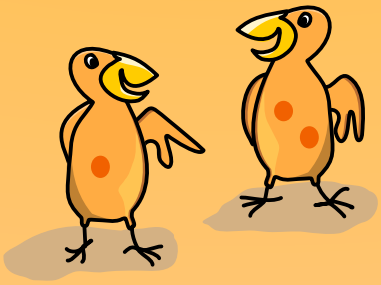
INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Aufgaben und Lösungen 2021

Schuljahre 11/12/13



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

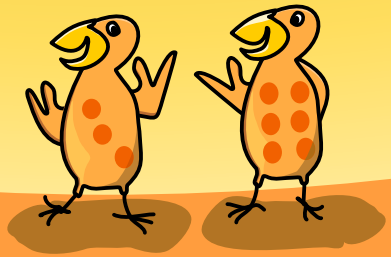
Herausgeber:

Susanne Datzko, Fabian Frei,
Jean-Philippe Pellet

0101001101010110010010010000001001011010101001101010010100110101001101001010010100010100101010100110101001100100101001001010000101

SVIA

www.svia-ssie-ssii.ch
schweizerischerverein für informatikinderausbildung//société suisse pour l'informatique dans l'enseignement//società svizzera per l'informatica nell'insegnamento



Mitarbeit Informatik-Biber 2021

Masiar Babazadeh, Susanne Datzko, Fabian Frei, Martin Guggisberg, 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
Bernadette Spieler: Pädagogische Hochschule Zürich

Die Aufgabenauswahl wurde erstellt in Zusammenarbeit mit den Organisatoren von Bebras in Deutschland, Österreich, Ungarn, Slowakei und Litauen. Besonders danken wir:
Valentina Dagienė, Tomas Šiaulys, Vaidotas Kinčius: Bebras.org
Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend: Bundesweite Informatikwettbewerbe (BWINF), Deutschland
Wilfried Baumann, Liam Baumann, Anoki Eischer, Thomas Galler, Benjamin Hirsch, Martin Kandlhofer, Katharina Resch-Schobel: 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: UK Bebras Administrator, Vereinigtes Königreich

Für den Support während den Wettbewerbswochen danken wir:
Hanspeter Erni: Schulleitung Sekundarschule Rickenbach

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

Diese Broschüren sind dem Andenken an Martin Guggisberg gewidmet.

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 2021 wurde vom Schweizerischen Verein für Informatik in der Ausbildung SVIA durchgeführt und von der Hasler Stiftung unterstützt.

HASLERSTIFTUNG

Dieses Aufgabenheft wurde am 24. August 2022 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 2021 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 2021 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/`

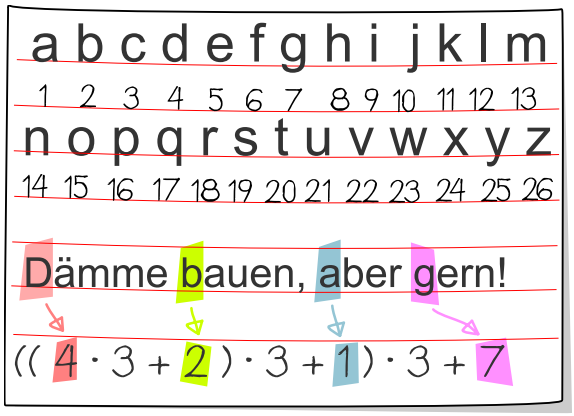
Inhaltsverzeichnis

1. Bibliothek

Susi ist mit Tim in der Biber-Bibliothek. Sie wollen ein Buch ausleihen: «Dämme bauen, aber gern!»

Tim geht zu Regal 1, greift in Reihe 3, Fach 6 und holt das Buch heraus. Susi ist beeindruckt. Tim erklärt Susi, wie man den Ort eines Buches bestimmt:

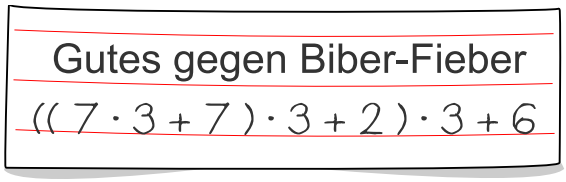
Man nimmt von jedem Wort im Titel den Anfangsbuchstaben und bestimmt seine Position im Alphabet. Nach und nach werden diese Positionen addiert, aber vor jedem Addieren wird der bisher erreichte Wert mit 3 multipliziert. Für das gewünschte Buch ergibt sich 136. Schon ist klar, wo das Buch steht.



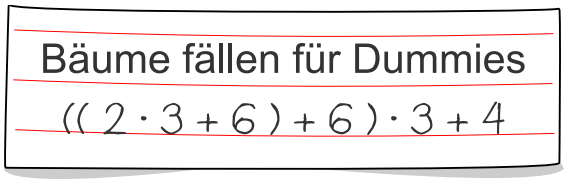
Nun stellt Susi für ihre Lieblingsbücher die entsprechenden Rechnungen auf. In einem Fall hat sie aber einen Fehler gemacht.

In welchem?

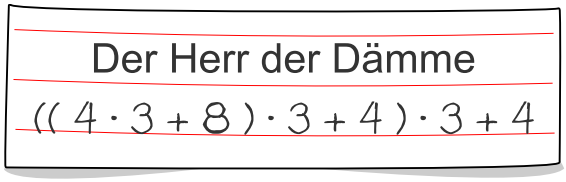
A)



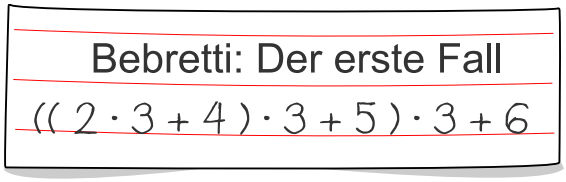
B)



C)

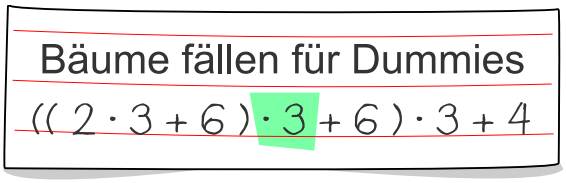


D)



Lösung

Susi hat fast alles richtig gemacht: Sie hat immer die richtigen Positionswerte addiert, und sie hat die Zwischenergebnisse immer mit 3 multipliziert – mit einer Ausnahme: In Antwort B hat sie Letzteres einmal vergessen.



Dies ist Informatik!

Mit den «Orts-Bestimmungs-Ausdrücken» ermöglicht die Bibliothek ihren Besuchern, die Standorte der Bücher genau zu bestimmen. So muss niemand lange suchen. Eines müssen die Bibliothek und Besucher aber beachten: Für verschiedene Bücher können die Ausdrücke und damit auch deren Ergebnisse gleich sein. Zum Beispiel stehen «Bäume fällen für Dummies» und «Biber finden Fichten dufte» im gleichen Fach. Die Fächer dürfen also nicht zu klein sein, oder sie müssen flexibel angepasst werden können.

Auch bei Daten, die in Computerspeichern abgelegt werden, ist es eine gute Idee, wenn ihr Speicherort direkt aus den Daten selbst berechnet werden kann. Dafür wurden in der Informatik *Hash-Funktionen* entwickelt: mathematische Funktionen, die aus dem Inhalt der Daten bzw. eines Teils der Daten einen Wert berechnen, der direkt den Speicherort angibt – so wie bei den Buchtiteln in dieser Biberaufgabe. Gute Hash-Funktionen sorgen dafür, dass sich in möglichst wenigen Fällen der gleiche Wert ergibt. Kommt eine solche Kollision doch einmal vor, kennt die Informatik gute Methoden, damit umzugehen.

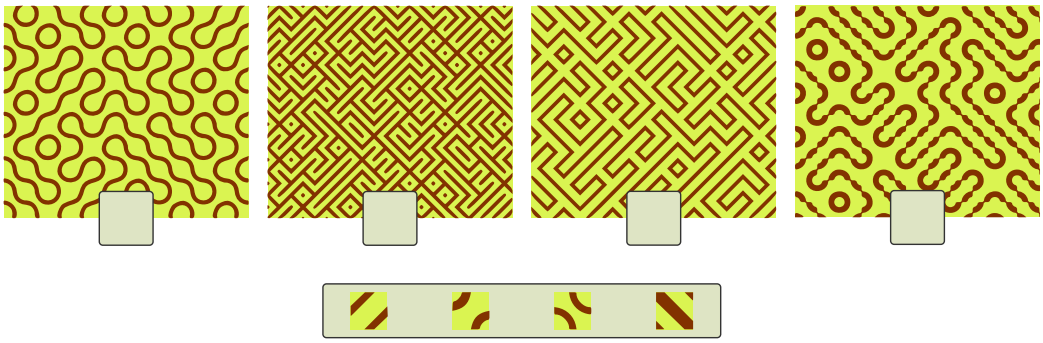
Stichwörter und Webseiten

- Hashfunktion: `https://de.wikipedia.org/wiki/Hashfunktion`
- Hashtabelle: `https://de.wikipedia.org/wiki/Hashtabelle`
- `http://www.abenteuer-informatik.de/PDF/ai2020_oa_leseversion.pdf`, Kapitel 11: Ordnung im Chaos

2. Fliesenmuster

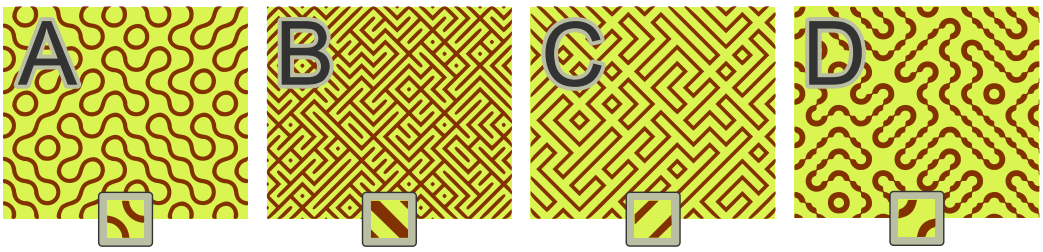
Die folgenden Muster wurde jeweils durch eine einzelne Fliese erzeugt. Die einzelnen Fliesen sind vergrössert dargestellt.

Ordne die Fliesen ihren möglichen Mustern zu.

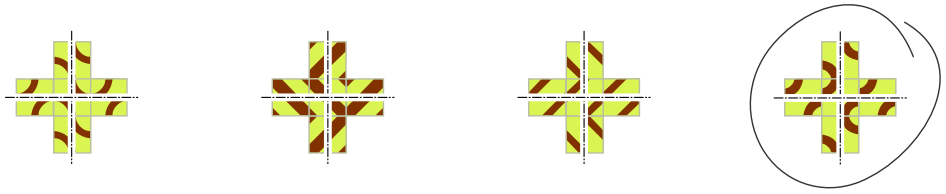


Lösung

Das ist die richtige Zuordnung:



Legt man jeweils 5 Fliesen aneinander und vergleicht sie genauer, erkennt man deutliche Unterschiede:



Fliese hat als einzige der Fliesen vier Seiten, die nicht genau zueinander passen. Nur dadurch können Linien mit veränderlicher Breite wie in Muster D entstehen. Fliese ist die einzige, die quadratischen Punkte in Muster B erzeugen kann, nämlich mit jeweils vier zusammenstossenden Dreiecken. Zudem hat sie den grössten Flächenanteil von Braun gegenüber Gelb, genau wie B; auch daran kann man die Zusammengehörigkeit erkennen. Somit verbleibt für die runden Formen von Fliese nur Muster A als mögliches Ergebnis und für die geraden Formen von Fliese nur Muster C.

Dies ist Informatik!

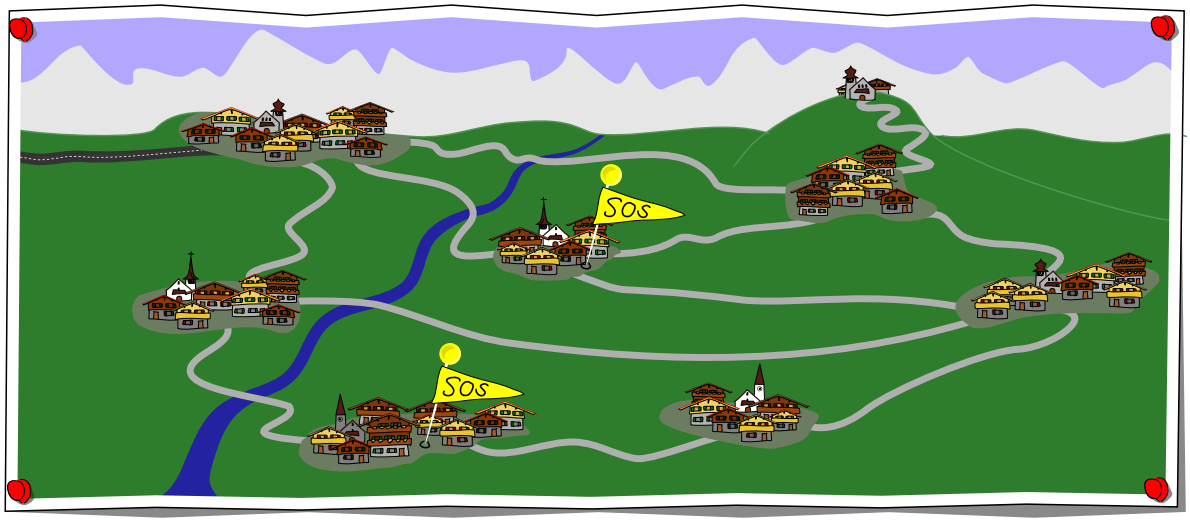
Diese Fliesen sind nach Sébastien Truchet (* 1657; † 1729) benannt, der an verschiedenen Varianten dieser Fliesen gearbeitet hat. Kacheln mit 4 gleichen Seiten bilden eine Untermenge von Truchet-Kacheln (Truchet-Kacheln müssen aber nicht unbedingt 4 gleiche Seiten haben, wie in 3 von den Mustern oben). Dass mit ganz einfachen Bausteinen komplexe Muster erstellt werden können, ist eine interessante Eigenschaft, die uns in der Informatik immer wieder begegnet. Truchet-Kacheln werden in der Mathematik und der Informatik untersucht und in Computerspielen verwendet, um Labyrinthe oder Dekorationen zu erstellen.

