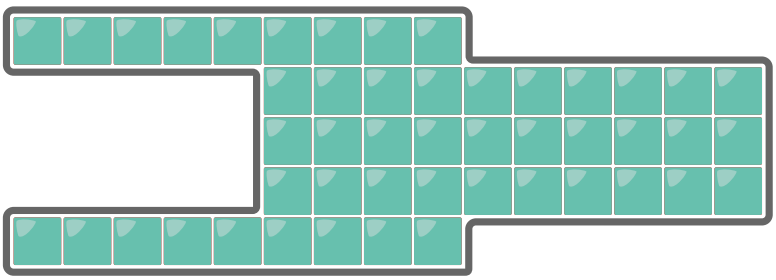
Luis mag es nicht, sich morgens im kalten Badezimmer umzuziehen, deswegen möchte er im neuen Haus eine Bodenheizung einbauen lassen. Der Heizungsmonteur empfiehlt ihm die innovative Hotspot-Bodenheizung: Ein Hotspot wird direkt unter einer Fliese montiert. Schaltet man den Hotspot ein, wird diese Fliese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

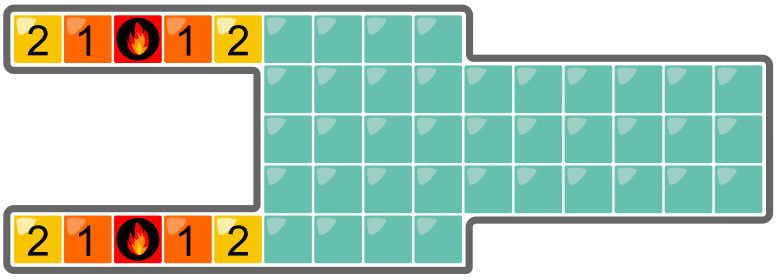


Lösung

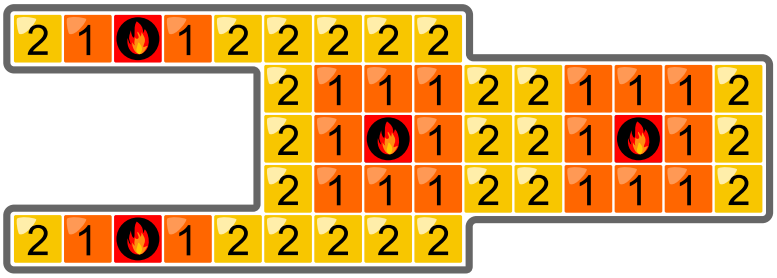
Wenn die 4 Hotspots wie im Bild ganz unten montiert werden, erwärmen sich alle Fliesen des Badezimmers nach dem Einschalten innerhalb 2 Minuten.

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.

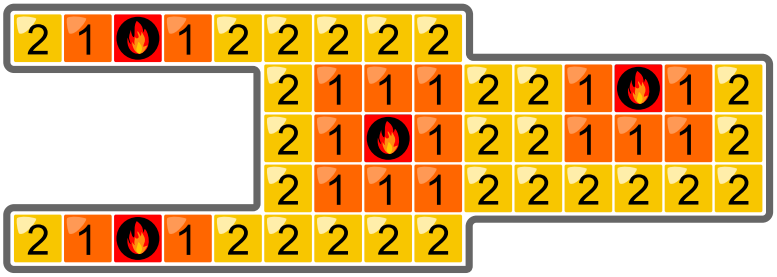
Es bietet sich jetzt an, beim Verteilen der Hotspots mit den beiden Gängen links zu beginnen. Mit je einem Hotspot in der Mitte der beiden Gänge werden gerade alle Fliesen des Ganges innerhalb von 2 Minuten erwärmt:

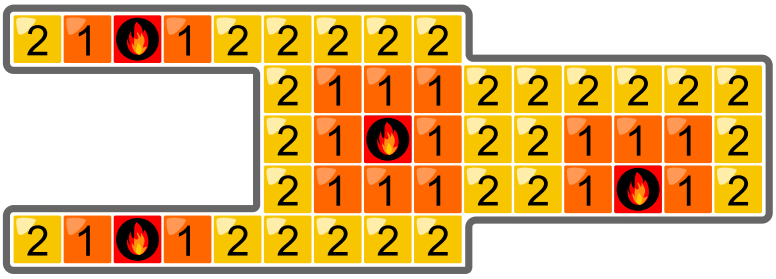


Die anderen zwei Hotspots können wir dann so platzieren:



