INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Aufgaben und Lösungen 2018
Alle Stufen

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

Herausgeber:
Christian Datzko, Susanne Datzko, Hanspeter Erni

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

Mitarbeit Informatik-Biber 2018

Andrea Adamoli, Christian Datzko, Susanne Datzko, Olivier Ens, Hanspeter Erni, Martin Guggisberg, Carla Monaco, Gabriel Parriaux, Elsa Pellet, Jean-Philippe Pellet, Julien Ragot, Beat Trachsler.

Herzlichen Dank an:
Juraj Hromkovič, Urs Hauser, Regula Lacher, Jacqueline Staub: ETHZ
Andrea Maria Schmid, Doris Reck: PH Luzern
Gabriel Thullen: Collège des Colombières
Valentina Dagienė: Bebras.org
Hans-Werner Hein, Ulrich Kiesmüller, Wolfgang Pohl, Kirsten Schlüter, Michael Weigend: Bundesweite Informatikwettbewerbe (BWINF), Deutschland
Chris Roffey: University of Oxford, Vereinigtes Königreich
Anna Morpurgo, Violetta Lonati, Mattia Monga: ALaDDIn, Università degli Studi di Milano, Italien
Gerald Futschek, Wilfried Baumann: Oesterreichische Computer Gesellschaft, Österreich
Zsuzsa Pluhár: ELTE Informatikai Kar, Ungarn
Eljakim Schrijvers, Daphne Blokhuis, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers: Eljakim Information Technology bv, Niederlande
Roman Hartmann: hartmannGestaltung (Flyer Informatik-Biber Schweiz)
Christoph Frei: Chragokyberneticks (Logo Informatik-Biber Schweiz)
Andrea Adamoli (Webseite)
Andrea Leu, Maggie Winter, Brigitte Maurer: Senarclens Leu + Partner

Die deutschsprachige Fassung der Aufgaben wurde ähnlich auch in Deutschland und Österreich verwendet.
Die französischsprachige Übersetzung wurde von Nicole Müller und Elsa Pellet und die italienischsprachige Übersetzung von Andrea Adamoli erstellt.

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Der Informatik-Biber 2018 wurde vom Schweizerischen Verein für Informatik in der Ausbildung SVIA durchgeführt und von der Hasler Stiftung unterstützt.

HASLERSTIFTUNG

Hinweis: Alle Links wurden am 1. November 2018 geprüft. Dieses Aufgabenheft wurde am 9. Oktober 2019 mit dem Textsatzsystem LaTeX erstellt.

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“ auf dem 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 2018 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

Die Stufen 3 und 4 hatten 9 Aufgaben zu lösen, jeweils drei davon aus den drei Schwierigkeitsstufen leicht, mittel und schwer. Die Stufen 5 und 6 hatten 12 Aufgaben zu lösen, jeweils vier davon aus den drei Schwierigkeitsstufen leicht, mittel und schwer. Jede der anderen Altersgruppen hatte 15 Aufgaben zu lösen, jeweils fünf davon aus den drei Schwierigkeitsstufen leicht, mittel und schwer.
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

Das international angewandte System zur Punkteverteilung soll dem erfolgreichen Erraten der richtigen Lösung durch die Teilnehmenden entgegenwirken.
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 („Kleiner Biber“: 108 Punkte, Stufen 5 und 6: 144 Punkte) 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.

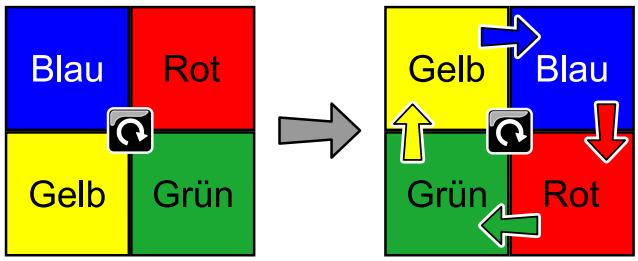
Für weitere Informationen:

SVIA-SSIE-SSII Schweizerischer Verein für Informatik in der Ausbildung
Informatik-Biber
Hanspeter Erni
`https://www.informatik-biber.ch/de/kontaktieren/`
`https://www.informatik-biber.ch/`
 `https://www.facebook.com/informatikbiberch`

Inhaltsverzeichnis

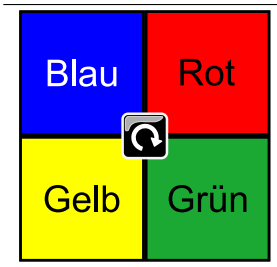
1. Simon sagt

Immer wenn Simon den Knopf in der Mitte drückt, bewegen sich die Vierecke wie gezeigt:

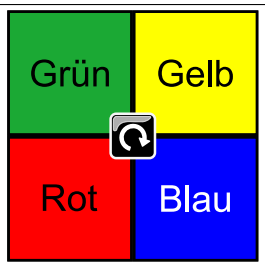


Simon drückt den Knopf in der Mitte noch weitere zwei Mal. Wie liegen die Vierecke dann?

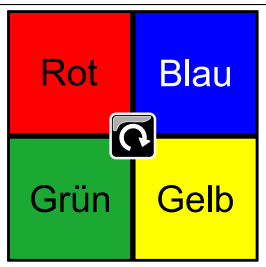
A)



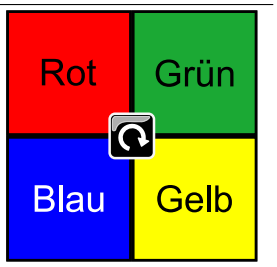
B)



C)

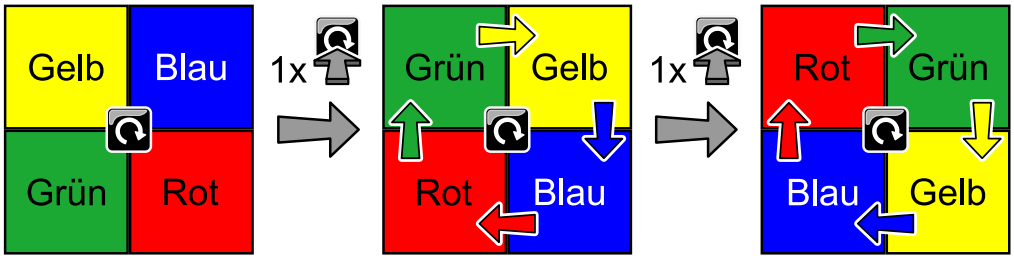


D)



Lösung

Die richtige Antwort ist D):



Dies ist Informatik!

Die Aufgabe beschreibt eine Maschine, die für vier Positionen jeweils einen bestimmten Status hat: Rot, Grün, Blau und Gelb. Ein Knopfdruck ändert den Status jeder Position in der Reihenfolge Rot → Blau → Gelb → Grün → Rot. Eine solche Maschine nennt man in der Informatik einen *endlichen Automaten*. Menschen fragen sich lieber, wo die Farben landen, also dass sie sich im Uhrzeigersinn bewegen.
Es gibt viele Spiele, wo man bestimmte Regeln befolgen muss. Das Spiel „Simon Says“ ist so ein Spiel. Auf Deutsch kennt man es als „Kommando Pimperle“. Der Spielleiter gibt Aufträge an die Mitspieler, die diese aber nur dann befolgen müssen, wenn der Spielleiter vorher „Kommando“ gesagt hat. Wenn er also beispielsweise sagt: „Alle strecken die Zunge raus“, macht niemand etwas, wenn er aber sagt: „Kommando alle hüpfen auf einem Bein“, müssen alle auf einem Bein hüpfen. Wer einen Fehler macht, fliegt raus.

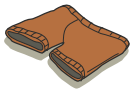
Stichwörter und Webseiten

Endlicher Automat

- `https://de.wikipedia.org/wiki/Simon_says_(Spiel)`
- `https://de.wikipedia.org/wiki/Endlicher_Automat`

2. Der Kleiderhaufen

Die Bibermama ordnet die Kleider ihres Sohnes Bruno auf dem Tisch.

Hemd	Unterhemd	Hose	Unterhose	Hosenträger	Socken	Schuhe
						

Bruno zieht seine Kleidung in der Reihenfolge an, in der sie auf dem Tisch liegen. Er beginnt immer mit dem Kleidungsstück, das zuoberst auf diesem Kleiderhaufen liegt. Bruno möchte aber seine Hosenträger nicht unter seinem Hemd tragen.

Welchen Kleiderhaufen kann Bruno verwenden?