Stichwörter und Webseiten

- Truchet-Kacheln: `https://en.wikipedia.org/wiki/Truchet_tiles`

3. SOS aus den Bergen

Einige Bergdörfer werden aus der grossen Stadt über folgendes Strassennnetz versorgt:

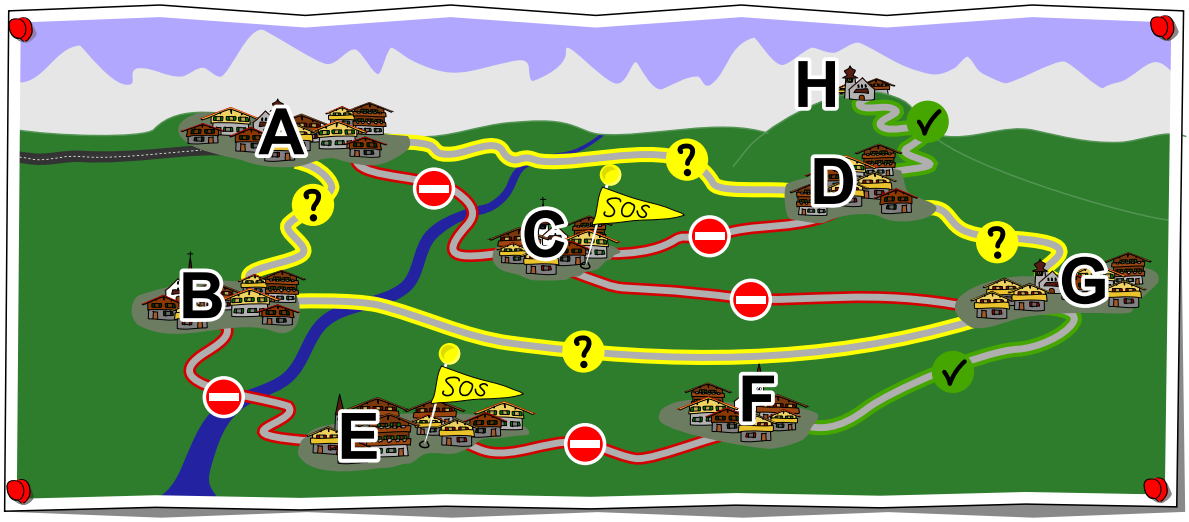


Nach einem Unwetter melden mehrere Dörfer, dass diese nicht mehr erreichbar sind, nämlich jene mit den SOS-Markierungen. Wir können daraus schliessen, dass einige Strassen blockiert sind.

Gib für jede Strasse zwischen den Dörfern in diesem Strassennetz an, ob diese (1) blockiert ist ⛔, (2) befahrbar ist ✓, oder (3) ob wir nicht ohne weitere Informationen sagen können, ob die Straße befahrbar oder blockiert ist ?.

Lösung

Die Karte zeigt, was wir über die Verbindungen im Strassennetz wissen:



Wir beginnen mit dem Aufspüren der blockierten Strassen. Die zwei Strassen, die zu Dorf E führen, können blockiert sein, da ansonsten Dorf E noch erreichbar wäre. Ebenso sind die drei Strassen zu Dorf C blockiert, da sonst Dorf C noch erreichbar wäre.

Als nächstes suchen wir die Strassen, die befahrbar sein müssen. Die Strasse zwischen Dorf G und F muss befahrbar sein, da ansonsten, aufgrund der blockierten Strasse zwischen Dorf F und E, das Dorf F nicht erreichbar wäre. Auch die Strasse zwischen der Kirche H und dem Dorf D muss befahrbar sein, da H erreichbar ist und nur über D erreicht werden kann.

Nun bleiben die nur möglicherweise befahrbaren Strassen übrig. Da die Dörfer B, G und D mehrfach mit dem Dorf A verbunden sind, können wir nicht sagen können, welche der verbleibenden Strassen befahrbar sind. So könnte das Dorf B beispielsweise über Dorf A, aber auch Dorf G erreicht werden. Dasselbe gilt auch für Dorf D. Das Dorf G kann entweder über Dorf B oder D versorgt werden. Irgendeine der Strassen im Kreislauf A – B – G – D – A könnte also blockiert sein und diese 4 Dörfer könnten trotzdem alle erreichbar bleiben.

Dies ist Informatik!

So wie in Strassennetzen können auch bei Computernetzwerken Verbindungen fehlerhaft, überlastet oder ganz defekt sein. Um Ausfälle zu verhindern, werden oft Sicherheitsmassnahmen, wie z. B. mehrere Verbindungen zu einem Ort, eingeplant. Dies nennt man *Redundanz*.

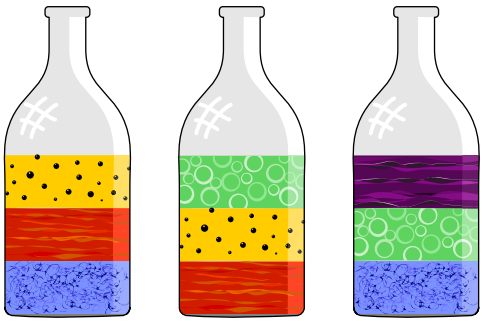
Das Beheben von Fehlern in einem System ist eine Aufgabe, die Informatiker sehr oft erledigen müssen, nicht nur in Computernetzwerken, sondern auch in der Softwareentwicklung. Um einen Fehler zu beheben, muss man seine genaue Quelle identifizieren, und dieser Prozess erfolgt meist schrittweise in mehreren Schritten. Einige Programmierer glauben, dass man nie alle Fehler und Bugs in einem Programm finden kann.

Stichwörter und Webseiten

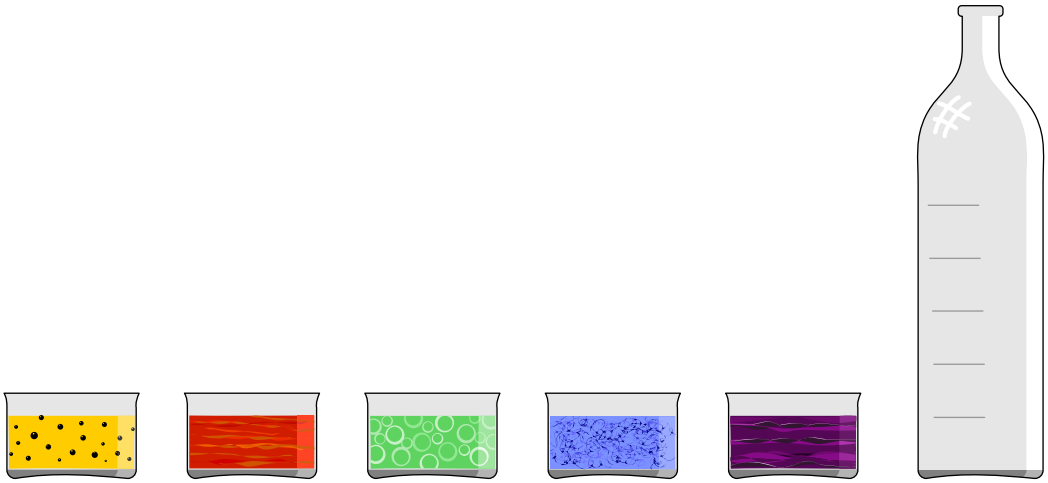
- Redundanz: `https://de.wikipedia.org/wiki/Redundanz_(Technik)`
- Debuggen: `https://de.wikipedia.org/wiki/Debuggen`

4. Schichte nach Dichte!

Mark hat Flaschen mit jeweils drei farbigen Flüssigkeiten, die übereinander geschichtet sind. Er weiss, dass sich die Flüssigkeiten mit geringerer Dichte immer über Flüssigkeiten mit grösserer Dichte bewegen. Nun möchte er sehen, wie es aussieht, wenn man alle farbigen Flüssigkeiten in eine Flasche gibt.

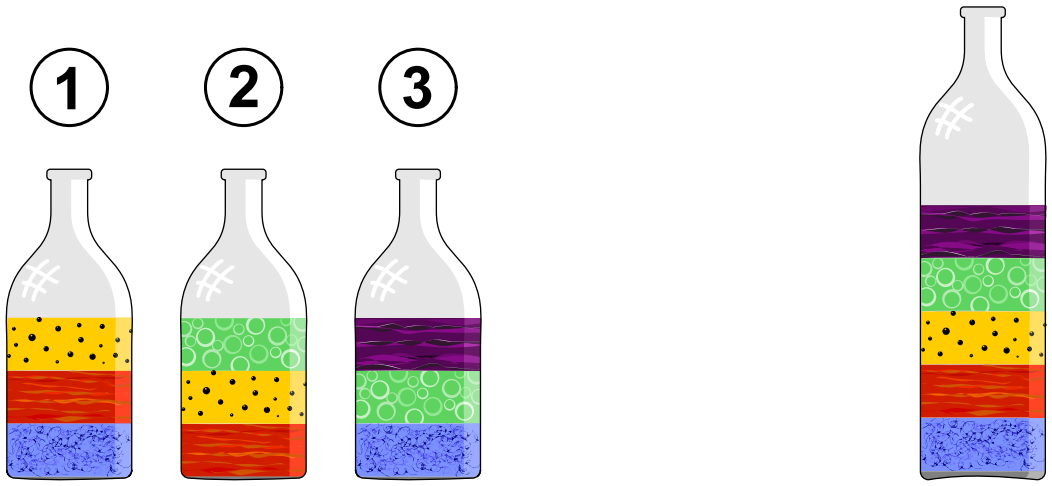


Ordne die fünf farbigen Flüssigkeiten in der Flasche, so wie sie am Ende angeordnet sind!



Lösung

Das Bild zeigt die richtige Anordnung der fünf farbigen Flüssigkeiten in der grossen Flasche.



Du findest die Reihenfolge nach folgendem Verfahren: Schritt für Schritt entfernst du in Gedanken aus den drei gegebenen Flaschen die Flüssigkeiten, die nicht auf einer anderen Flüssigkeit liegen, und gibst sie in die grosse Flasche.

Zu Beginn hat nur in den beiden Flaschen 1 und 3 blaue Flüssigkeit und dort ist sie ganz unten, sie liegt also nirgendwo auf einer anderen Flüssigkeitsschicht. Die rote Flüssigkeit ist zwar in Flasche 2 ganz unten. Aber in Flasche 1 liegt sie auf der blauen Flüssigkeit und muss deshalb eine geringere Dichte als die blaue haben. Also wird als erstes die blaue Flüssigkeit aus den Flaschen entfernt und in die grosse Flasche gegeben.

Jetzt ist die rote Flüssigkeit die einzige, die nicht auf einer anderen Flüssigkeit liegt. Sie wird aus den Flaschen 1 und 2 entfernt und in die grosse Flasche gegeben. Danach kommt die gelbe, dann die grüne und zum Schluss die violette Flüssigkeit, die die geringste Dichte hat und über der keine andere Flüssigkeit liegt.

Dies ist Informatik!

Bei der Lösung dieser Aufgabe hast du die Anordnung der Flüssigkeiten in den drei Flaschen der Aufgabenstellung ausgewertet und die Flüssigkeiten nach ihrer Dichte sortiert.

Ein Stoff hat viele messbare Eigenschaften: Siedetemperatur, Schmelztemperatur, elektrische Leitfähigkeit und eben die Dichte. In diesem Fall wurde die Dichte als Kriterium verwendet, Stoffe zu sortieren.

In vielen Computerprogrammen spielt das Sortieren von Daten eine wichtige Rolle. Das Verfahren, das bei dieser Aufgabe zur Ermittlung der Reihenfolge der Flüssigkeitsschichten verwendet wurde, nennt man *topologisches Sortieren*. Es wird zum Sortieren von Objekten angewendet, für die Beziehungen der Art Vorgänger/Nachfolger bekannt sind.

Stichwörter und Webseiten

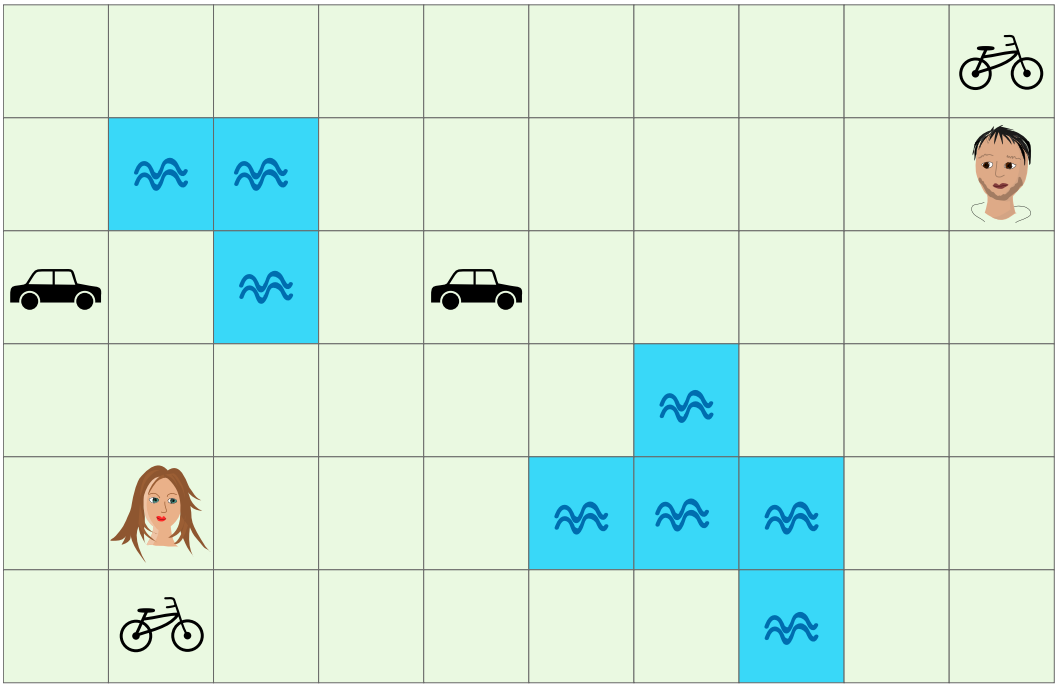
- Sortieren, Ordnung: `https://de.wikipedia.org/wiki/Sortierung`
- Topologisches Sortieren: `https://de.wikipedia.org/wiki/Topologische_Sortierung`, `https://www.ac.tuwien.ac.at/files/teaching/ss12/AD1/top_sortieren.pdf`

5. Es pressiert!

Zwei Freunde wollen sich möglichst bald treffen. Von einem Feld können sie sich zu einem benachbarten Feld links, rechts, oben oder unten bewegen.

Zu Fuss benötigen sie dafür 1 Minute. Wenn sie auf ein Feld mit einem Fahrzeug gelangen, können sie es benutzen.