Die folgenden beiden Platzierungen sind ebenfalls möglich:





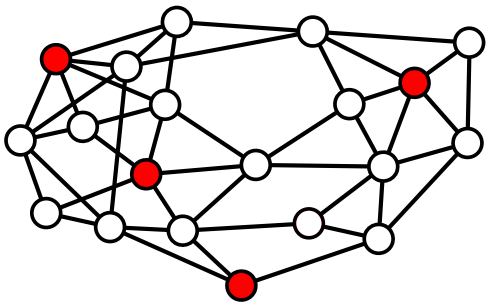
Wenn das Badezimmer eine andere Form hätte, könnten bei gleicher Fläche auch schon 2 Hotspots ausreichen, um das gesamte Badezimmer in 2 Minuten zu erwärmen.

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

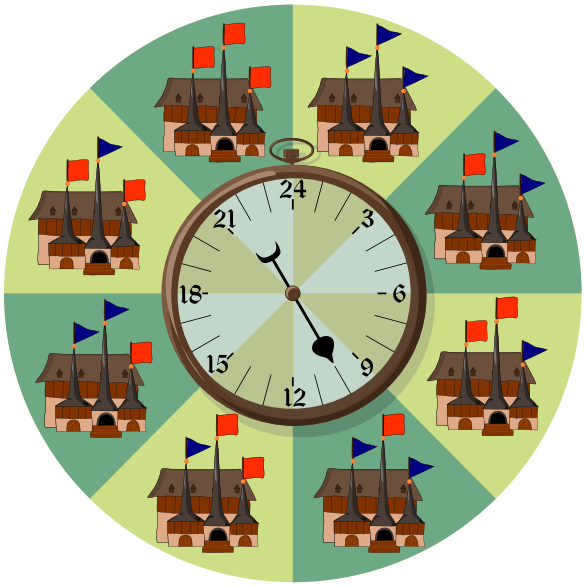
- Dominating set: `https://en.wikipedia.org/wiki/Dominating_set`

13. 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.

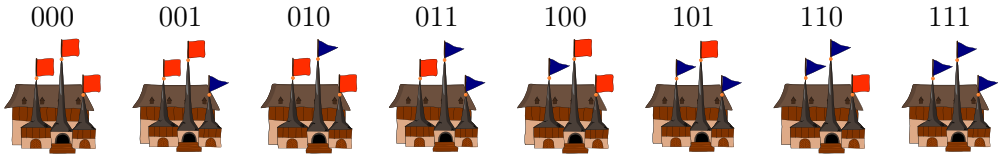
Die momentane Anordnung erfordert bei fast jedem Übergang nur einen Flaggenwechsel. Nur um Mitternacht müssen drei Flaggen gleichzeitig gewechselt werden. Die Biber wünschen sich eine bequeme Anordnung, bei der immer nur eine Flagge gewechselt werden muss.

Finde eine solche bequeme Anordnung.



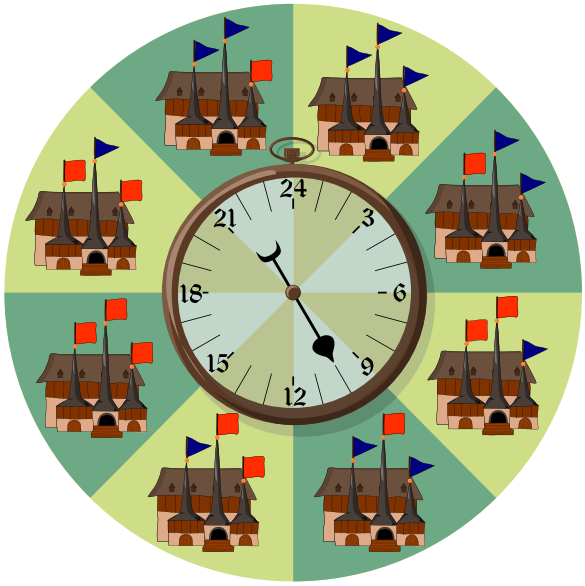
Lösung

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.