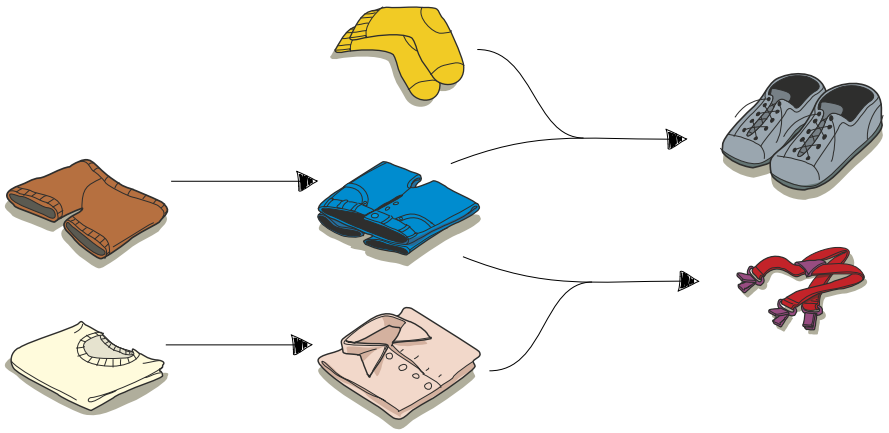
A)	B)	C)	D)
			

Lösung

B) ist der richtige Kleiderhaufen.
Um die Lösung zu finden, sollten wir den Kleiderhaufen von oben her anschauen. Dabei muss das Hemd vor den Hosenträgern kommen.
Die Antworten A), C) und D) sind unter anderem falsch, weil die Hosenträger vor dem Hemd angezogen werden.

Dies ist Informatik!

„Bevor du in einen Raum gehen kannst, musst du die Tür öffnen.“ Dies ist ein Beispiel für eine klare Einschränkung: Die Tür muss offen sein, bevor du hineingehen kannst, um das zu bekommen, was du möchtest.
Diese Aufgabe kann gelöst werden indem geprüft wird, welche Listen die Bedingung der Form erfüllen, „Das Kleidungsstück X muss vor der Kleidungsstück Y angezogen werden.“ Wenn eine Liste von Dingen alle Bedingung erfüllt, ist sie korrekt. Wenn nur eine Bedingung nicht erfüllt wird, ist sie nicht korrekt.
Bei den Kleidungsstücken der Aufgabe gibt es noch mehr Einschränkungen als die Hosenträger: beispielsweise können die Schuhe nicht vor den Socken oder der Hose angezogen werden.



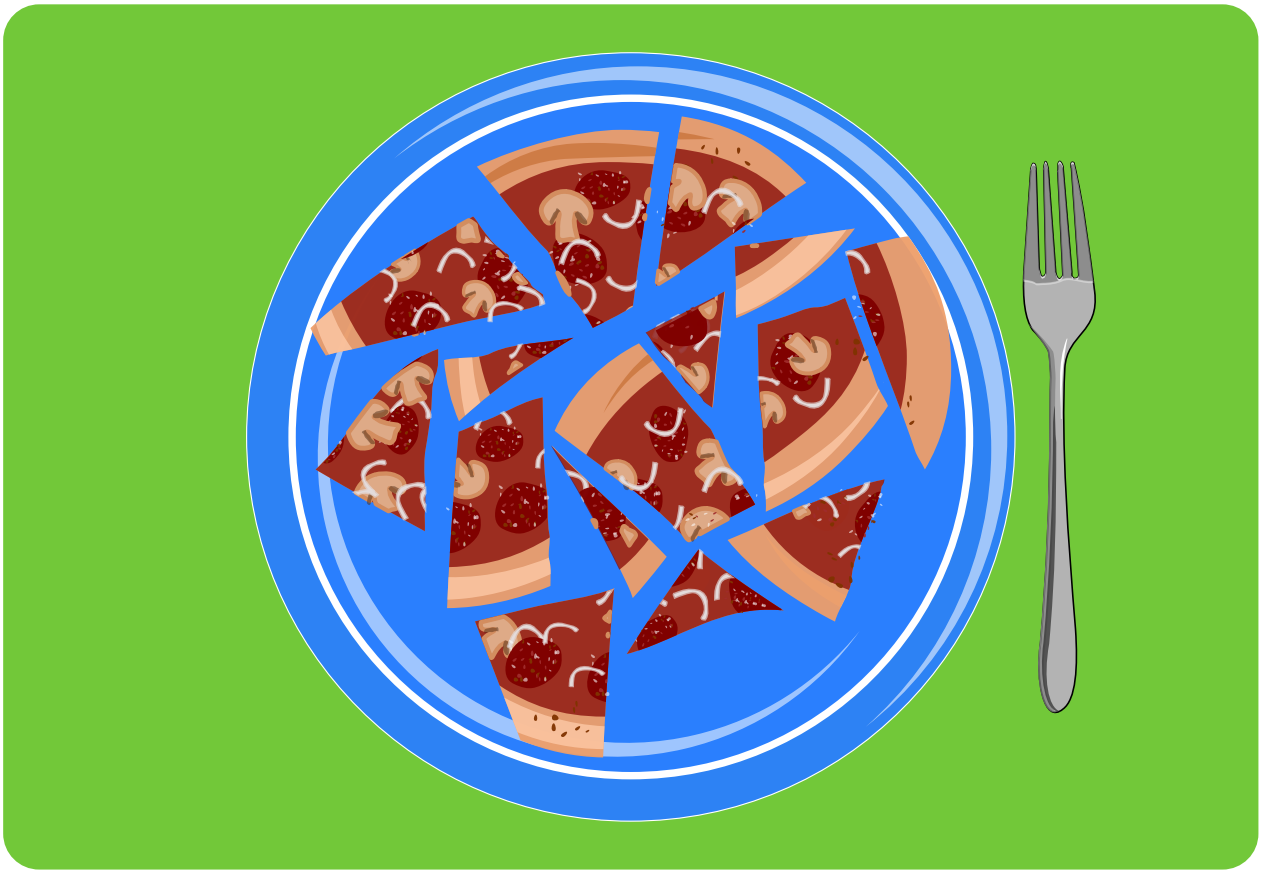
Stichwörter und Webseiten

Scheduling

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

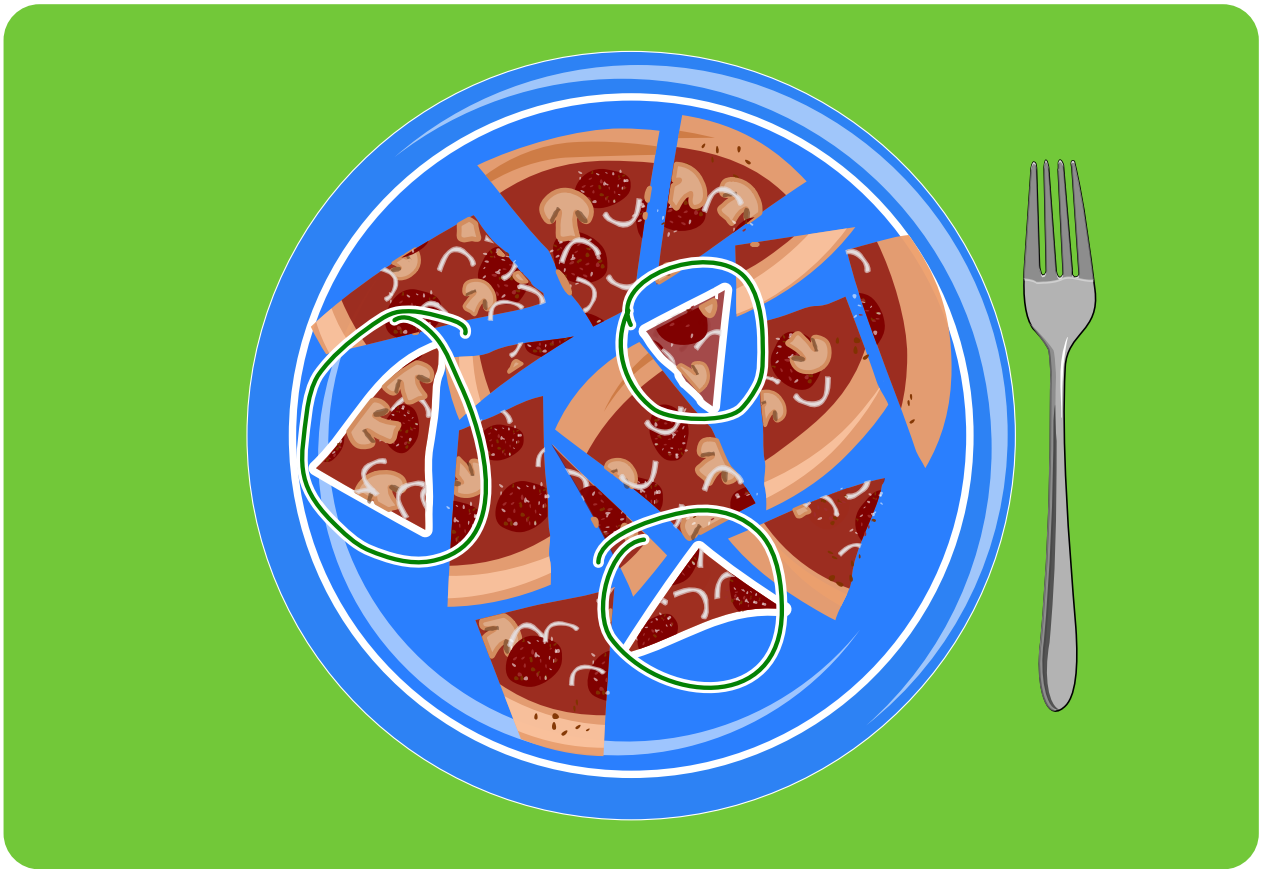
3. Pizza

Lucillas Mutter hat die Pizza in Stücke geschnitten. Lucilla möchte alles mit der Hand essen. Alle randlosen Pizzastücke soll sie aber mit der Gabel zu essen. Welche sind das?