Mit einem Fahrrad schaffen sie in einer Minute 2 Felder und mit einem Auto 5 Felder. Dabei sind Richtungsänderungen möglich. Wasserflächen können sie nicht überqueren.

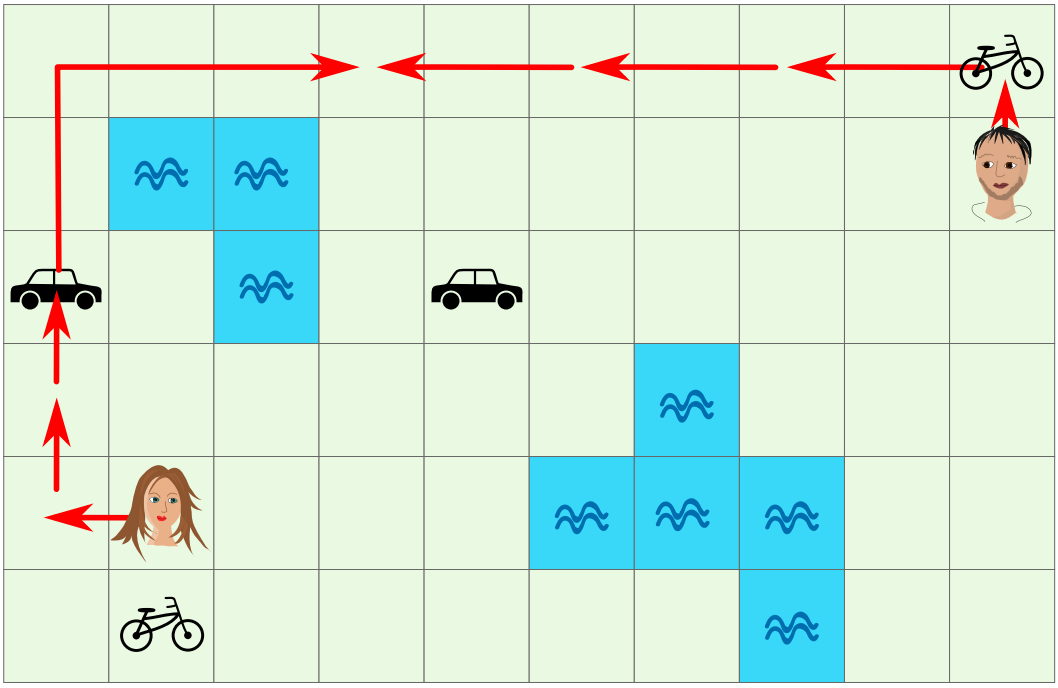


Wie viele Minuten benötigen die beiden Freunde mindestens, um sich auf einem Feld zu treffen?

A) 1 Minute
B) 2 Minuten
C) 3 Minuten
D) 4 Minuten
E) 5 Minuten
F) 6 Minuten

Lösung

Die richtige Antwort ist D) 4 Minuten. Das Bild zeigt eine Route, mit der sich die beiden Freunde in 4 Minuten auf einem Feld treffen können.



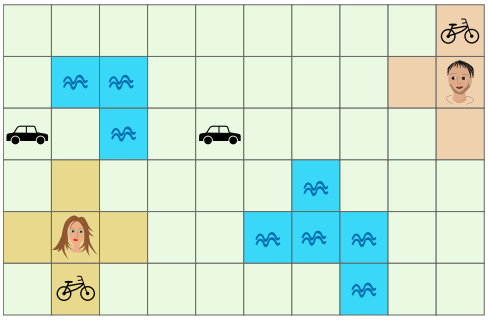
Nun muss noch bewiesen werden, dass sie sich nicht in 3 Minuten treffen können: Die beiden Freunde sind 11 Felder voneinander entfernt. In 3 Minuten können sie aber zu Fuss insgesamt nur 6Felder näher zueinander kommen. Wenn einer das Fahrad erreicht hat und der andere zu Fuss geht, dann können sie in 3 Minuten 9 Felder näher zueinander kommen, was auch nicht reicht. Selbst wenn beide zu einem Fahrrad gehen, reicht es nicht. Denn dann könnten sie in 3 Minuten 12 Felder näher zueinander kommen, die beiden Fahrräder sind aber 13 Felder voneinander entfernt.

Also bleibt nur die Option, ein Auto zu benützen. In 3 Minuten kann nur das Mädchen ein Auto erreichen. Es bleibt dann aber keine Zeit mehr, das Auto zu benutzen. Und in 3 Minuten kann der Junge kein Feld mit einem Auto erreichen.

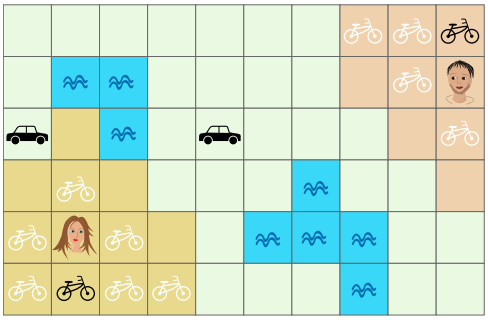
Dies ist Informatik!

Wie hast du die Aufgabe gelöst? Hast du zufällig eine schnelle Route gefunden und gehofft, dass es keine schnellere gibt? Oder hast du viele Möglichkeiten ausprobiert und dir die schnellste gemerkt?

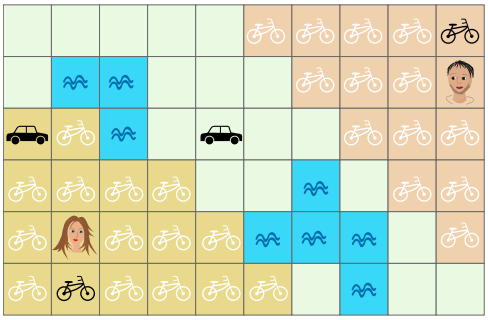
Computerprogramme, die für diese Art von Problemen entwickelt worden sind, arbeiten meist nach einem Verfahren, das man *Breitensuche* nennt. Bei dieser Aufgabe geht die Breitensuche folgendermassen:



1. Markiere alle Felder, die von den beiden Freunden in einer Minute erreicht werden können.

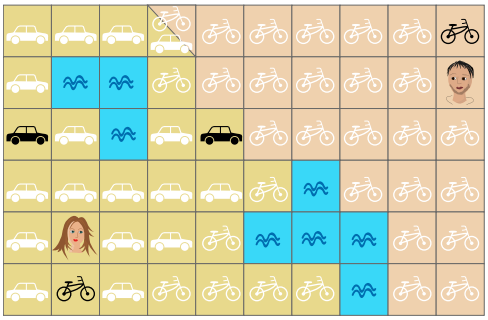


2. Markiere alle Felder, die in (höchstens) einer Minute von den Feldern aus erreicht werden können, die in Schritt 1 markiert worden sind. Notiere auch das verwendete Verkehrsmittel.



3. Markiere alle Felder, die in einer Minute von den Feldern aus erreicht werden können, die in Schritt 2 markiert worden sind.

Weil die beiden Gebiete, die du bisher markiert hast, sich nicht überlappen, können sich die Freunde nach drei Minuten noch nicht treffen.



4. Markiere nun alle Felder, die in einer Minute von den in Schritt 3 markierten Feldern erreicht werden können.

Nun überlappen sich die beiden Gebiete in einem Feld. Es kann nach 4 Minuten von dem Mädchen mit einem Auto und von dem Jungen mit einem Fahrrad erreicht werden. Navigationssysteme finden den schnellsten Weg zwischen zwei Punkten. Sie achten dabei darauf, dass die Route über geeignete Strassen und Wege verläuft - und nicht etwa querfeldein und durch Flüsse. Diese Aufgabe ähnelt dem Navigationsproblem, nur müssen hier zwei Personen zu einem gemeinsamen – anfangs noch unbekannten - Ziel geführt werden und nicht nur eine Person zu einem festen Ziel.

Weil ein Computer bei der Breitensuche systematisch vorgeht, findet er auch Lösungen, die nicht direkt ins Auge springen. Manchmal ist ein Umweg mit weniger Ampeln schneller, als die kürzeste Route

zwischen Start und Ziel. Eine Bahnverbindung mit Umsteigen kann schneller sein als eine direkte Busverbindung. In der Informatik kennt man mehrere Verfahren, um die beste Lösung zu einem Problem dieser Art zu finden. Abgesehen von der Breitensuche, die gerade beschrieben worden ist, gibt es auch einen Ansatz, den man *Branch and Bound* nennt (engl. für *Verzweigen und Begrenzen*).

Bei der Breitensuche wird jede Teillösung, die mit einer bestimmten Anzahl von Arbeitsschritten erreicht wird, berücksichtigt. Bei *Branch and Bound* verfolgt man Teillösungen nicht weiter, wenn man weiss, dass sie nicht zur optimalen Lösung führen können.

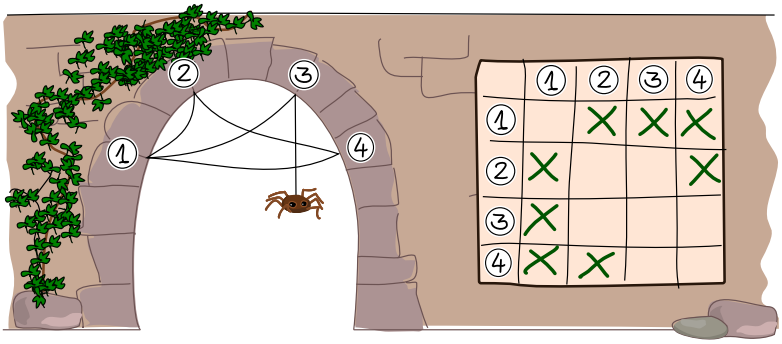
Wenn ein Problem zu komplex wird, würde es auch für den schnellsten Computer der Welt zu lange dauern, alle Möglichkeiten durchzuspielen, um die beste Lösung zu finden. In der Praxis reicht es bei einem Navigationssystem häufig aus, eine sehr gute Route zu finden, auch wenn es nicht die bestmögliche ist. (Wenn du dein Ziel in 78 Minuten erreichen kannst, macht es dir vermutlich nichts aus, wenn man es theoretisch auch in 77 Minuten erreichen könnte.)

Stichwörter und Webseiten

- Breitensuche: `https://de.wikipedia.org/wiki/Breitensuche`
- Branch and Bound Algorithmus: `https://de.wikipedia.org/wiki/Branch-and-Bound`

6. Theklas Netze

Spinne Thekla möchte möglichst viele verschiedene Netze bauen. Deshalb hat sie sich ein Verfahren ausgedacht, den genauen Aufbau ihrer Netze festzuhalten.

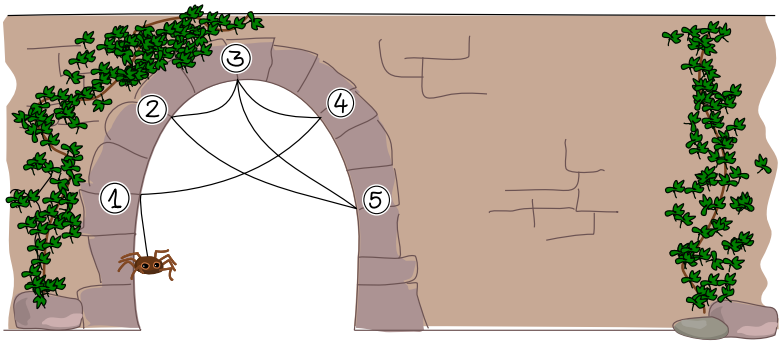


Das macht sie so: Sie nummeriert die Endpunkte des Netzes von 1 bis N und verwendet Felder in einem Raster nach folgender Regel:

- Wenn es einen Faden gibt, der Endpunkt x mit Endpunkt y verbindet, dann wird das Feld in Spalte x und Zeile y mit einem Kreuzchen markiert.

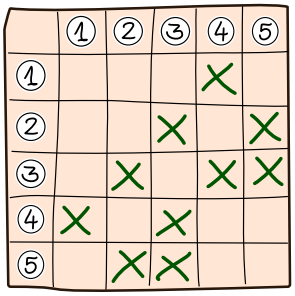
Ein Faden der Endpunkt x mit Endpunkt y verbindet, verbindet auch Endpunkt y mit Endpunkt x.

Thekla baut nun dieses Netz:

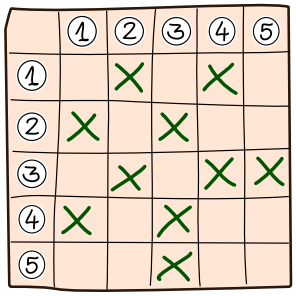


Wie hält Thekla den Aufbau dieses Netzes fest?

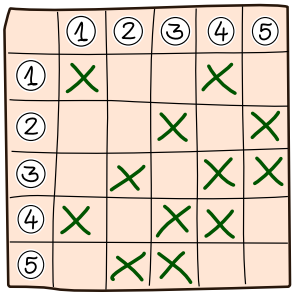
A)



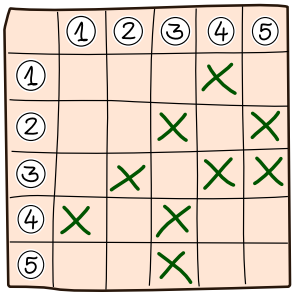
B)



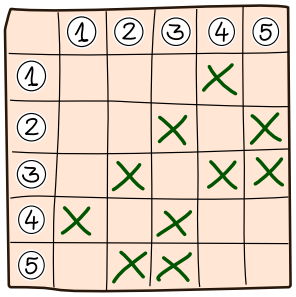
C)



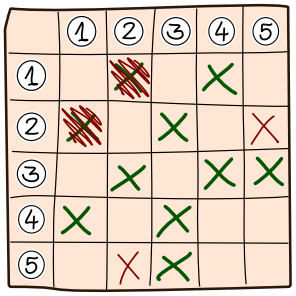
D)



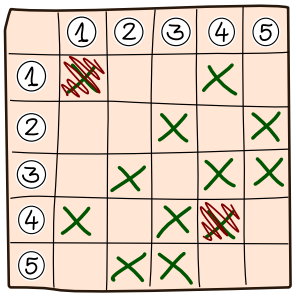
Lösung



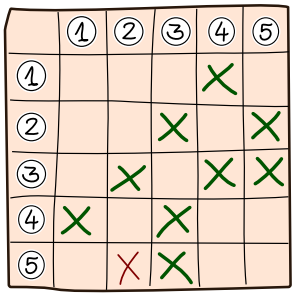
Antwort A ist richtig, denn alle Felder sind gemäss der Regel richtig markiert.