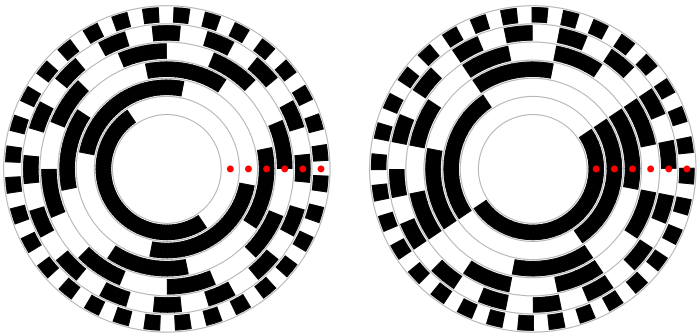
Diese Methode (Spiegeln der bestehenden Zahlenfolge und Ändern der nächsthöheren Stelle von 0 zu 1) kann man beliebig lange fortsetzen, um solche Anordnungen für beliebig viele statt nur drei

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.



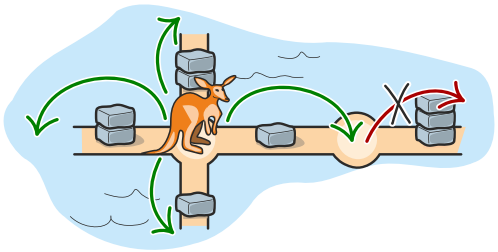
Im linken Bild sehen wir, dass sich der vierte Sensor genau auf der Grenze zwischen Schwarz und Weiss befindet. Der Sensor liest also entweder 001010 oder 001110. Beide Optionen sind akzeptabel, da sich der echte Winkel ja genau in der Mitte befindet. Wenn wir keinen Gray-Code haben, sieht das ganze 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

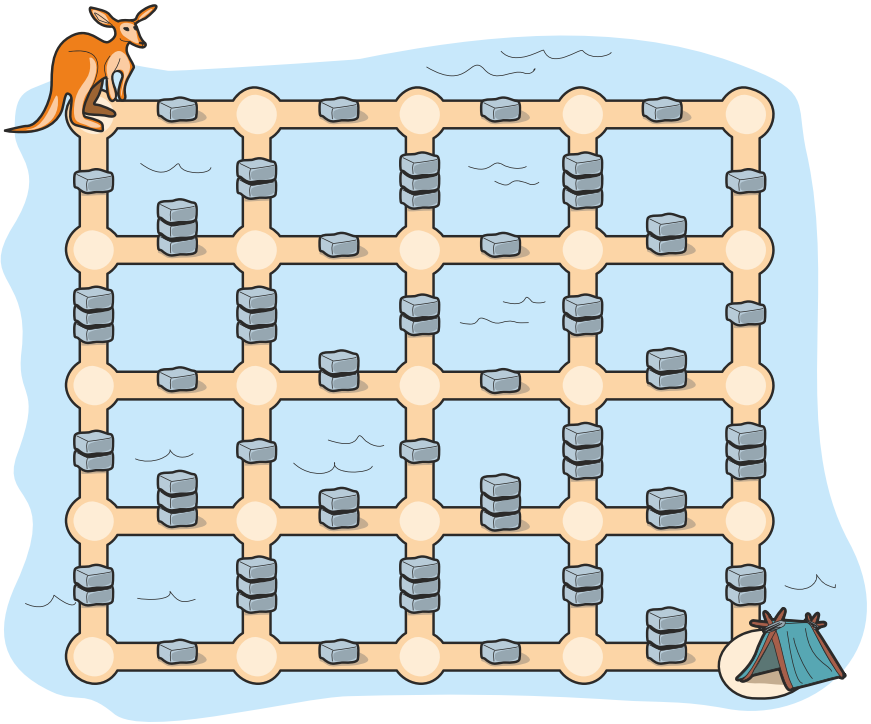
- Gray-Code: `https://de.wikipedia.org/wiki/Gray-Code`

14. Hüpfendes Känguru

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



Das Känguru möchte auf dem kürzesten Weg nach Hause kommen.