Lösung

Die richtige Antwort ist:



Diese drei Stücke haben jeweils keinen Rand, alle anderen Stücke haben einen Rand.

Dies ist Informatik!

Für jedes Stück muss Lucilla sich fragen, ob es einen Rand hat oder nicht. Solche Entscheidungen für eine Menge von Dingen muss ein Computer auch öfters machen. Man nennt dies eine Verzweigung. In einem Computerprogramm würde man dann vielleicht so etwas schreiben:

```
WENN das Stück einen Rand hat
    DANN iss es mit der Hand
    SONST iss es mit der Gabel
```

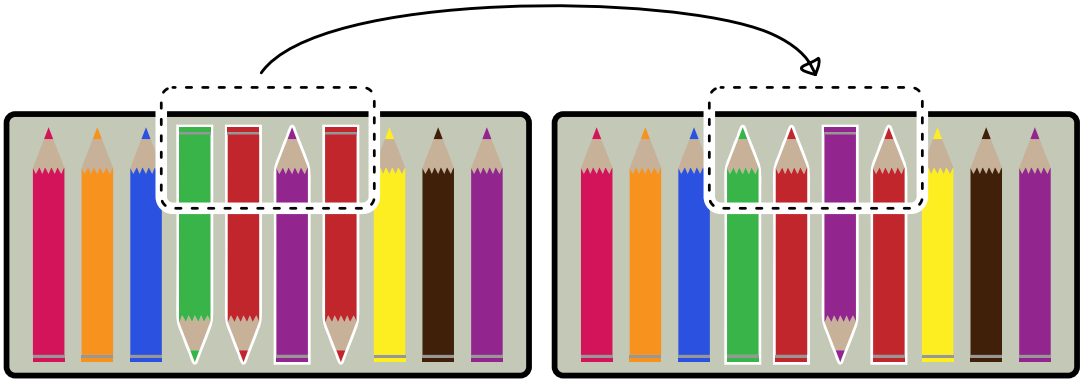
Stichwörter und Webseiten

Verzweigung

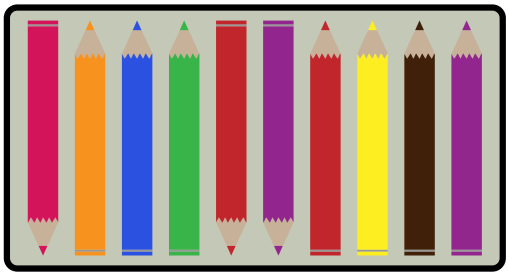
- `https://de.wikipedia.org/wiki/Bedingte_Anweisung_und_Verzweigung`

4. Adas Farbstifte

Ada hat eine Schachtel mit 10 Farbstiften. Einige zeigen nach oben, einige zeigen nach unten. Ada möchte, dass alle Farbstifte nach oben zeigen.
Als Spiel dreht sie immer nur zwei oder mehr nebeneinanderliegende Farbstifte gleichzeitig um. Im folgenden Beispiel dreht sie den vierten, fünften, sechsten und siebten Farbstift gleichzeitig um.

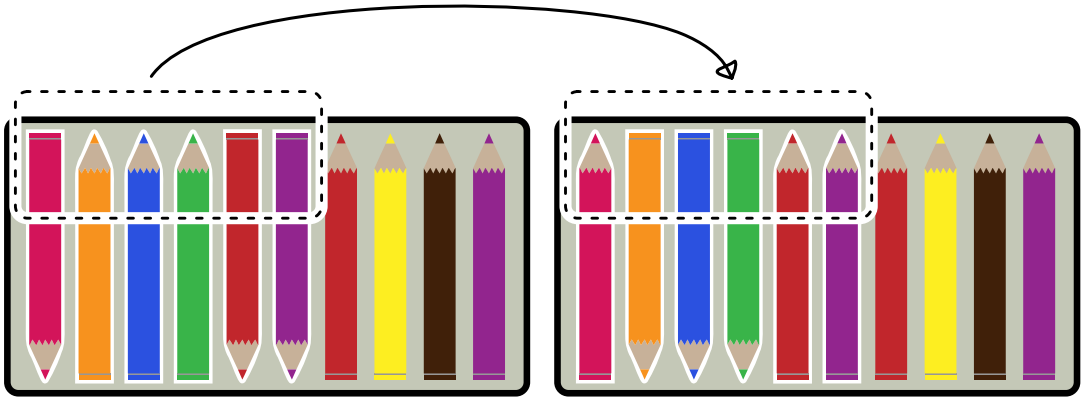


Ada möchte, dass alle Farbstifte in der Schachtel nach oben zeigen. Sie will so wenig Schritte wie möglich machen. Mit wie wenig Schritten schafft sie es?

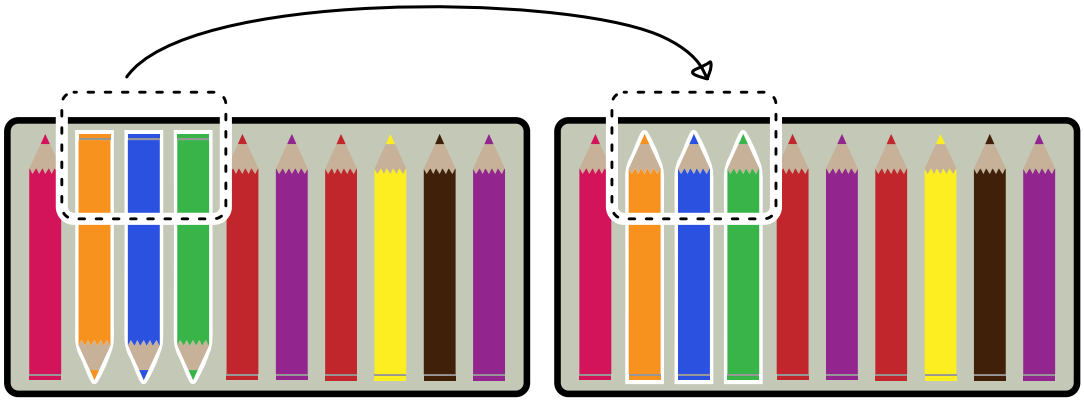


Lösung

Es genügt, zwei mal Farbstifte nach der Spielregel umzudrehen. Einmal würde nicht genügen, da die Farbstifte nicht alle nebeneinander liegen.
Wenn Ada aber die Stifte 1 bis 6 umdreht...



...und danach die Stifte 2 bis 4 umdreht, ...



...liegt alles wie gewünscht.

Dies ist Informatik!

Ada hat wahrscheinlich genug Zeit, viel mehr als zwei Schritte zu machen, damit die Stifte wie gewünscht liegen. Und wenn sie keine Lust mehr hat, kann sie natürlich auch einen einzelnen Stift umdrehen.
Computer haben es nicht so einfach. Wenn man einen Computer programmiert hat, hält er sich an die Spielregeln. Computer haben es auch nicht mit einigen wenigen Stiften zu tun, sondern müssen vielleicht dieselbe Spielregel auf ganz viele Daten anwenden.
Ein Beispiel, wo das wichtig ist, ist der Festspeicher von Computern. Heutzutage baut man in der Regel eine SSD (Solid State Disk) ein. Das Löschen einer einzelnen Speicherinformation geht nicht alleine, sondern nur in Form von Blöcken. Beim Schreiben muss der Computer also darauf achten, dass der ganze Block unbenutzt ist, sonst muss er erst die benutzten Teile lesen, dann den Block löschen und dann den Block mit den alten und neuen Daten neu beschreiben. Das ist viel langsamer, als wenn der Computer unbenutzte Blöcke kennt und schon einmal löscht, während er sonst nichts zu tun hat. Wenn er viel schreiben muss, lohnt es sich für ihn, die Daten immer in ganzen Blöcken zu speichern, denn ob er nun nur eine einzelne Speicherinformation oder einen ganzen Block schreibt, ist für ihn gleich schnell.