Bei Antwort B wurde eine zusätzliche Verbindung fälschlich eingezeichnet (Endpunkt 1 nach Endpunkt 2 in beide Richtungen) und eine wurde vergessen (Endpunkt 2 nach Endpunkt 5 in beide Richtungen).



Antwort C: Nach der hier vorgeschriebenen Regel können in der Diagonale von links oben nach rechts unten keine Markierung vorhanden sein. Denn das wären ja Verbindungen eines Endpunkts mit sich selbst. Dies könnte zwar in manchen Netzen durchaus erlaubt sein kann, kommt in unserem Spinnennetz aber nicht vor. Bei Antwort C hätten wir aber 2 solche Verbindungen (bei Endpunkt 1 und bei Endpunkt 4).



Antwort D: Alle Darstellungen von Netzen sollten symmetrisch sein bezüglich der Diagonalen von links oben nach rechts unten. In dieser Antwort findet sich zwar die Verbindung von Endpunkt 2 nach Endpunkt 5, die entsprechende Verbindung zurück von Endpunkt 5 nach Endpunkt 2 fehlt aber.

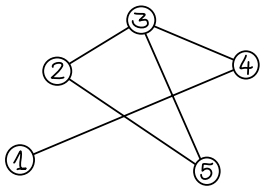
Dies ist Informatik!

Das Spinnennetz kann als *Graph* betrachtet werden, ein Konzept, das häufig in der Informatik verwendet wird.

Ein Graph besteht aus *Knoten* (den Endpunkten des Netzes) und *Kanten* (den Fäden zwischen den Endpunkten). Graphen werden zum Beispiel auch verwendet, um Objekte und die Beziehungen zwischen Objekten darzustellen. Ein Graph könnte beispielsweise zeigen, wie Personen in sozialen Medien befreundet sind, oder Flüge zwischen Ländern.

In dieser Aufgabe wird gezeigt, wie man die Struktur eines Spinnennetzes in einem Raster speichern kann. Gewisse Eigenschaften, wie zum Beispiel das genaue Aussehen des Netzes, gehen dabei verloren.

In vielen Fällen ist man aber nicht an den genauen geometrischen Eigenschaften eines Netzes interessiert, sondern nur an seiner Struktur. Die wesentlichen Informationen bleiben erhalten: Wie viele Knoten gibt es? Und zwischen welchen Knotenpaaren gibt es eine Kante?



	1	2	3	4	5
1	0	0	0	1	0
2	0	0	1	0	1
3	0	1	0	1	1
4	1	0	1	0	0
5	0	1	1	0	0

Die vorgestellte Möglichkeit ist nur eine von vielen Möglichkeiten, die Struktur eines Netzes festzuhalten. Die Methode ist nicht sehr sparsam, denn es werden für jede Verbindung beide Richtungen gespeichert, was nicht notwendig wäre, und auch die freien Diagonalenfelder wären gar nicht notwendig. Dafür weist dieses Verfahren den Vorteil auf, dass Darstellungsfehler teilweise erkannt werden können. Antwort C und Antwort D konnten zum Beispiel als falsch erkannt werden, ohne auf das Netz Bezug zu nehmen.

Die vorgestellte Darstellungsform wird *Adjazenzmatrix* genannt.

Stichwörter und Webseiten

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

7. Frucht auf Frucht

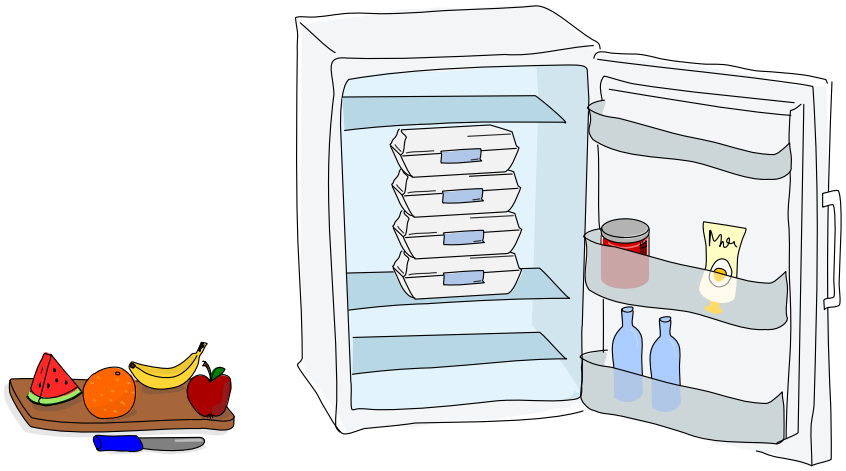
Paps, Mams, Dorie und Ron Biber packen abends für den nächsten Tag vier Frühstücksboxen, jede mit einer anderen Frucht: Apfel, Banane, Orange und Wassermelone. Die Boxen werden im Kühlschrank aufeinander gestapelt. Morgens sind die Bibers noch sehr müde und nehmen sich beim Verlassen des Baus einfach die oberste Box, ohne sie genauer anzuschauen.

Man weiss nicht genau, in welcher Reihenfolge die Bibers den Bau verlassen, aber auf jeden Fall geht Mams vor Dorie und Paps immer als Letzter.

Die Familienmitglieder mögen unterschiedliche Früchte. Die Tabelle gibt an, was jedes Familienmitglied mag.

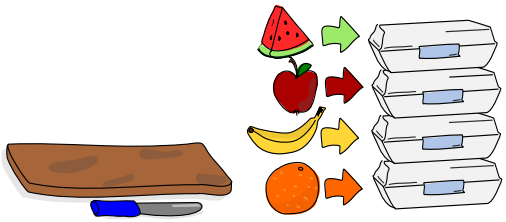
Paps	—	—	✓	—
Mams	✓	—	✓	✓
Dorie	✓	✓	✓	—
Ron	✓	✓	—	✓

Packe die Früchte so in die Boxen, dass alle Bibers eine Box mit Früchten nehmen, die sie mögen.



Lösung

Es gibt nur eine Möglichkeit, die Früchte so zu verteilen, dass alle garantiert etwas bekommen, das sie mögen:



Paps mag nur Orangen und geht als Letzter. Also kommen Orangen in die unterste Box. Ron geht als erster, zweiter oder dritter. Mams geht ausserdem vor Dorie, daher ist die Reihenfolge klar, wenn man weiss, wann Ron den Bau verlässt. Folgende drei Reihenfolgen sind möglich:

1.	Mams	Mams	Ron
2.	Dorie	Ron	Mams
3.	Ron	Dorie	Dorie
4.	Paps	Paps	Paps

Mams, Dorie und Ron könnten also alle als zweite gehen. In die zweite Box muss daher eine Frucht hinein, die alle drei mögen, und das erfüllt nur der Apfel. Für die oberste Box bleiben also nur Bananen und Wassermelone. Da Mams Bananen nicht mag, müssen wir hier die Wassermelone zuordnen. Damit bleiben die Bananen für die dritte Box.

Dies ist Informatik!

Die richtige Reihenfolge ist in vielen Bereichen der Informatik sehr relevant: Viele Berechnungen erfordern Zwischenergebnisse, die zunächst ermittelt werden müssen, bevor man zum Endresultat kommt. Werden die Rechenschritte auf unterschiedlichen Computern durchgeführt, könnten ohne sorgfältige Planung sogenannte *Deadlocks* entstehen. Das sind Situationen, in denen zwei oder mehr Computer aufeinander warten und auf diese Weise das Programm nie zum Ende kommt. Die falsche Reihenfolge führt meistens aber einfach zu Fehlern (wie bei den Bibers zu Missmut über die erwischten Früchte). Wenn zum Beispiel etwas mit der Formel $Z \leftarrow (A + B) \cdot (A - B)$ berechnet werden soll, kann man das in folgende Schritte eines Programms aufteilen:

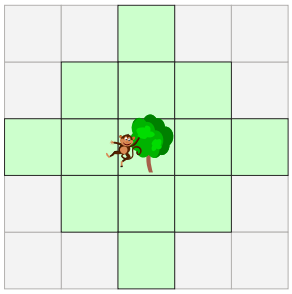
Eingabe A
Eingabe B
Berechne $X \leftarrow A + B$
Berechne $Y \leftarrow A - B$
Berechne $Z \leftarrow X \cdot Y$

Versucht man aber z. B. den Rechenschritt $Z \leftarrow X \cdot Y$ auszuführen, bevor man X berechnet hat, führt dies zu einem Fehler und dem Abbruch des Programms. Oder es wird ein Standardwert für X verwendet, was meistens zu einem falschen Resultat führt. Beim Programmieren ist also die Reihenfolge, in der die Anweisungen durchgeführt werden, relevant.

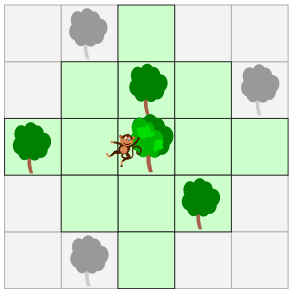
Stichwörter und Webseiten

- Deadlock: `https://de.wikipedia.org/wiki/Deadlock_(Informatik)`

8. Kletteräffchen Koko



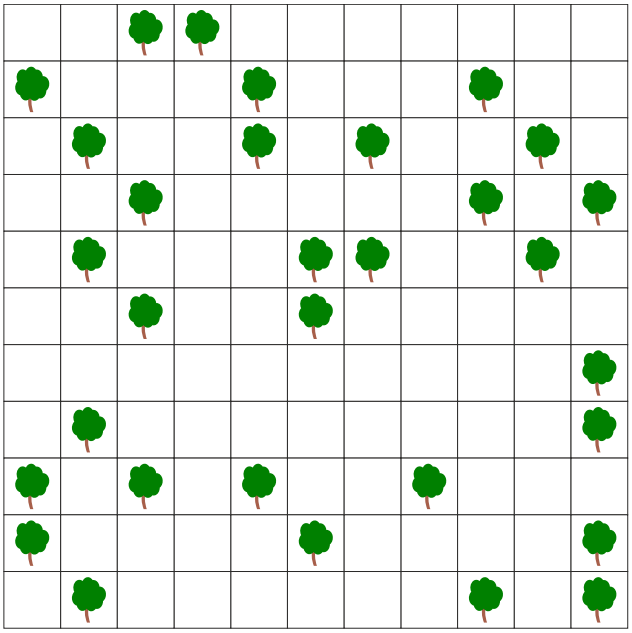
Kletteräffchen Koko kann von einem Baum so weit springen, wie es der grüne Bereich



In folgendem Beispiel erreicht Koko die farbigen Bäume mit einem Sprung. Mit zwei Sprüngen sind auch die beiden grauen Bäume oben erreichbar, nicht aber der graue Baum unten.

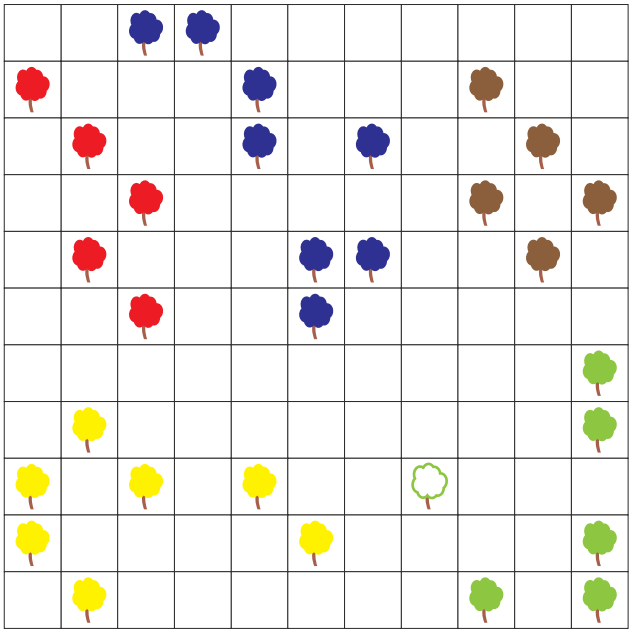
Es gibt Gruppen von Bäumen, zwischen denen sich Koko mit mehreren Sprüngen beliebig bewegen kann, ohne jemals den Boden zu berühren.

Markiere alle Bäume der grössten solchen Gruppe.



Lösung

Im Bild unten haben zwei Bäume dieselbe Farbe, wenn Koko vom einen zum anderen gelangen kann, ohne den Boden zu berühren.

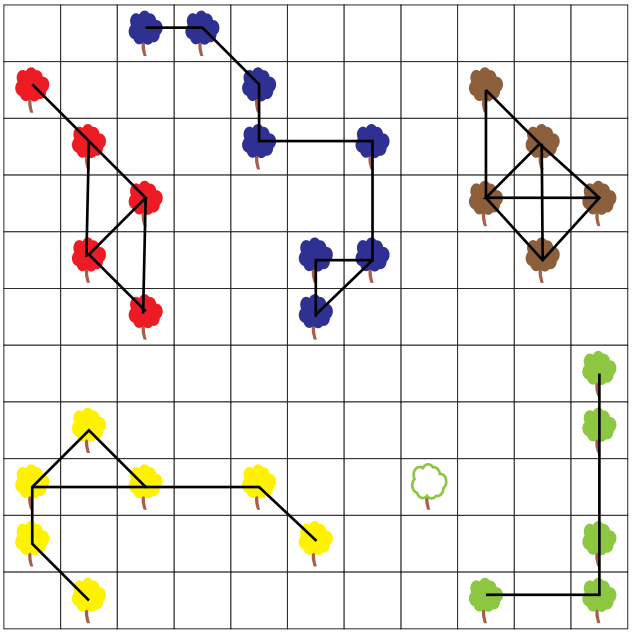


Wir sehen, dass die blaue Baumgruppe mit ihren acht Bäumen die grösste ist.

Dies ist Informatik!

Wenn Koko von einem Baum direkt zum nächsten springen kann, sind sie quasi miteinander verbunden. Wir können dies als eine Linie zwischen den Bäumen darstellen, so wie unten gezeigt. Wir haben also einen Graphen mit Bäumen als Knoten und Kanten zwischen verbundenen Bäumen. Koko kann genau dann springend von einem Baum zu einem anderen gelangen, wenn die Kanten einen Weg zwischen den beiden Bäumen bilden.