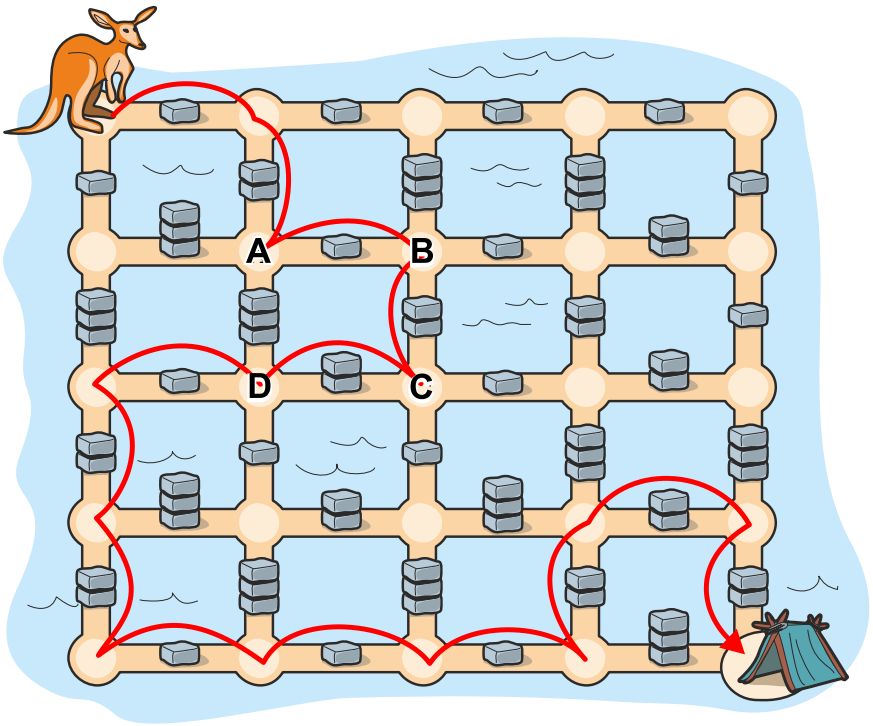


Wie viele Sprünge muss das Känguru machen, um auf dem kürzesten Weg nach Hause zu kommen?

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

Lösung

Die richtige Antwort ist E) 14 Sprünge:



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

Dies ist Informatik!

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

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

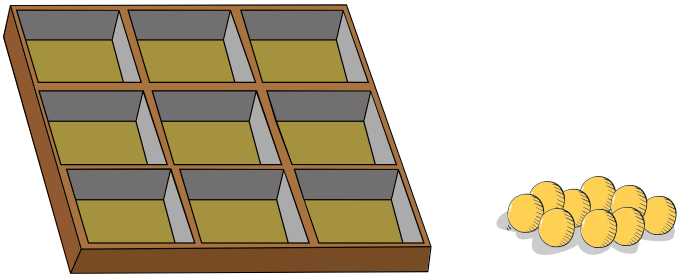
Die Aufgabe zeigt, dass es manchmal effizienter ist, von hinten nach einer Lösung zu suchen. Man spricht dann von einer *Rückwärtssuche*. Im vorliegenden Fall braucht es dadurch weniger Backtracking, weil es am Ende gar keine Optionen mehr gibt. Man kann nicht allgemein sagen, ob eine *Vorwärtssuche*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

Stichwörter und Webseiten

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

15. Fächer und Murmeln

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



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

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

Wie viele unterschiedliche Muster kann Hira so erzeugen?

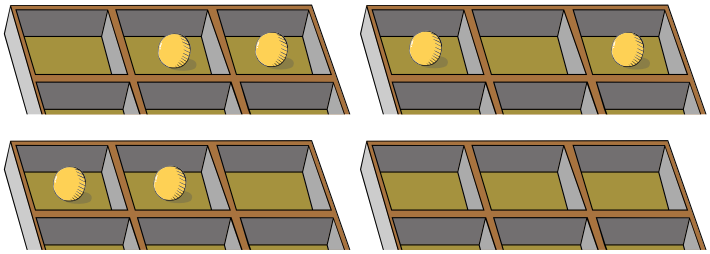
(Die Schachtel kann nicht rotiert werden. Das Muster mit nur einer Murmel links oben ist beispielsweise unterschieden vom Muster mit nur einer Murmel rechts oben.)

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

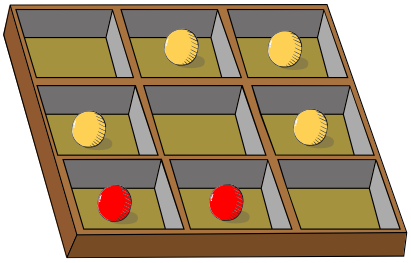
Lösung

Die richtige Antwort ist: B) 16.