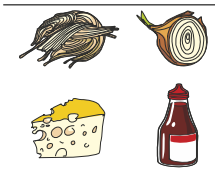
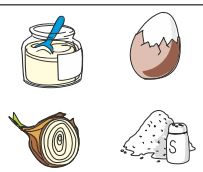
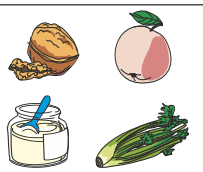
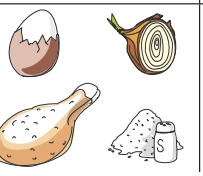
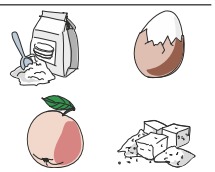
Stichwörter und Webseiten

Effizienz

- `https://www.codechef.com/problems/ADACRA`
- `https://de.wikipedia.org/wiki/Effizienz_(Informatik)`
- `https://de.wikipedia.org/wiki/Landau-Symbole`
- `https://de.wikipedia.org/wiki/TRIM`

5. Ähnliche Gerichte

Ein Koch möchte 2 Gerichte zubereiten. Diese beiden Gerichte sollen nicht ähnlich sein. Zwei Gerichte sind für den Koch ähnlich, wenn sie mindestens 2 gleiche Zutaten haben.

Pasta	Eiersalat	Nusssalat	Hühnersuppe	Torte
				

Welche Gerichte sind ähnlich?

A) Hühnersuppe und Pasta

B) Hühnersuppe und Nusssalat

C) Hühnersuppe und Eiersalat

D) Nusssalat und Torte

Lösung

Die richtige Antwort ist C) Hühnersuppe und Eiersalat.
In der Hühnersuppe und im Eiersalat sind jeweils Ei, Zwiebel und Salz drin.
In den anderen Kombinationen von Speisen ist höchstens eine gemeinsame Zutat drin: Hühnersuppe und Nusssalat haben keine gemeinsame Zutat. Hühnersuppe und Pasta enthalten gemeinsam Zwiebel. Nusssalat und Torte haben keine gemeinsame Zutat.

Dies ist Informatik!

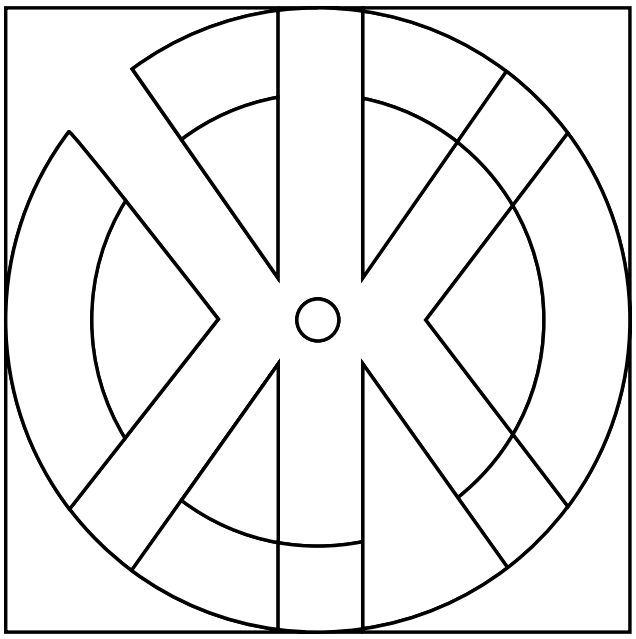
An vielen Stellen muss man Dinge vergleichen und herausfinden, was gleich und was verschieden ist. Biologen vergleichen beispielsweise das Erbgut von Bakterien, Chemiker vergleichen Eigenschaften von Stoffen, Astronomen vergleichen Galaxien, Sterne und Planeten, und so weiter.
Um Dinge zu vergleichen, muss man definieren, welche Eigenschaften man vergleicht. Dazu kann man dann bestimmen, ab wann zwei Dinge ähnlich sein sollen oder nicht. So kann man beispielsweise sagen, dass ein Tisch und ein Stuhl beide aus Holz sind, also ähnlich sind. Man kann ebenso sagen, dass ein Tisch nicht dazu gedacht ist, dass man sich auf ihn setzt. Während ein Stuhl nicht dazu gedacht ist, dass man auf ihm einen Brief schreibt (auch wenn beides natürlich denkbar ist). Man kann aber auch sagen, dass zwei Holzstühle erst dann ähnlich sind, wenn sie aus demselben Holz gemacht sind.
In dieser Aufgabe sind fünf Gerichte mit je vier Zutaten zu vergleichen. Biologen, Chemiker, Astronomen und viele andere Wissenschaftler vergleichen nicht nur so wenige Dinge sondern gleich tausende, Millionen oder Milliarden von Dingen, evtl. mit vielen verschiedenen Eigenschaften, die alle in die „Ähnlichkeit“ hineinspielen. Hier kommt die Informatik ins Spiel, die das automatisierte Vergleichen von grossen Datenmengen aufgrund vordefinierter Ähnlichkeitsmasse ermöglicht.

Stichwörter und Webseiten

Objekt, Eigenschaft, Ähnlichkeitsmass, Big Data

- `https://de.wikipedia.org/wiki/Ähnlichkeitsanalyse`
- `https://de.wikipedia.org/wiki/Big_Data`

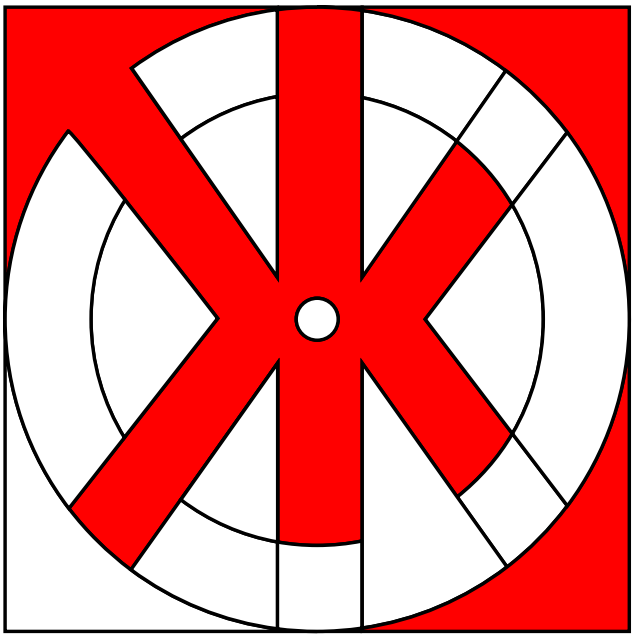
6. Muster einfärben



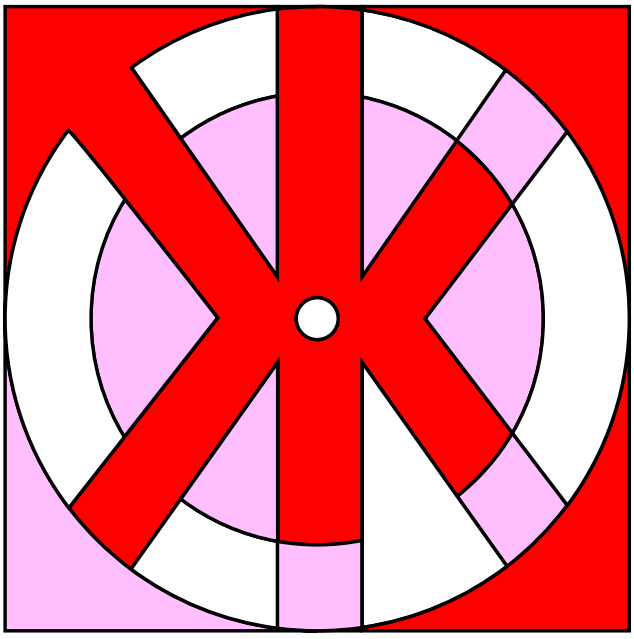
Die Biber möchten das Muster oben einfärben. Hilf Ihnen dabei und färbe die Flächen so ein, dass zwei Flächen nebeneinander verschiedene Farben haben. Nutze zudem so wenig Farben wie möglich.