Wir nennen eine Gruppe von Knoten *zusammenhängend*, wenn sie alle über Kanten miteinander verbunden sind. Wenn wir eine solche Gruppe nicht mehr grösser machen können, ohne den Zusammenhang zwischen ihnen zu verlieren, dann sprechen wir von einer *Zusammenhangskomponente*. Ein Graph lässt sich eindeutig in solche Zusammenhangskomponenten aufteilen, unten sind sie farbig markiert.



Eine Zusammenhangskomponente lässt sich einfach bestimmen, indem man bei einem beliebigen Knoten beginnt und dann alle über Kanten erreichbaren Knoten sucht.

Stichwörter und Webseiten

- Zusammenhangskomponenten,
  `https://de.wikipedia.org/wiki/Zusammenhang_(Graphentheorie)`

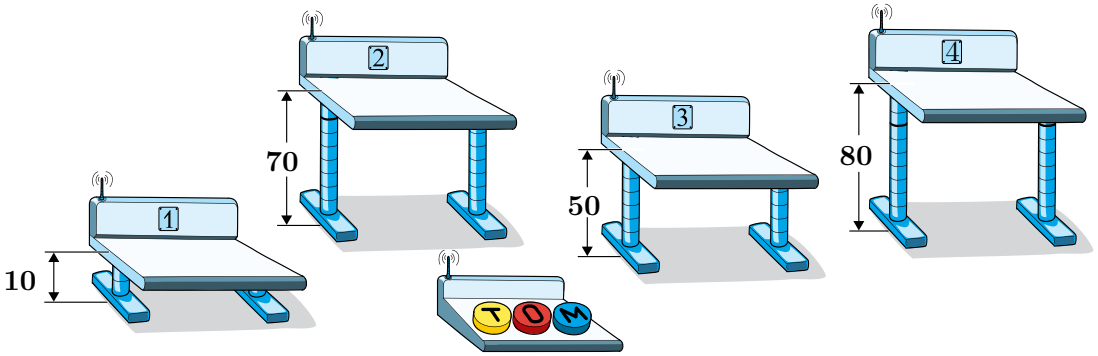
9. Verflixte Pulte

Im Schulzimmer hat es Pulte mit elektrisch einstellbarer Höhe. Für den Unterricht sollten alle Pulte auf die Höhe 60 cm eingestellt sein. Mit den Tasten T, O und M einer Fernbedienung kann die Höhe der Pulte verändert werden. Jemand hat mit der Fernbedienung gespielt und sie umprogrammiert. Jetzt funktionieren die drei Tasten folgendermassen:

- T erhöht jedes der Pulte 1, 2 und 3 um jeweils 10 cm.
- O senkt jedes der Pulte 2, 3 und 4 um jeweils 10 cm.
- M erhöht jedes der Pulte 1, 3 und 4 um jeweils 10 cm.

Diese Aktionen werden jedes Mal ausgeführt, wenn die Taste gedrückt wird.

Momentan sind die Höhen der Pulte 1, 2, 3 und 4 auf 10 cm, 70 cm, 50 cm und 80 cm eingestellt:



Wie kann die Höhe für alle vier Pulte auf 60 cm eingestellt werden?

A) Drücke 4 × T, 5 × O und 1 × M.
B) Drücke 5 × T, 1 × O und 0 × M.
C) Drücke 3 × T, 4 × O und 2 × M.
D) Drücke 2 × T, 4 × O und 6 × M.

Lösung

Die korrekte Antwort ist C) Drücke 3 × T, 4 × O und 2 × M. Bei der Fernbedienung stellst du fest, dass alle drei Tasten mit jedem Mal Drücken die Höhe um 10 cm verändern, also immer um denselben Betrag. Zwei der Tasten lassen die Pulte höher werden (T und M) und nur eine Taste senkt die Pulte ab (O). Zudem verändern alle drei Tasten die Höhe von jeweils drei Pulten, es bleibt also immer ein Pult, dessen Höhe unverändert bleibt. Die Taste O hat keinen Einfluss auf Pult 1, wir können also Pult 1 gar nicht nach unten fahren.

Pult 1 ist um 50 cm zu tief. Daraus schliessen wir, dass wir genau 5 Mal die Taste T oder M drücken müssen (die Anzahl der Tastendrücke auf T und auf M muss zusammengerechnet genau 5 sein). Das kann man als Gleichung ausdrücken: $T + M = 5$. Damit können wir Lösung D) ausschliessen, weil dort $T + M = 8$ gilt. Nach der Tastenfolge von Lösung D) hätte das Pult 1 die Höhe $10 + 20 + 60 = 90$ cm, nämlich die Ausgangshöhe 10 cm plus $2 \cdot 10$ cm für T plus $6 \cdot 10$ cm für M.

Pult 2 ist 10 cm zu hoch. M hat keinen Einfluss auf Pult 2. Somit gilt für die richtige Lösung $T - O = -1$. Damit können wir die Lösung B) ausschliessen, denn damit hätte das Pult 2 am Ende die Höhe $70 + 50 - 10 = 110$.

Pult 3 ist 10 cm zu tief, also gilt: $T - O + M = 1$. Damit können die Lösungen A) und B) ausgeschlossen werden. Bei A) wäre die Höhe von Pult 3 am Ende wieder dieselbe, nämlich $50 + 40 - 50 + 10 = 50$ cm; bei B) wäre die Höhe von Pult 3 am Ende $50 + 50 - 10 = 90$ cm. Nun sind alle Lösungen ausgeschlossen bis auf Lösung C).

Es muss aber noch geprüft werden, ob Lösung C) auch für Pult 4 die richtige Höhe ergibt. Pult 4 ist 20 cm zu hoch und T hat keinen Einfluss auf die Höhe von Pult 4. Also muss zwei Mal O gedrückt werden und für jedes Drücken von M braucht es ein zusätzliches Drücken von O. Mit der Tastenfolge von Lösung C) ist die Höhe von Pult 4 am Ende $80 - 40 + 20 = 60$ cm.

Da wir schon vorher für die Pulte 1, 2 und 3 festgestellt hatten, dass Lösung C) funktioniert, sind wir sicher, dass diese Lösung funktioniert.

Alternativ kann die Lösung mit vier linearen Gleichungen gesucht werden. Für jedes Pult schreibt man mit einer Gleichung auf, welche Tasten die Höhe des Pultes verändern und was die gesuchte Höhenveränderung ist. Beispielsweise verändert sich die Höhe von Pult 1 nur mit T und M und die gewünschte Höhenanpassung ist 50 cm, was man mit 5 Tastendrücken erreichen kann (weil es pro Tastendruck 10 cm sind).

Da es vier Pulte und drei Tasten sind, ergeben sich vier lineare Gleichungen mit drei Unbekannten:

$$\begin{aligned} T + M &= 5 \\ T - O &= -1 \\ T - O + M &= 1 \\ -O + M &= -2 \end{aligned}$$

Wenn man die dritte Gleichung von der ersten abzieht, erhält man $O = 4$. Eingesetzt in die zweite Gleichung erhalten wir $T = 3$. Nur für $M = 2$ sind alle Gleichungen erfüllt. Somit ist es die einzige Lösung.

Dies ist Informatik!

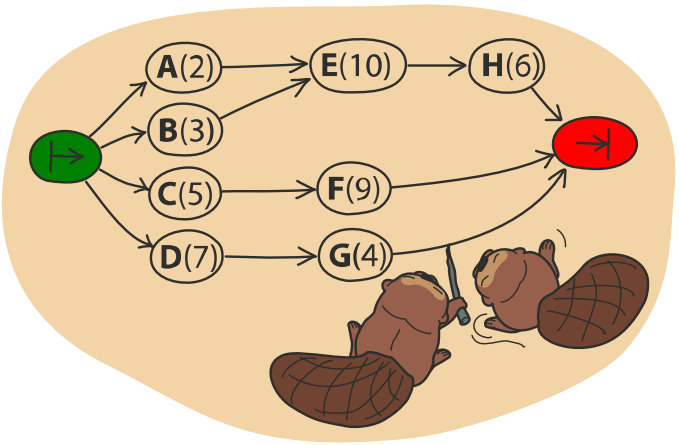
Dies ist eine typische Aufgabe aus dem Bereich der *diskreten Optimierung*, genauer spezifiziert, der *linearen Programmierung*. Diese Aufgabe ist gegeben durch eine Menge von Einschränkungen. In diesem Spezialfall kann man sie alle als lineare Gleichungen formulieren. Die Zielsetzung ist typisch für die Informatik. Man sucht eine Folge von Aktionen, die zu einem vorgegebenen Ziel führen. Man könnte sogar die ganze Aufgabe als die Suche nach einem Weg in einem vierdimensionalen Raum mit drei erlaubten Bewegungsaktionen beschreiben, nämlich vom Punkt $(10, 70, 50, 80)$ zum Punkt $(60, 60, 60, 60)$. In dieser Aufgabe gibt es nur eine Lösung, aber solche Aufgaben haben oft viele Lösungen, was eine Optimierung als Zielsetzung ermöglicht. Dann sucht man das Minimum der linearen Funktion $T + M + O$.

Stichwörter und Webseiten

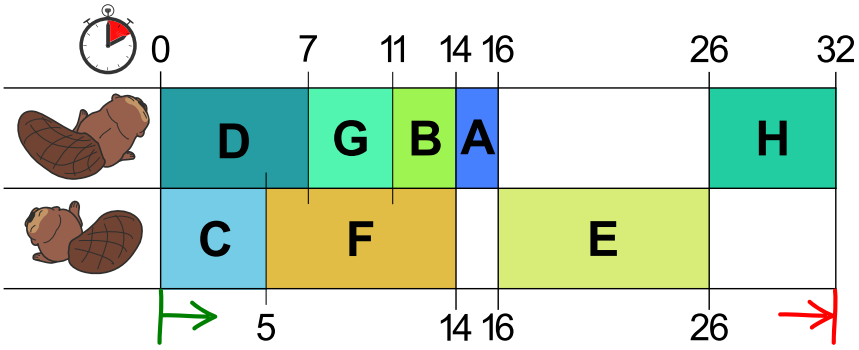
- Diskrete Optimierung, ganzzahlige lineare Optimierung:
  `https://de.wikipedia.org/wiki/Ganzzahlige_lineare_Optimierung`
- Erreichbarkeit in gerichteten Konfigurationsgraphen:
  `https://de.wikipedia.org/wiki/Erreichbarkeitsproblem_in_Graphen`

10. Biber-Arbeit

Das Bauen eines Biberdamms lässt sich in mehrere Teilaufgaben zerlegen (Bäume fällen, Äste entfernen, Stämme zum Wasser transportieren, usw.). Das Bild rechts zeigt alle 8 Teilaufgaben A, B, C, D, E, F, G, H, jeweils mit der Anzahl der Stunden, die man zu ihrer Erledigung braucht. Die Teilaufgaben sind nicht ganz unabhängig voneinander: Ein Pfeil von X nach Y bedeutet, dass Teilaufgabe X vollständig erledigt sein muss, bevor man mit Teilaufgabe Y anfängt.

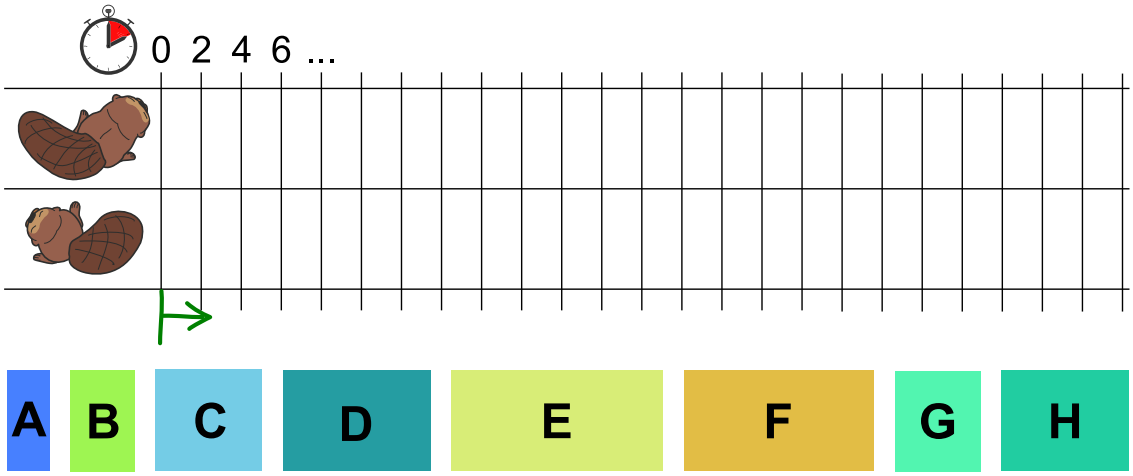


Biberin Ulla will dem Biber Otso helfen, den Damm schneller zu bauen. Sie teilen die Teilaufgaben untereinander auf und erstellen folgenden Arbeitsplan, der die Abhängigkeiten aus dem Bild oben erfüllt.



Damit würde der Damm in 32 Stunden fertig. Das geht aber schneller!

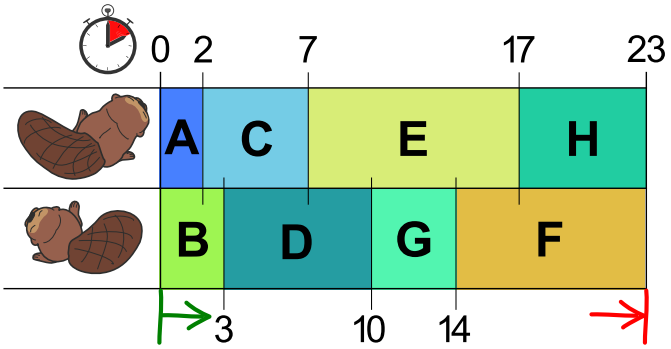
Erstelle einen Arbeitsplan, mit dem der Damm in möglichst kurzer Zeit fertig wird.



Lösung

Im Arbeitsplan aus der Aufgabe hat der erste Biber eine lange Pause (10 Stunden), und der zweite Biber hat insgesamt 8 Stunden lang Leerlauf. Die beiden wären schneller fertig, wenn sie ständig arbeiteten.

Man kommt zu einem insgesamt schnelleren Arbeitsplan, wenn man darauf achtet, dass die beiden grössten Aufgaben E(10) und F(9) nicht vom selben Biber ausgeführt werden. Hier ist ein möglicher Arbeitsplan, der mit 23 Stunden auskommt. Schneller geht es nicht, denn die beiden Biber arbeiten ohne Pause.



Dies ist Informatik!