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



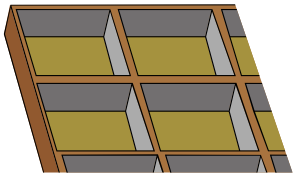
Genauso hat Hira 4 Möglichkeiten, die zweite Zeile zu füllen. Danach kann sie jedoch nichts mehr wählen, denn in den drei Spalten muss ja auch eine gerade Anzahl Murmeln liegen. Liegen in den zwei oberen Zeilen eine ungerade Anzahl Murmeln (also genau eine), so muss Hira in die dritte Zeile dieser Spalte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



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

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

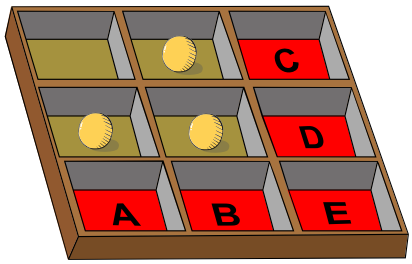
Eine zweite Option für das Zählen der Möglichkeiten ist die folgende: Man betrachtet dazu erst einen 2×2-Teil der Schachtel.



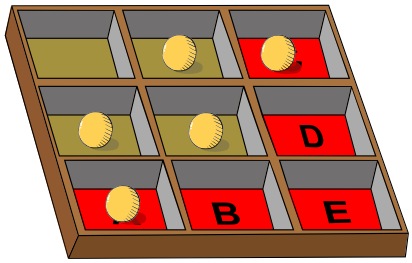
In diesem Teil gibt es 4 Fächer und jede kann entweder eine oder keine Murmel beinhalten. Daher gibt es $2^4 = 16$ verschiedene Möglichkeiten diesen Teil mit Murmeln zu füllen.

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

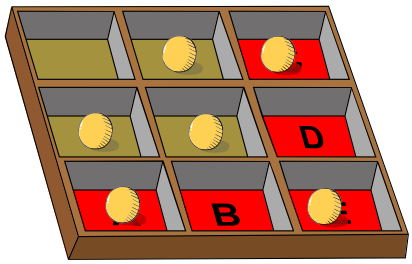
Zum Beispiel könnte Hira den betrachteten 2×2-Teil wie folgt füllen:



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



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



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

Dies ist Informatik!

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

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

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

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

Stichwörter und Webseiten

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

A. Aufgabenautoren

Tony René Andersen
Michael Barot
Wilfried Baumann
Maksim Bolonkin
Andrey Brodnik
Sarah Chan
Marios O. Choudary
Valentina Dagienė
Tolmantas Dagys
Christian Datzko
Susanne Datzko
Amirmohammad Djazbi
Nora A. Escherle
Lidia Feklistova
Fabian Frei
Gerald Futschek
Jens Gallenbacher
Tom Grubb
Mathias Hiron
Juraj Hromkovič
Alisher Ikramov
Thomas Ioannou
Ungyeol Jung
Vaidotas Kinčius
Ritambhra Korpal
Regula Lacher
Vu Van Luan
Pedro Marcelino
Hamed Mohebbi
Kwangsik Moon
Xavier Muñoz
Vania Natali
Rana R. Natawigena
Ágnes Erdősné Németh
Andrei Nicolicioiu
Jean-Philippe Pellet
Wolfgang Pohl
Raymond Chandra Putra
Peter Rossmanith
Vipul Shah
Fei Shang
Wenpan Sheng
Timur Sitdikov
Maciej M. Sysło
Congyu Tian
Jiří Vaníček
Troy Vasiga
Fan Wang
Yang Xing
Binru Zhi

B. Sponsoring: Wettbewerb 2020

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

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

http://www.verkehrshaus.ch/

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

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

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

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

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

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

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

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

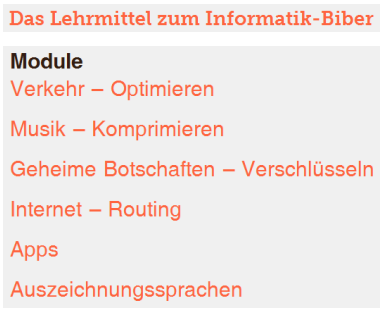
http://www.phlu.ch/
Pädagogische Hochschule Luzern

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

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

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

C. Weiterführende Angebote



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

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.