Lösung

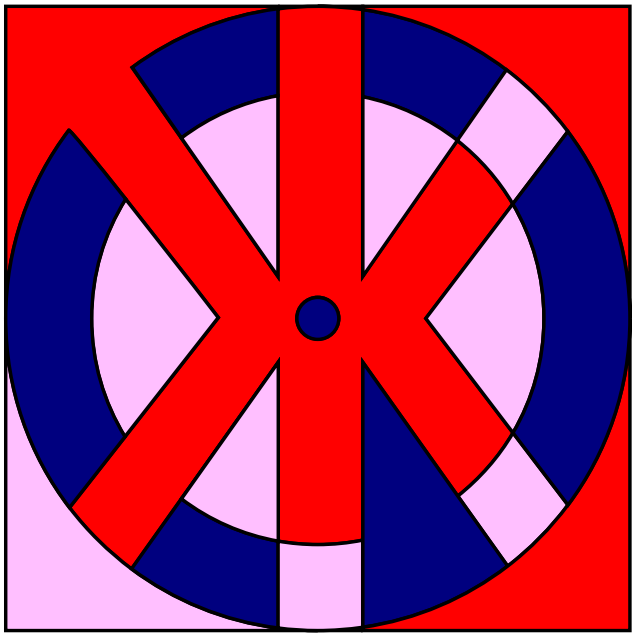
Es genügen drei verschiedene Farben.
Es gibt ganz unterschiedliche Lösungen, je nachdem mit welcher Startfarbe man beginnt. Beginnt man zum Beispiel mit Rot in der Ecke oben links und färbt alle Flächen, die unberührt sind, in derselben Farbe ein, sieht die Figur so aus:



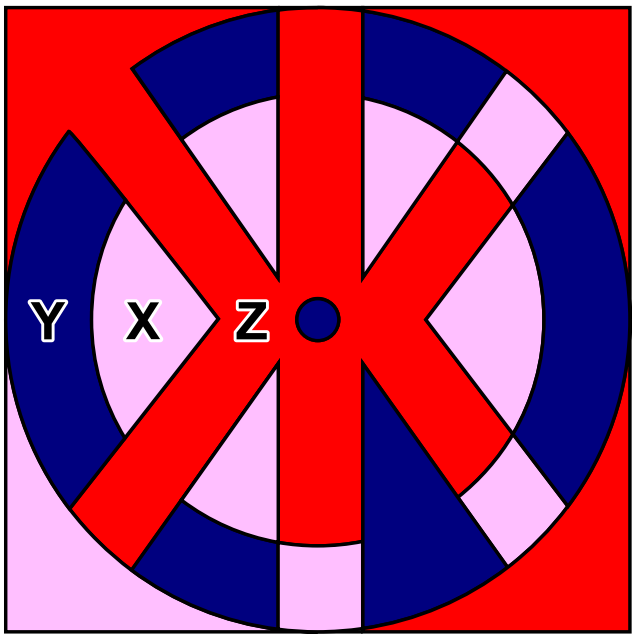
Wenn man mit einer zweiten Farbe (z.B. Rosa) mit der Ecke unten links weiter macht und immer die nächstmögliche Fläche in dieser Farbe ausfüllt, sieht die Figur so aus:



Eigentlich ist man hier schon fertig, denn man hat die Figur mit den drei Farben Rot, Rosa und Weiss ausgefüllt. Man kann aber natürlich auch alle weissen Flächen noch einmal mit einer dritten Farbe (z.B. Blau) ausfüllen:



Weniger als drei Farben gehen nicht. Die Fläche X grenzt links an die Fläche Y. Die Flächen X und Y müssen also verschiedene Farben haben. Die Flächen X und Y grenzen aber beide an die Fläche Z. Damit kann Z nicht dieselbe Farbe wie X oder Y haben und muss daher also eine dritte Farbe haben.



Dies ist Informatik!

Wie viele Farben braucht man höchstens, um beliebige Flächen so auszumalen, dass zwei benachbarte Flächen nicht dieselbe Farbe haben? Die richtige Antwort ist: Vier Farben genügen, solange man keine „Enklaven“ zulässt. Eine Enklave ist eine abgeschlossene Teilfläche, die zu einer anderen Fläche gehört, aber nicht mit ihr verbunden ist, wie beispielsweise Büsingen am Hochrhein oder Campione d'Italia oder ... ganz spannend, der Ort Baarle in Niederlanden und Belgien. Dies nennt man den Vier-Farben-Satz oder auch das Vier-Farben-Theorem.

Der Beweis, dass vier Farben genügen, ist nicht einfach. Vor 200 Jahren wusste man bereits, dass fünf Farben genügen. Erst 1976 konnten die Mathematiker Kenneth Appel und Wolfgang Haken beweisen, dass vier Farben ausreichen. Dazu haben sie Computer verwendet, um eine Vielzahl von Ausnahmen und Gegenbeispielen zu überprüfen. Da es nicht mehr möglich ist, diese alle per Hand zu überprüfen, gab es viele Mathematiker, die den Computereinsatz hinterfragt haben. Auch heute gibt es Mathematiker, die in Frage stellen, ob es zulässig ist, einen Computer zum Beweisen einzusetzen. Der Vier-Farben-Satz wird an vielen Stellen angewendet, beispielsweise um Flugpläne zu erstellen, wenn Flugzeuge Korridoren zugeteilt werden, damit sie immer genügend Abstand haben, oder auch um Frequenzbereiche von Natel-Sendemasten zuzuteilen, so dass diese sich nicht stören und der Empfang trotz vieler Sendemasten nicht schlechter wird.

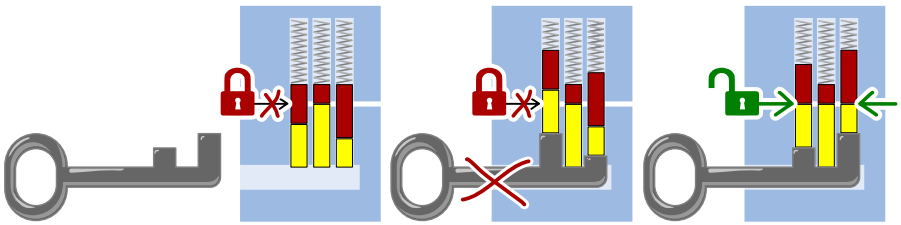
Stichwörter und Webseiten

Vier-Farben-Satz

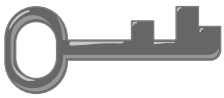
- `https://de.wikipedia.org/wiki/Vier-Farben-Satz`
- `http://www.mathepedia.de/Vier-Farben-Satz.html`
- `https://de.wikipedia.org/wiki/Enklave`
- `https://de.wikipedia.org/wiki/Baarle`

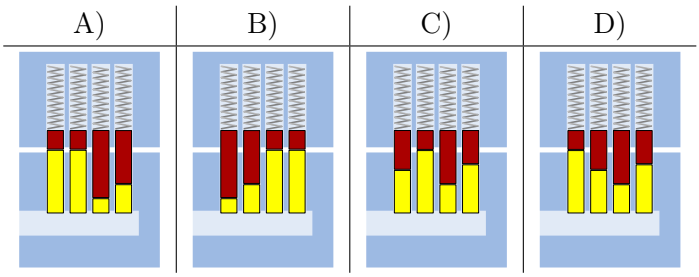
7. Sicherheitsschloss

Henry arbeitet bei einem Schlüsseldienst. Die Schlösser funktionieren so:



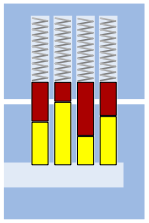
Zu welchem Schloss passt der folgende Schlüssel:



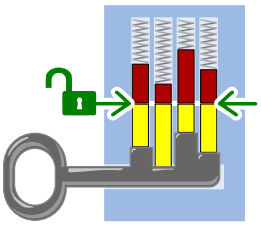


Lösung

Die richtige Antwort ist C):



Wenn man den Schlüssel hineinschiebt, sind die Höhen der vier Bartelemente zusammen mit den vier Stiften gleich, so dass der Zylinder gedreht werden kann:



Dies ist Informatik!

Ob ein Schlüssel sperrt oder nicht hängt davon ab, ob alle seine Glieder zu den einzelnen Elementen des Schlosses passen. Dabei muss ein langer Teil des Barts bei den kurzen Stiften und ein kurzer Teil des Barts bei den langen Stiften sein. Diese beiden Muster müssen zueinander passen. In unserem Fall ist es das, wenn sie genau gegensätzlich sind. Die Suche nach passenden Mustern ist eine Grundaufgabe der Informatik. Beispiele sind die Suche nach dem Vorkommen eines Wortes in einem Text oder die Suche nach ähnlichen Bildern.