Menschen sind ungeduldig, und deshalb wird oft verlangt, dass Arbeiten möglichst schnell erledigt werden. Wenn eine Arbeit aufgeteilt werden kann (ob nun unter mehreren Menschen, Maschinen oder Bibern), dann spielt die Verteilung der Teilaufgaben auf die Arbeitenden für die Schnelligkeit eine wichtige Rolle. Genauso müssen auch Computer ihre Berechnungsaufgaben oft geschickt auf verschiedene Prozessorkerne aufteilen.

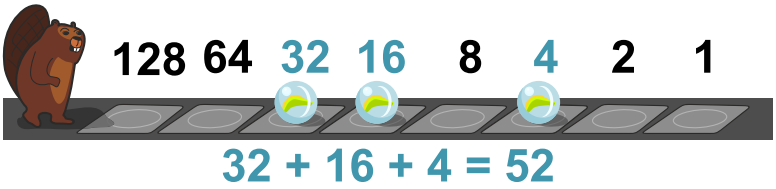
Für solche *Scheduling-Probleme* gibt es viele verschiedene, von der Informatik gut untersuchte Strategien. Der erste Arbeitsplan in dieser Biberaufgabe wurde so erstellt, dass unter den anstehenden Teilaufgaben, jene mit der längsten Dauer einem gerade arbeitslosen Biber zugeteilt wurde – in diesem Fall eine schlechte Strategie. Oft funktioniert es besser, wenn kurze Teilaufgaben zuerst erledigt werden: Die Strategie *Shortest-Job-Next* (engl. für kürzeste Teilaufgabe zuerst) minimiert zudem auch die durchschnittliche Wartezeit bis zur Fertigstellung pro Teilaufgabe.

Stichwörter und Webseiten

- Scheduling: `https://de.wikipedia.org/wiki/Scheduling`
- Shortest-Job-Next: `https://de.wikipedia.org/wiki/Shortest-Job-Next`

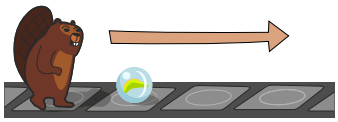
11. Murmelzahlen

Die Biber haben eine besondere Art, Zahlen darzustellen.



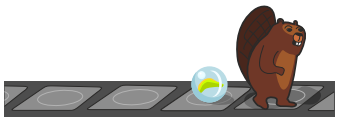
Die verschiedenen Felder haben verschiedene Gewichtungen und eine Murmel auf dem Feld bestimmt, dass der Wert übernommen wird. Im Beispiel oben wird die Zahl 52 dargestellt.

Der Biber bewegt sich Feld für Feld von links nach rechts über ein Band. Auf manchen Feldern des Bands können Murmeln liegen.



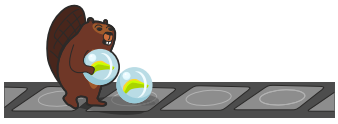
Immer, wenn der Biber auf ein Feld mit einer Murmel kommt und er die Hände frei hat, hebt er die Murmel auf und trägt sie dann mit sich.

Beim ersten freien Feld legt er die Murmel wieder ab.



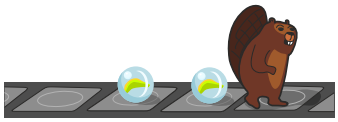
Der Biber kann immer nur eine Murmel tragen und auf jedem Feld hat nur eine Murmel Platz.

Trägt der Biber schon eine Murmel, wenn er ein Feld mit einer anderen Murmel erreicht, ...



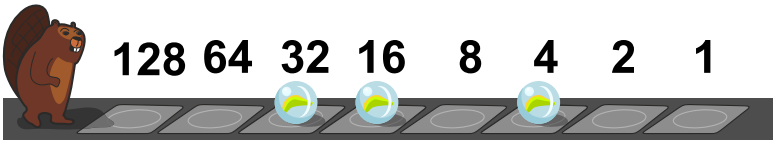
... dann geht er an ihr vorbei ...

... und legt seine Murmel auf das nächste freie Feld.



Danach kann er die nächste Murmel wieder aufheben.

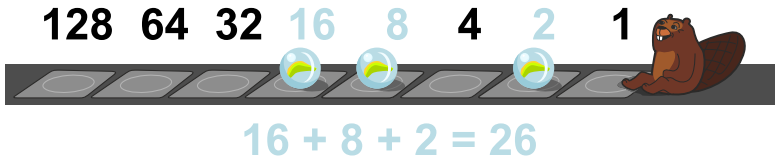
Welche Zahl wird durch die Murmeln dargestellt, wenn der Biber den Bereich überquert hat?



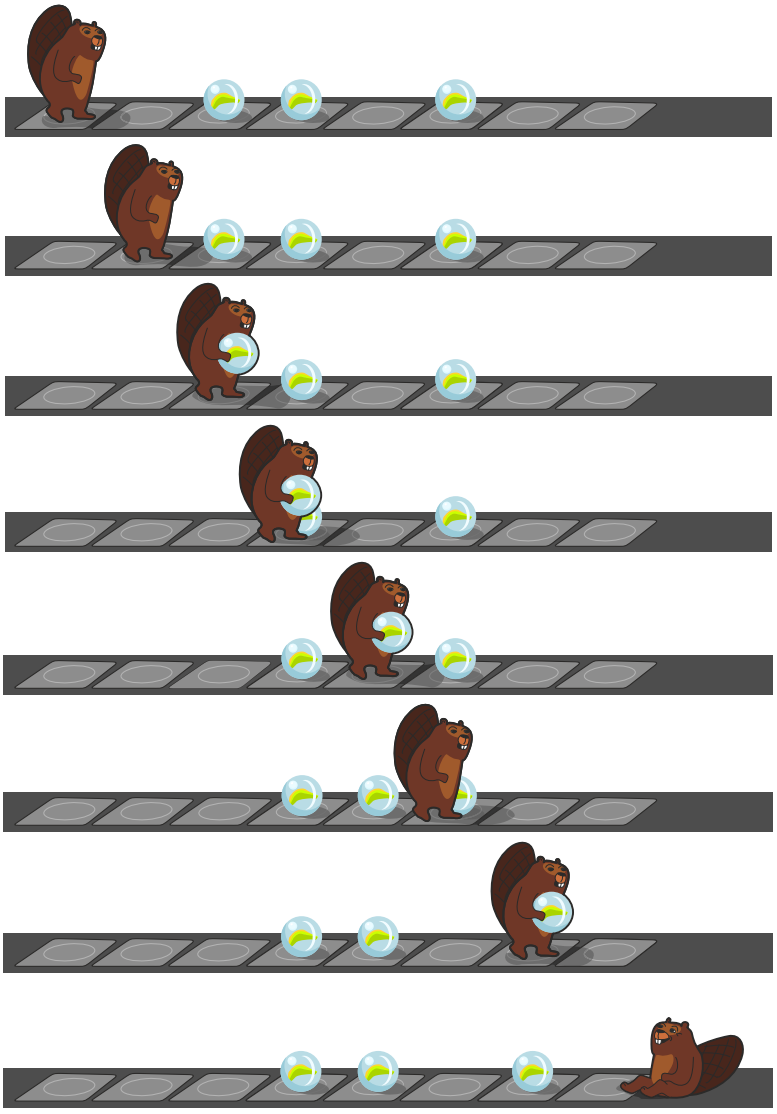
A) 10
B) 26
C) 28
D) 104

Lösung

Die richtige Anwort ist B) 26



Die folgende Abbildung zeigt den Ablauf:



Dies ist Informatik!

In der Informatik führen oft auch relativ einfache Operationen zu interessanten Ergebnissen. Diese Aufgabe ist ein gutes Beispiel dafür. Das Vorgehen des Bibers ist ein *Algorithmus*. Er beruht darauf, dass der Biber 2 verschiedene Zustände einnehmen kann (Murmel tragend oder nicht) und dass er auf seinem Weg 2 verschiedene Arten von Feldern vorfinden kann (belegte und leere).

Insgesamt ist das Resultat nach Ausführen des Algorithmus genau so, als hätte man jede Murmel auf dem Band um eins nach rechts verschoben. In der Zahlendarstellung der Biber entspricht dies einer Division durch 2. Würde der Biber von rechts nach links über das Band marschieren, würde die Zahl hingegen mit 2 multipliziert. Wenn in einer Reihe von Nullen und Einsen alles gleich viel nach links oder nach rechts vorschoben wird, bezeichnet man das in der Information oft als *Bitshift*.

So einfach das Beispiel in dieser Biberaufgabe ist, enthält es doch einige der wesentlichen Elemente einer *Turingmaschine*. Eine Turingmaschine (benannt nach dem Mathematiker Alan Turing) ist ein spezieller, sehr simpel strukturierter Computer. Eine Turingmaschine kann grundsätzlich alle Algorithmen ausführen, die ein herkömmlicher Computer ausführen kann. In der Praxis werden trotzdem keine Turingmaschinen als Computer verwendet, denn wir können Computer bauen, die zwar komplizierter, aber viel effizienter sind. Turingmaschinen sind vor allem in der Theorie nützlich. Durch ihre simple Struktur kann man relativ einfach Aussagen über Turingmaschinen beweisen. Und wenn man beweisen kann, dass eine Aufgabe für Turingmaschinen nicht lösbar ist, dann kann keiner unserer Computer sie lösen.

Eine Turingmaschine besteht aus:

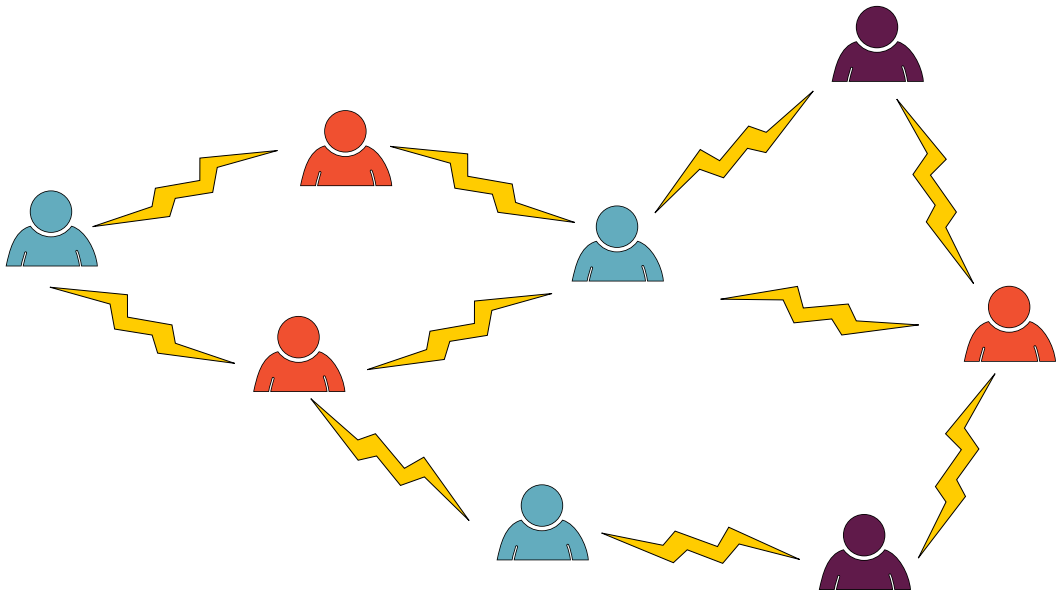
- Einem beliebig langen *Band*, bestehend aus einzelnen *Feldern*. In jedem Feld kann ein *Symbol* stehen. Das sind bei unserem Beispiel die Felder, über die sich der Biber bewegt.
- Einer endlichen Menge von *Symbolen*. Oft benützt man nur `0` und `1` als Symbole. In unserem Beispiel steht eine Murmel für `1` und eine freie Stelle für `0`.
- Einem Lese-Schreib-Kopf, der sich auf dem Band in beide Richtungen bewegen und dabei die Symbole auf dem Band lesen und auch neue Symbole schreiben kann. In unserem Beispiel hat der Biber die Rolle des Lese-Schreib-Kopfs.
- Einer endlichen Menge von sogenannten *Zuständen*. Das Verhalten des Lese-Schreib-Kopfs kann sich je nach Zustand ändern. In unserem Fall gibt es nur zwei Zustände, nämlich «Murmel tragend» und «nicht Murmel tragend».
- Einer Menge von Regeln: Was passiert, abhängig vom Zustand, wenn ein bestimmtes Symbol vom Band gelesen wird? Mögliche Aktionen sind: ein Wechseln des Zustands, das Schreiben eines neuen Symbols auf das Band und das Bewegen des Lese-Schreib-Kopfs um ein Feld nach links oder rechts.

Stichwörter und Webseiten

- Turingmaschine: `https://de.wikipedia.org/wiki/Turingmaschine`
- Bitweiser Operator, Bitweise Verschiebungen: `https://de.wikipedia.org/wiki/Bitweiser_Operator#Bitweise_Verschiebungen`

12. Teamwork

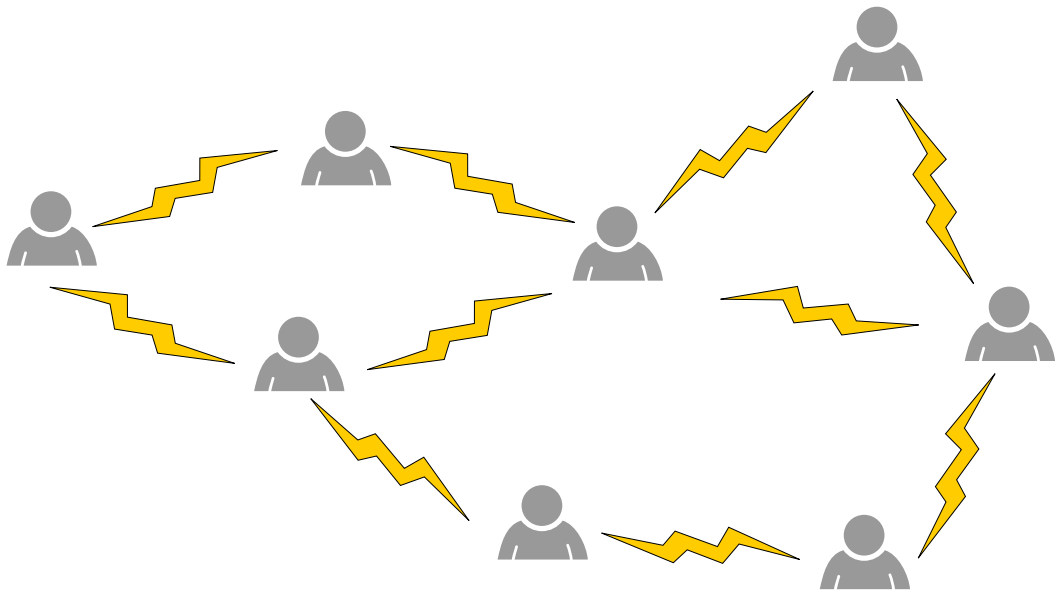
Für ein Projekt sollst du acht Personen in Gruppen aufteilen. Zwischen zwei Personen steht ein Blitz, wenn sie nicht zusammenarbeiten wollen. Dann möchtest du sie nicht derselben Gruppe zuordnen.



Mit den Abneigungen im Beispiel oben ist eine Aufteilung in drei Gruppen (rot, blau, violett) möglich. Zwischen zwei Personen derselben Farbe steht also nirgends ein Blitz.

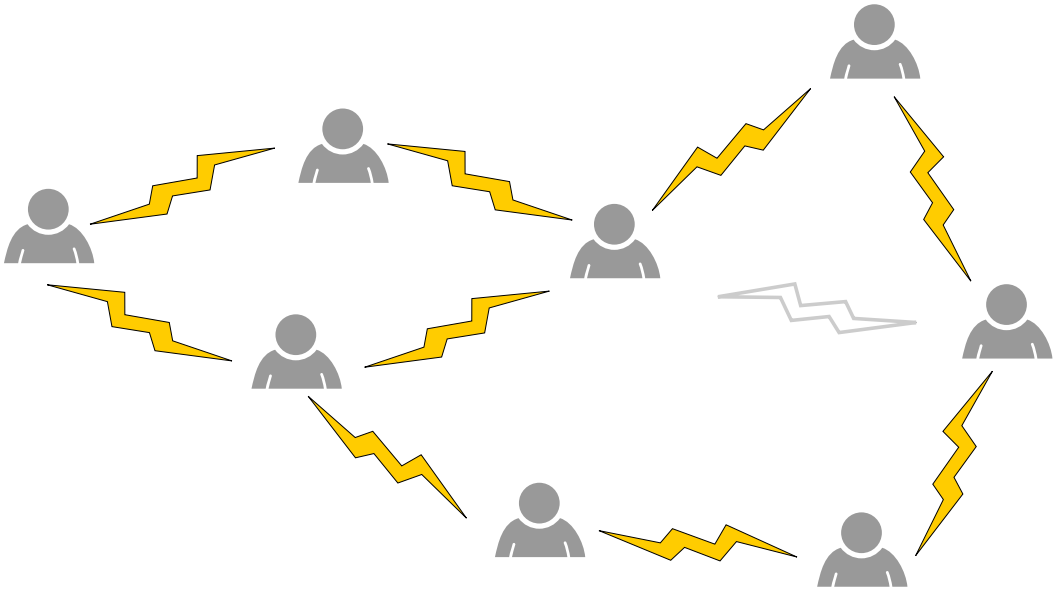
Wenn du die richtigen beiden Personen zur Zusammenarbeit überzeugst (also einen Blitz entfernst), dann ist sogar eine Aufteilung in nur zwei Gruppen (nur zwei Farben) möglich.

Entferne den richtigen Blitz.

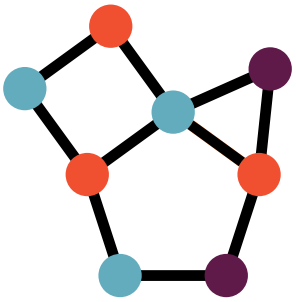


Lösung

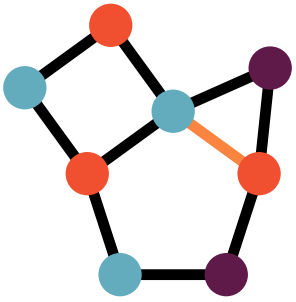
Die richtige Lösung ist:



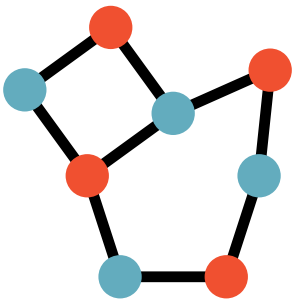
Wir stellen die Situation noch etwas abstrakter als sogenannten *Graphen* dar, mit den Personen als *Knoten* (Kreise) und den Blitzen als *Kanten* (Linien).



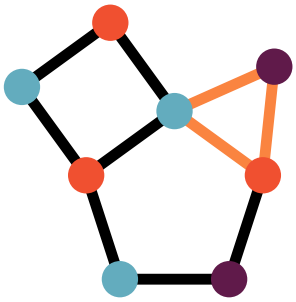
Die einzig mögliche Option ist die orange markierte Kante.



Nach dem Löschen dieser Kante, können wir die Knoten mit zwei Farben einfärben.

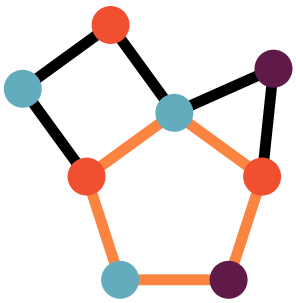


Jede Farbe steht für eine Gruppe. Wir sehen, dass nirgends zwei Personen der gleichen Gruppe die Zusammenarbeit verweigern: Benachbarte Knoten haben überall verschiedene Farben.



Um zu sehen, dass das Löschen dieser Linie die einzig mögliche Option ist, betrachten wir zunächst das orange markierte Dreieck.

Wenn eine Linie ausserhalb dieses Dreiecks gelöscht wird, benötigen wir immer noch drei Farben allein schon für die drei Knoten des Dreiecks.



Betrachten wir nun das orange markierte Fünfeck:

Wenn eine Kante ausserhalb dieses Fünfecks gelöscht wird, bleibt dieses intakt und es ist unmöglich, das Fünfeck nur mit zwei Farben zu färben: Wenn wir das Fünfeck im Uhrzeigersinn durchlaufen, müssen wir zwischen den beiden Farben abwechseln. Wenn wir jedoch den letzten Knoten erreichen, hat er dieselbe Farbe wie der erste Knoten, da die Anzahl der Kreise im Fünfeck ungerade ist, wie schon beim Dreieck.

Die einzige mögliche Lösung ist es somit, die gemeinsame Kante Dreiecks und des Fünfecks zu löschen.

Dies ist Informatik!

Viele im Alltag auftretende Probleme können als sogenannte *Graphfärbungsprobleme* formuliert werden. In unserer Biberaufgabe repräsentieren die *Knoten* eines *Graphen* die Personen und eine *Kante* zwischen zwei Personen zeigt, dass sie sich weigern, in einer Gruppe zusammenzuarbeiten. Wenn wir die Knoten mit k Farben einfärben, kann dies als Zuordnung der Personen zu je einer von k Gruppen verstanden werden. Eine solche Färbung heisst *zulässig*, wenn für jedes Paar von zwei Knoten, die direkt durch eine Kante verbunden sind, diese beiden Knoten unterschiedliche Farben haben. In unserem Fall ist eine Färbung also genau dann zulässig, wenn in jeder Gruppe alle Personen zusammenarbeiten. Eine Kante wird als *kritisch* bezeichnet, wenn das Löschen dieser Kante eine Färbung des Graphen mit weniger Farben als zuvor ermöglicht. (Dabei dürfen alle Knoten des Graphen umgefärbt werden.) Für uns ist eine Kante also genau dann kritisch, wenn nach Überzeugung der entsprechenden beiden Personen zur Zusammenarbeit eine Aufteilung in weniger Gruppen möglich ist.

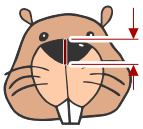
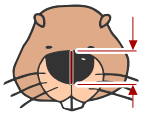
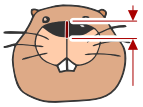
Stichwörter und Webseiten

- Graphentheorie: `https://de.wikipedia.org/wiki/Graphentheorie`
- Färbung eines Graphen: `https://de.wikipedia.org/wiki/Färbung_(Graphentheorie)`
- Kritischer Graph: `https://de.wikipedia.org/wiki/Kritischer_Graph`

13. Zählen durch Nicken

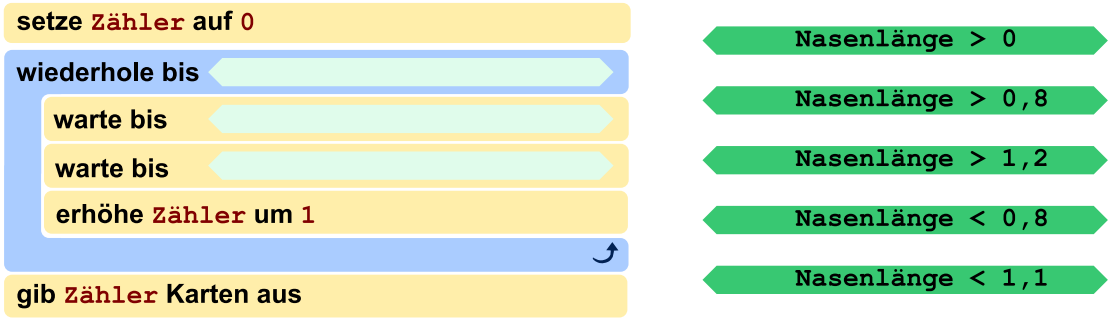
Ein neuer Eintrittskartenautomat soll so funktionieren: Ein Kunde nickt so oft mit dem Kopf – senkt also den Kopf und schaut dann wieder geradeaus – wie viele Karten er kaufen möchte. Danach hebt der Kunde den Kopf, und dann gibt der Automat die Karten aus.

Der Automat hat dazu eine Kamera eingebaut. Sie kann die Nasen der Kunden erkennen und misst ständig die Nasenlänge. Das Steuerungsprogramm des Automaten speichert das aktuelle Messergebnis unter dem Namen `Nasenlänge` und unterscheidet die Kopfhaltungen der Kunden mit Hilfe dieser Tabelle:

Kameramessung	**Wert** `Nasenlänge`	**Kopfhaltung**
	1	Der Kunde schaut geradeaus.
	1,3	Der Kunde hat den Kopf gesenkt.
	0,7	Der Kunde hat den Kopf gehoben.

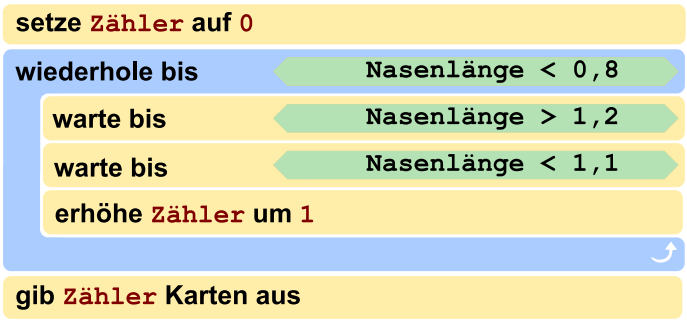
Das Steuerungsprogramm ist fast fertig – siehe unten.

Vervollständige das Steuerungsprogramm!



Lösung

Die einzige richtige Antwort ist:



Die Struktur des Programms ist vorgegeben: Es gibt eine zentrale Wiederholungsanweisung; oft auch *Schleife* genannt. Die letzte der in dieser Schleife wiederholten Anweisungen erhöht den Zähler der auszugebenden Karten. Folglich muss mit den zwei `warte bis`-Anweisungen ein Nicken des Kunden wahrgenommen werden: also dass der Kunde zuerst den Kopf gesenkt hat und dann wieder geradeaus schaut. Der unter Nasenlänge gespeicherte Wert muss also zuerst bei etwa 1,3 und dann wieder bei 1 liegen. Das entspricht den Bedingungen `Nasenlänge > 1,2` gefolgt von `Nasenlänge < 1,1`.

Anweisungen in der Schleife werden wiederholt, bis der Kunde den Kopf hebt: also ein Wert gemessen wird, der deutlich kleiner als 1 ist. Die einzige dazu passende Bedingung ist `Nasenlänge < 0,8`.

Vielleicht ist dir aufgefallen, dass das Programm nicht exakt die Werte aus der Tabelle verwendet. In der Praxis kann nämlich nicht kontinuierlich gemessen werden, sondern nur mit einer bestimmten Frequenz (beispielsweise 25 Mal pro Sekunde). Da kann es vorkommen, dass zum Beispiel der exakte Wert für das Geradeausschauen von 1,0 gar nicht gemessen wird, weil zuvor beispielsweise 0,95 und danach 1,03 gemessen wird.

Dies ist Informatik!

Maschinelles Sehen (engl. *machine vision*, auch als *computer vision* bezeichnet) ist ein Teilgebiet der Informatik, in dem momentan intensiv geforscht wird. Sowohl theoretische Überlegungen als auch praktische Anwendungen sind von grosser Wichtigkeit.

Eine bedeutsame Anwendung des maschinellen Sehens ist, Behinderten eine bessere Möglichkeit zu geben, autonom mit ihrer Umwelt zu interagieren. Je nach Stärke der Behinderung kann ein Mensch beispielsweise nur noch sehr wenige Muskeln kontrollieren. Der weltbekannte Physiker Stephen Hawking (1942–2018) nutzte Bewegungen seiner Wangenmuskeln, um ein Programm zur Sprachausgabe zu steuern, nachdem er die Kontrolle über seine weitere Muskulatur grösstenteils verloren hatte.

Das konkrete Beispiel jedoch könnte auch für Musiker eingesetzt werden: sie nutzen in der Regel beide Hände, um ihr Instrument zu bedienen. Handelsübliche Geräte bieten hierfür ein Fusspedal.

Bestimmte Musiker wie Organisten nutzen jedoch auch die Füsse zum Spielen und könnten durch ein einfaches Nicken beispielsweise Noten automatisch umblättern.

Im Gegensatz zum Beispiel dieser Aufgabe, wo die Werte konkret und fest einprogrammiert sind, wird maschinelles Sehen oftmals mit *maschinellem Lernen* (engl. *machine learning*) kombiniert. Dann wird das Programm auf bestimmte Gesten trainiert, indem ihm für jede Geste viele Beispiele und Gegenbeispiele gezeigt werden. So baut die Software eine statistische Einschätzung auf, was wie zu interpretieren ist.