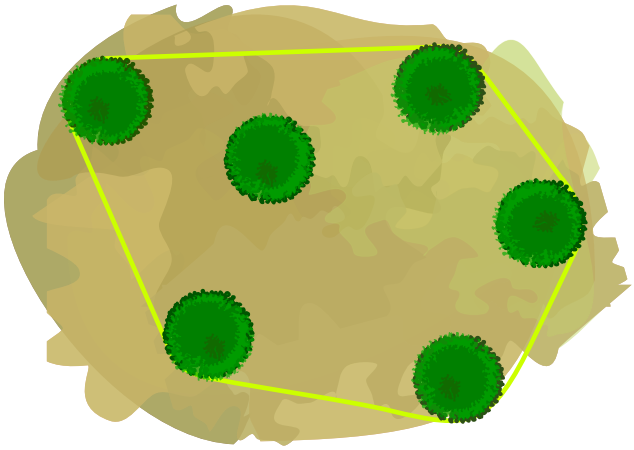
Stichwörter und Webseiten

Mustererkennung, Schloss

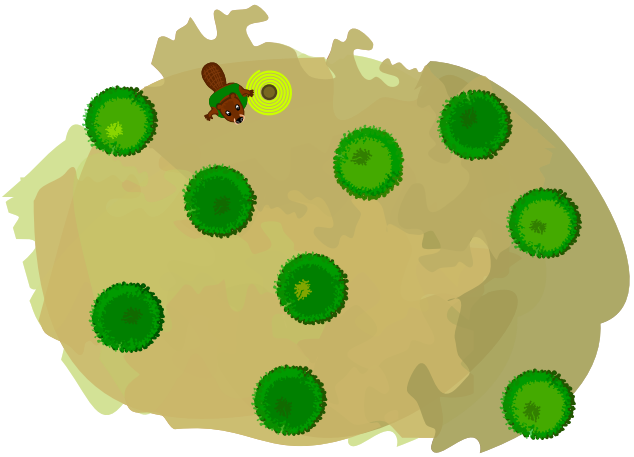
- `https://de.wikipedia.org/wiki/Schloss_(Technik)#Stiftschloss`

8. Buschkreis

Die Biber spannen ein langes Band um Büsche, die sie fällen wollen.
Gestern wollten sie sechs Büsche fällen. Das Band hat aber nur fünf Büsche berührt. Aus der Luft sah das so aus:



Heute wollen die Biber diese neun Büsche fällen:



Wie viele Büsche berührt das Band diesmal?

A) 3 Büsche

B) 4 Büsche

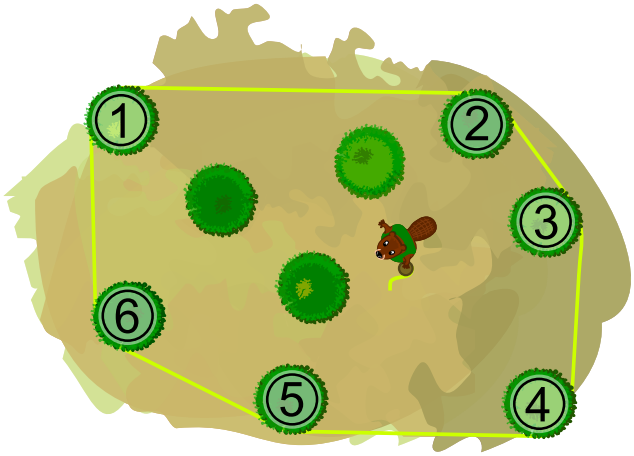
C) 5 Büsche

D) 6 Büsche

E) 9 Büsche

Lösung

Antwort D) ist richtig. Die Biber spannen das Band so um die Büsche:



Das Band berührt die sechs nummerierten Büsche.

Dies ist Informatik!

Das Band um die Büsche umschliesst das kleinste Vieleck ohne Einbuchtungen, auf der alle Büsche stehen, die gefällt werden sollen. Die einzige Einschränkung ist, dass die Grenzlinien der Fläche gerade Linien sind. Wenn die Büsche im Lösungsbild Punkte wären, hätte das Band also die Form eines Sechsecks.
Ein kleinstes Vieleck, das alle Punkte aus einer vorgegebenen Menge enthält, nennt man in der Mathematik die *konvexe Hülle* dieser Punktmenge. Dabei steht *konvex* für etwas, das sich wie die konvexe Linse einer Lupe nach aussen dehnt. Eine *Hülle* ist etwas, das etwas anderes so wie die Haut den Körper umschliesst, dabei aber nicht grösser als nötig ist. Das Band der Biber um alle Büsche umschliesst diese also wie eine konvexe Hülle.
In der Informatik werden häufig konvexe Hüllen einer Menge von Punkten bestimmt:

- Mustererkennung: Ist in einem Bild ein Gesicht zu sehen?
- Handschrifterkennung: Ist ein handgeschriebenes Zeichen der Buchstabe B?
- Geographische Informationssysteme: Wie gross ist ein Überschwemmungsgebiet oder ein Flusssystem?
- Verpackungen: Was ist die kleinste Menge an Material, das genügt, einen bestimmten Gegenstand zu verpacken?

Die Informatik kennt Verfahren, welche die konvexe Hülle einer Punktmenge effizient berechnen können. Sie funktionieren also auch dann noch gut, wenn die Punktemenge sehr gross ist.

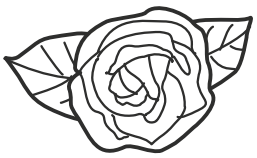
Stichwörter und Webseiten

Graph, Konvexe Hülle

- `https://de.wikipedia.org/wiki/Konvexe_Hülle`
- `https://en.wikipedia.org/wiki/Convex_hull_algorithms`
- `https://brilliant.org/wiki/convex-hull`

9. Claras Blumen

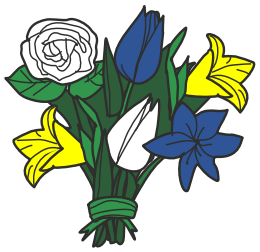
Clara mag bunte Blumensträusse und besucht daher einen Blumenladen. Da sind die folgenden Blumenarten zu finden:

Gladiolen	Lilien	Tulpen	Rosen
			

Jede Blumenart ist in den folgenden Farben erhältlich: Weiss, Blau und Gelb.
Clara möchte einen Blumenstrauss mit sechs Blumen, der die folgenden Bedingungen erfüllt:

1. Jede Farbe Weiss, Blau und Gelb soll genau zweimal vorkommen.
2. Blumen der gleichen Art sollen nicht die gleiche Farbe haben.
3. Jede Blumenart soll höchstens zweimal vorkommen.

Welcher Blumenstrauss erfüllt alle drei Bedingungen?

A)	B)	C)	D)
			

Lösung

Die richtige Antwort ist D). Im Blumenstrauss A) gibt es drei weisse Blüten (Regel 1 ist verletzt), in B) drei Rosen (Regel 3 ist verletzt), und im Blumenstrauss C) haben zwei Blüten derselben Blumenart die gleiche Farbe (Regel 2 ist verletzt).

Dies ist Informatik!

Allgemeine Informatikprobleme werden durch eine Reihe von Einschränkungen beschrieben. Die Aufgabe besteht darin, eine Lösung zu finden, die all diese Einschränkungen oder so viele wie möglich erfüllt.
In der Informatik hat man häufig komplexere Aufgaben, bei denen die Beschränkungen beispielsweise durch logische Operatoren wie die UND-Verknüpfung (*A und B* bedeutet, dass die beiden Bedingungen A und B gleichzeitig erfüllt werden müssen, wie die drei Regeln in unserer Aufgabe) oder die ODER-Verknüpfung (*A oder B* bedeutet, dass es genügt, wenn der Bedingungen erfüllt wird) gegeben sind.

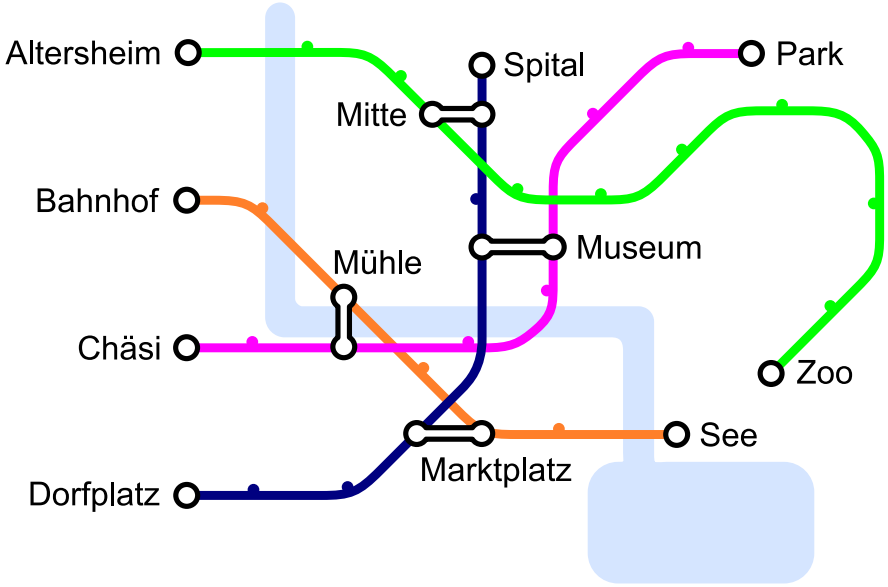
Stichwörter und Webseiten

Bedingungen, logische Operatoren

- `https://bookofbadarguments.com/de/?view=allpages`
- `https://de.wikipedia.org/wiki/Boolesche_Algebra`
- `https://www.iep.utm.edu/prop-log/`