Stichwörter und Webseiten

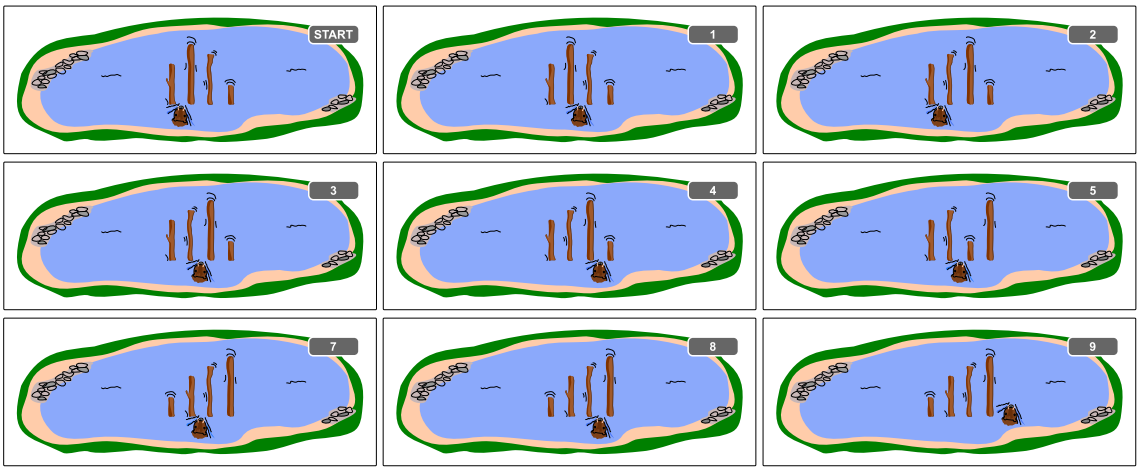
- Maschinelles Sehen, Machine Vision, Computer Vision:
  `https://de.wikipedia.org/wiki/Computer_Vision`
- Maschinelles Lernen, Machine Learning:
  `https://en.wikipedia.org/wiki/Machine_learning`
- `https://en.wikipedia.org/wiki/Stephen_Hawking#Disability`
- `https://de.wikipedia.org/wiki/Notenwender`

14. Beaver Sort

Biber Hamid sortiert die Baumstämme im See. Von links nach rechts sollen sie immer länger werden.

- Hamids Startposition ist zwischen den beiden Stämmen ganz links.
- Wenn er zwischen zwei benachbarten Stämmen ist, vergleicht er diese:
  - Wenn der rechte Stamm länger ist als der linke, schwimmt er eins nach rechts.
  - Wenn hingegen der linke Stamm länger ist, dann vertauscht er die beiden Stämme. Nach dem Vertauschen schwimmt er eins nach rechts, wenn er in der Startposition ist, und sonst eins nach links.
- So macht Hamid weiter, bis er rechts von allen Stämmen angekommen ist. Dann weiss er, dass alle Stämme korrekt sortiert sind.

Das Beispiel zeigt, wie Hamid 4 Baumstämme sortiert. Er macht dabei insgesamt 9 Vergleiche.



Die Anzahl der Vergleiche hängt davon ab, wie die Stämme am Anfang liegen. Für 4 Stämme muss Hamid mindestens 3 Vergleiche machen (wenn die Stämme bereits richtig sortiert sind) und höchstens 9 Vergleiche (wenn die Stämme alle genau verkehrt sortiert sind). Bei 4 Baumstämmen muss Hamid also mit 3 bis 9 Vergleichen rechnen.

Hamid muss nun 40 verschieden lange Baumstämme sortieren. Mit wie vielen Vergleichen muss er rechnen?

A) Mit 0 bis 20 Vergleichen
B) Mit 3 bis 40 Vergleichen
C) Mit 39 bis 120 Vergleichen
D) Mit 39 bis 1521 Vergleichen

Lösung

Für 40 Stämme benötigt Hamid im besten Fall bereits 39 Vergleiche, nämlich dann, wenn alle Baumstämme bereits sortiert sind und er somit immer nach rechts schwimmt, bis er fertig ist. Damit können die Antworten A) und B) nicht richtig sein.

Im ungünstigsten Fall sind die Baumstämme am Anfang genau verkehrt herum sortiert. Hier ist die Anzahl der Vergleiche schwieriger zu bestimmen. Wir beobachten zuerst Folgendes: Die Baumstämme links von Hamid sind unter sich immer richtig sortiert. Wenn Hamid zu einem Baumstamm kommt, der kürzer ist als die k links von ihm, dann tauscht er ihn zuerst ganz nach links durch, wobei er k Vergleiche macht. Dann kommt er zurück und schwimmt sogar noch eins weiter nach rechts, wobei er $k-1$ weitere Vergleiche macht, aber nichts mehr vertauscht. Insgesamt sind das $k+(k-1)=2k-1$ Vergleiche. Dies passiert bei jeder der $n-1$ Positionen zwischen zwei Baumstämmen und die Anzahl Baumstämme links von Hamid ist dabei im Durchschnitt $\frac{n}{2}$. Somit macht er insgesamt $(n-1)\cdot(2\cdot\frac{n}{2}-1)=(n-1)^2$, also $39^2=1521$ Vergleiche. Also ist Antwort C) mit höchstens 120 Vergleichen falsch und Antwort D) ist korrekt.

Dies ist Informatik!

Das Sortieren von Elementen ist eine klassische Aufgabenstellung der Informatik. Effizientes Sortieren spart dabei viel Zeit. Der hier präsentierte Sortieralgorithmus heisst *Gnome Sort* und wurde ursprünglich von dem iranischen Informatiker Hamid Sarbazi-Azad als *Stupid Sort* vorgestellt. Später hat der niederländische Informatiker Dick Grune dieses Sortierverfahren als Gnome Sort bezeichnet, mit der Vorstellung, dass ein Gartenzwerg (engl. *garden gnome*) so Blumentöpfe sortieren könnte.

Die Analyse der Laufzeit von Algorithmen (also das Herausfinden davon, wie lange ein Algorithmus brauchen kann, bis er fertig ist) ist in der Informatik sehr wichtig. Oft fragt man, was die Laufzeit im besten Fall, im durchschnittlichen Fall und im schlechtesten Fall ist, und zwar in Abhängigkeit von der Eingabegrösse n. (In unserem Fall nehmen wir dafür die Anzahl Baumstämme.) Um den Vergleich zwischen verschiedenen Laufzeiten möglichst einfach zu halten, begnügt man sich meistens damit, sie nur grob anzugeben. Gnome Sort braucht im besten Fall nur lineare Laufzeit - man schreibt dafür $\mathcal{O}(n)$ -, im durchschnittlichen Fall quadratische Laufzeit - man schreibt dafür $\mathcal{O}(n^2)$ - und im schlimmsten Fall ebenfalls quadratische Laufzeit - man schreibt also wieder $\mathcal{O}(n^2)$.

Stichwörter und Webseiten

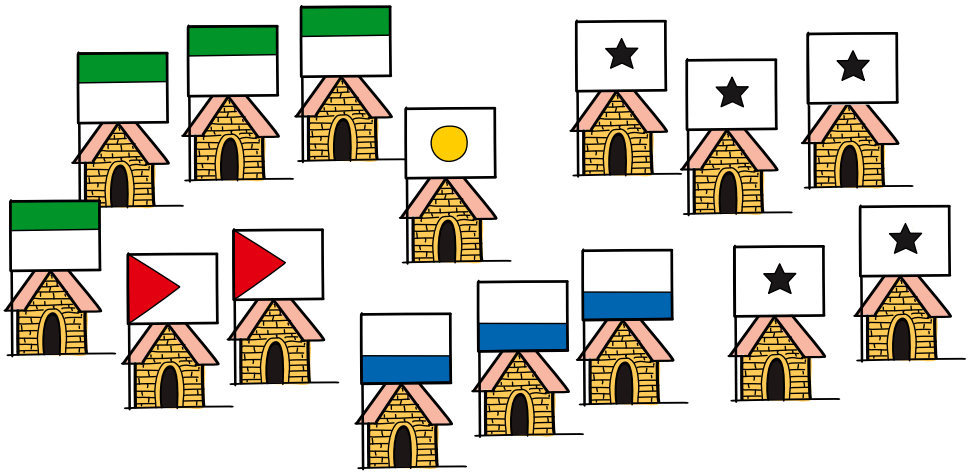
- Gnome Sort: `https://de.wikipedia.org/wiki/Gnomesort`
- Sortierverfahren: `https://de.wikipedia.org/wiki/Sortierverfahren`
- Laufzeitanalyse
- O-Notation, Landau-Symbole: `https://de.wikipedia.org/wiki/Landau-Symbole`

15. Die Clans von Beavaria

In Beavaria leben fünf einst verfeindete Clans mit jeweils einigen Häusern, wie man im Bild erkennen kann: Mac Intoshs, Apfler, Mac Rosofts, Androidier und Freidosier. Da die Zeiten lange schon friedlich sind, beschliessen sie, das Vereinigungsritual durchzuführen. Die Regeln dafür sind:

- Es dürfen sich immer nur zwei Clans zur selben Zeit vereinigen.
- In jedem Haus der sich vereinigenden Clans wird eine Woche lang gefeiert, um den Pakt zu besiegeln. Die Dauer des Vereinigens in Wochen ist daher gleich der Anzahl der Häuser in den beiden Clans.
- Nach dieser Zeit sind die beiden Clans nur noch ein Clan. Nun kann das Vereinigen der Clans fortgesetzt werden.

Die Clans beschliessen, die Vereinigung in möglichst kurzer Zeit durchzuführen. Das geht nur, indem man die Reihenfolge der Vereinigungen sorgfältig plant.



Wie viele Wochen dauert es mindestens bis alle Clans vereinigt sind?

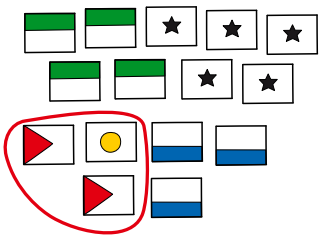
A) 15 Wochen
B) 33 Wochen
C) 35 Wochen
D) 50 Wochen
E) 120 Wochen

Lösung

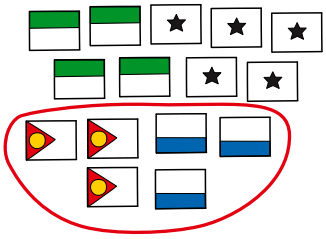
Die richtige Antwort ist B) 33 Wochen.

Die optimale Strategie zur Minimierung der Gesamtzahl der Wochen, die für die Vereinigung aller Clans benötigt werden, besteht darin, die Anzahl der jeweils beteiligten Häuser zu minimieren. Die Häuser der zuerst vereinigten Clans werden dann ja auch bei den weiteren Vereinigungen berücksichtigt. Daher ist es sinnvoll, zunächst jeweils die kleinsten Clans zu vereinigen, so dass die grossen nicht zu oft an der Zeremonie teilnehmen müssen. Um dies zu erreichen, sollten in jedem Schritt die beiden Clans mit den wenigsten Häusern vereinigt werden.

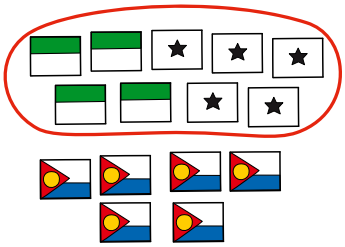
Dieser schrittweise Prozess ist in folgender Tabelle dargestellt, die Clannamen lassen wir der Übersicht halber weg und nennen nur in Klammern die Clangrössen:



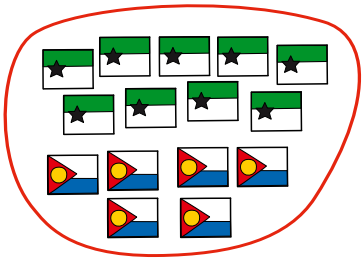
(1) und (2) werden zu einem Clan (3) vereinigt, dafür braucht man drei Wochen.



(3) und (3) werden in sechs Wochen zu (6) vereinigt.



Nun werden (4) und (5) in neun Wochen zu (9) vereinigt.



Zuletzt werden die verbliebenden (6) und (9) vereinigt. Das braucht fünfzehn Wochen.

Insgesamt ist der Prozess also nach $15 + 9 + 6 + 3 = 33$ Wochen abgeschlossen, das ist Antwort B).

Dies ist Informatik!

Dies ist ein *Optimierungsproblem* — eine Aufgabe, einen Wert (in diesem Fall die Dauer der Vereinigung) zu minimieren oder zu maximieren. Hierfür werden oft *Algorithmen* verwendet, die zum Beispiel für ressourcenschonende, umweltfreundliche Produktionsprozesse heute nicht mehr wegzudenken sind.

In diesem Fall kann das Problem sogar mit einem einfachen *Greedy-Algorithmus* optimal gelöst werden. Hierbei tut man in jedem Schritt das, was nach einer einfachen Idee gerade am besten zu sein scheint. In unserem Fall ist die Idee, dass wir zuerst kleine Clans vereinigen, weil dies am wenigsten Zeit kostet.

Stichwörter und Webseiten

- Optimierungsproblem - `https://de.wikipedia.org/wiki/Optimierungsproblem`
- Greedy-Algorithmus - `https://de.wikipedia.org/wiki/Greedy-Algorithmus`

A. Aufgabenautoren

Michael Barot
Liam Baumann
Wilfried Baumann
Lucia Budinská
Sarah Chan
Anton Chukhnov
Kris Coolsaet
Valentina Dagienė
Christian Datzko
Susanne Datzko
Janez Demšar
Fabian Frei
Gerald Futschek
Jens Gallenbacher
Thomas Galler
Mark Edward M. Gonzales
Martin Guggisberg
Mathias Hiron
Juraj Hromkovič
Svetlana Jaksic
Martin Kandlhofer
Ulrich Kiesmüller
Dong Yoon Kim
Vaidotas Kinčius
Regula Lacher
Taina Lehtimäki
Angélica Herrera Loyo
Tom Naughton
Mochammad Irfan Noviana
Gabriela Lourdes Rodríguez Parada
Jean-Philippe Pellet
Hannah Piper
Jonatan Pipping
Zsuzsa Pluhár
Wolfgang Pohl
Rodrigo Santamaría
Tomas Šiaulys
Bernadette Spieler
Cuttle.org Team
Ezra Templonuevo
Eslam Wageed
Michael Weigend
Mija Zaļūksne

B. Sponsoring: Wettbewerb 2021

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

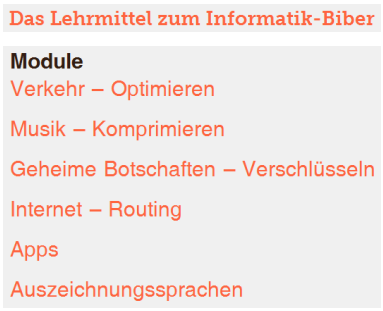
http://www.phlu.ch/
Pädagogische Hochschule Luzern

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.phzh.ch/
Pädagogische Hochschule Zürich

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.