10. Liniennetz

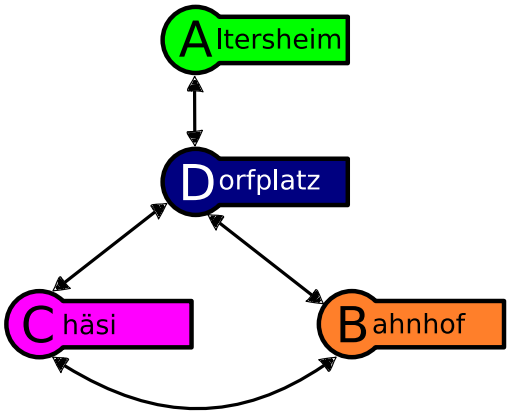
Es gibt vier Linien, die an den Stationen „Altersheim“, „Bahnhof“, „Chäsi“ und „Dorfplatz“ starten. Es gibt vier Kreuzungsstationen „Museum“, „Marktplatz“, „Mühle“ und „Mitte“, an denen man von einer Linie auf die andere wechseln kann.



Johannes möchte zum Zoo. Er weiss, dass er nur einmal die Linie wechseln muss. Bei welcher Station startet seine Linie?

Lösung

Die richtige Antwort ist die Station „Dorfplatz“. Wenn man vom Zoo rückwärts fährt, gibt es nur eine Kreuzungsstation („Mitte“), an der man auf die Linie wechselt, die an der Station „Dorfplatz“ startet.
Man kann das Liniennetz auch mit Hilfe eines Graphen darstellen, der darlegt, von welcher Linie man mit einmal Umsteigen auf welche andere Linie kommt:



Von	Nach		
Altersheim ↔ Zoo	Dorfplatz ↔ Spital		
Bahnhof ↔ See	Chäsi ↔ Park	Dorfplatz ↔ Spital	
Chäsi ↔ Park	Bahnhof ↔ See	Dorfplatz ↔ Spital	
Dorfplatz ↔ Spital	Bahnhof ↔ See	Chäsi ↔ Park	Altersheim ↔ Zoo

Wenn man also mit einmal Umsteigen auf die Linie „Altersheim ↔ Zoo“ möchte, kann man das nur von der Linie „Dorfplatz ↔ Spital“ von der Station „Dorfplatz“ her kommend, denn nur von der erreicht man die Linie „Altersheim ↔ Zoo“ mit einem Mal Umsteigen.

Dies ist Informatik!

Kommt Dir das Liniennetz irgendwie bekannt vor? Richtig, viele Liniennetze von Bussen, Trams oder Metros sehen so aus. Das ist kein Zufall sondern eine Erfindung von Harry Beck, der 1931 einen solchen Plan für das Londoner U-Bahn-System erfand.
In der Informatik nennt man einen solchen abstrakten Plan einen Graph, der aus Knoten (die Stationen) und Kanten (die Strecke zwischen zwei Stationen) besteht. In unserem Fall wird noch zwischen Knoten, die nur eine oder zwei Kanten haben (Endstationen sowie reguläre Stationen), und Knoten die mehrere Kanten haben (Kreuzungsstationen) unterschieden.
Graphen werden auch noch an vielen anderen Orten verwendet. Beziehungen von Menschen in sozialen Netzwerken, Routenplaner oder auch Shopping-Vorschläge werden mit Graphen modelliert. Es ist also eine wichtige Informatik-Kompetenz, mit Graphen umgehen zu können.

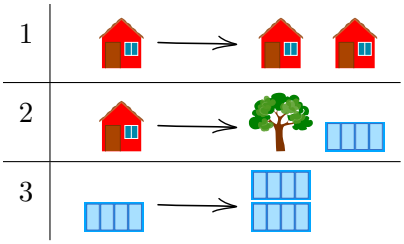
Stichwörter und Webseiten

Liniennetz, Graph

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

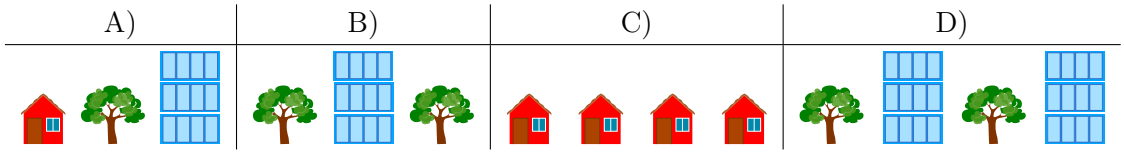
11. Planet Z

Die Bewohner des Planeten Z bauen ihre Städte immer auf die gleiche Weise. Sie beginnen eine Stadt mit einem Haus. Dann ersetzen Sie einzelne Objekte nach folgenden drei Regeln:



Wenn man zuerst Regel 1, dann Regel 2 und dann zwei Mal Regel 3 anwendet, erhält man beispielsweise die Stadt ganz rechts im Bild:

Beachte, dass die Reihenfolge der einzelnen Objekte nicht verändert werden kann.
Welche der folgenden Städte kann nicht *auf dem Planeten Z stehen?*



Lösung

Die richtige Antwort ist B). Bäume können nur durch Regel 2 in eine Stadt gebracht werden. Regel 2 besagt aber, dass rechts neben einem Baum ein Block stehen muss. In Stadt B) ist rechts neben dem rechten Baum kein Block. Es gibt auch keine Regel, mit der man Blöcke entfernen kann. Deshalb kann die Stadt B nicht auf dem Planeten Z stehen.
Stadt A) kann man durch Anwendung der folgenden Regeln bauen: 1, 2, 3 und dann wieder 3.
Stadt C) kann man bauen, indem man Regel 1 drei Mal anwendet.
Stadt D) kann man so bauen: Zuerst wendet man Regel 1 an. Anschliessend wendet man Regel 2 auf jedes Haus an und danach wendet man Regel 3 auf jeden Block zweimal an.

Dies ist Informatik!

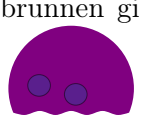
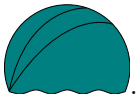
Die Regeln aus der Aufgabe nennt man Ersetzungsregeln: Ein Symbol oder ein Objekt wird durch eine Folge von anderen Symbolen oder Objekten ersetzt. Wenn immer nur ein Symbol oder Objekt ersetzt wird, nennt man ein solches Regelsystem kontextfrei. Ein Symbol oder Objekt wird durch etwas anderes ersetzt, ohne den Kontext des Symbols oder Objekts (also das, was rechts und links von dem Symbol oder Objekt ist) zu beachten.
Ersetzungsregeln verwendet man in der Informatik beispielsweise, um die Syntax einer Programmiersprache zu definieren. Die Symbole oder Objekte sind Schlüsselbegriffe und die Regeln beschreiben, wie man sie zu einem (syntaktisch) korrekten Programm zusammensetzt. In der Aufgabe sind die Symbole oder Objekte die Häuser, Bäume und Blöcke. Die Symbole und Objekte zusammen mit den Ersetzungsregeln bilden dann eine Grammatik, die eine Sprache beschreiben.
Wenn der Computer ein Programm in Maschinensprache übersetzt (compiliert) oder direkt ausführt (sogenannte „interpretierte“ Programme, häufig auch Skripte genannt), prüft er zunächst, ob der Programmtext tatsächlich den Regeln der Programmiersprache folgt. Er versucht also, mit Hilfe eines Syntaxbaumes die Ersetzungsregeln nachzubauen, die aus dem Startsymbol (in der Aufgabe ein Haus) den Programmtext (in der Aufgabe die vier verschiedenen Antwortmöglichkeiten) erzeugen. Das ist in unserem Fall einfach möglich, da *Wörter* (Abfolgen von Symbolen oder Objekten, in Fall dieser Aufgabe eine Stadt) immer grösser werden, aber nie kleiner.

Stichwörter und Webseiten

Ersetzungsregel, Grammatik, kontextfreie Sprachen

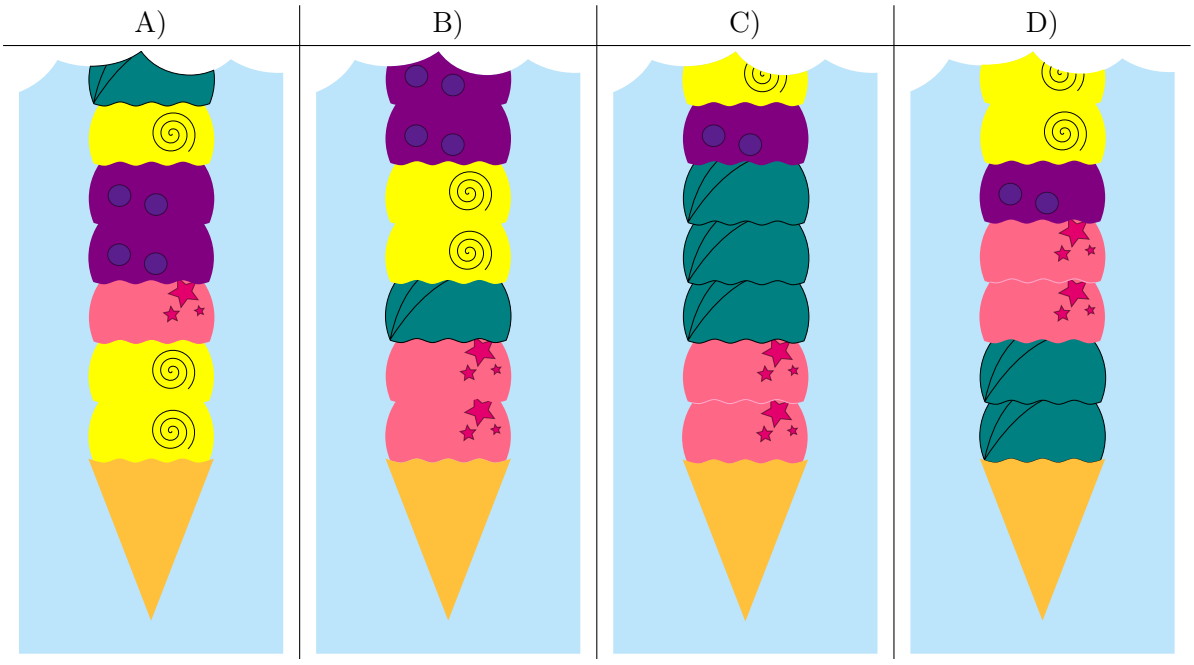
- `https://de.wikipedia.org/wiki/Produktionsregel`
- `https://de.wikipedia.org/wiki/Kontextfreie_Sprache`
- `https://de.wikipedia.org/wiki/Syntaxbaum`

12. Gelateria

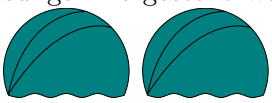
Beim Dorfbrunnen gibt es zwei Gelaterias. Sie bieten dieselben vier Glace-Sorten an:  ,  ,  und  .
Die erste Gelateria verwendet die folgenden Anweisungen für ein Cornet:

1. Nimm ein leeres Cornet.
2. Wähle eine zufällige Glace-Sorte und gib zwei Kugeln dieser Glace-Sorte hinzu.
3. Gib eine Kugel mit einer der drei anderen Glace-Sorten hinzu.
4. Wenn die gewünschte Kugelzahl erreicht ist, höre auf. Ansonsten mache mit Schritt 2 weiter.

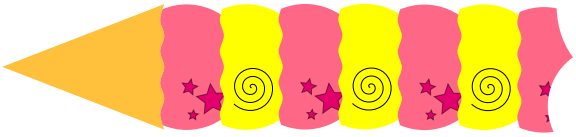
Die zweite Gelateria folgt überhaupt keinen Anweisungen.
Auf dem Bild siehst du die unteren Kugeln einiger Cornets. Welches Cornet stammt mit Sicherheit aus der zweiten Gelateria?



Lösung

Die richtige Antwort ist D). Das ist das einzige Cornet, das ganz eindeutig nicht nach den Anweisungen hergestellt worden ist. Es beginnt zwar korrekt mit zwei Kugeln der gleichen Glace-Sorte  gefolgt von einer Kugel einer anderen Glace-Sorte  . Dann jedoch kommen zwei Kugeln unterschiedlicher Glace-Sorten  , obwohl nach den Anweisungen eigentlich wieder zwei gleiche Glace-Sorten kommen müssten.
Die Antworten A), B), und C) sind nicht korrekt. Alle diese Cornets entsprechen, soweit man es erkennen kann, den Anweisungen.

Dies ist Informatik!

Muster in Cornets, Texten oder Bildern können durch Anweisungsfolgen erzeugt werden. Informatikerinnen und Informatiker entwickeln Computerprogramme, mit denen Muster oder Abweichungen von Mustern erkannt werden können. Manchmal entstehen Muster durch Wiederholung von Anordnungen. Dieses Muster  ist beispielsweise ein einfaches Muster, das durch Wiederholung von  und  entstanden ist. Solche Muster sind leicht zu erkennen. In der Aufgabe ist die Sache schwieriger, weil die Anweisungen der ersten Gelateria auch Zufallsentscheidungen enthalten.
Im allgemeinen Fall kann man niemals ganz sicher sein, ob eine Reihenfolge durch Zufall oder durch eine Folge von Anweisungen erzeugt worden ist. Bei den Beispielen dieser Aufgabe konnten wir nur bei einem Cornet sagen, dass es sicher nicht den Anweisungen entsprach und deshalb aus der zweiten Gelateria stammen musste. Man kann aber aufgrund der Zusammensetzung des Cornets niemals sicher entscheiden, ob es aus der ersten Gelateria stammt, denn es könnte ja auch zufälligerweise zu den Anweisungen passen.

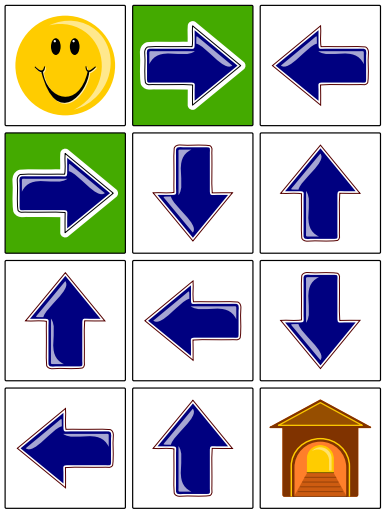
Stichwörter und Webseiten

Mustererkennung

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

13. Das Pfeil-Labyrinth

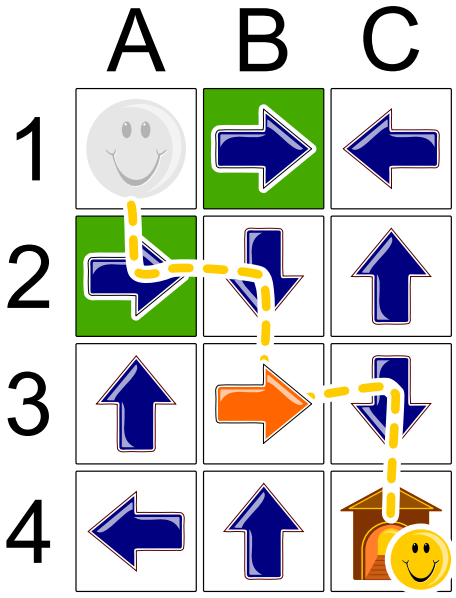
Das Smiley möchte nach Hause. Aber dazu muss es durch das Pfeil-Labyrinth. Es kann einen der beiden Eingänge (grüne Felder) benutzen. Wenn es auf einem Feld mit einem Pfeil steht, muss es das Feld in Richtung des Pfeiles verlassen. So wie die Pfeile momentan ausgerichtet sind, kann es unmöglich nach Hause finden.



Welchen Pfeil muss man ändern, so dass das Smiley nach Hause kommt?

Lösung

Der Pfeil, dessen Richtung geändert wurde, ist rot markiert. Der Weg des Smiley nach Hause ist zu erkennen: A1 – A2 – B2 – B3 – C3 – C4.



Man kann sogar beweisen, dass dies die einzig mögliche Lösung ist. Wenn man am Ziel C4 beginnt und rückwärts geht, kann man aus zwei Richtungen zum Ziel C4 gelangen: Von C3 und von B4. Der Pfeil in B4 zeigt aber in die falsche Richtung und müsste geändert werden. Weil aber kein Nachbarfeld nach B4 zeigt, müsste ein zweiter Pfeil geändert werden, was nicht erlaubt ist.
Folglich ist das Ziel nur über das andere Nachbarfeld C3 erreichbar. Der Pfeil in diesem Feld steht ja auch schon richtig und zeigt auf das Ziel. Nun gibt es in den Nachbarfeldern von C3 keinen Pfeil, der auf C3 zeigt. Einer der beiden Pfeile müsste also geändert werden, B3 oder C2. Nun gibt es aber keinen Pfeil, der auf C2 zeigt. C3 ist also von einem Eingang nicht zu erreichen, ohne dass ein weiterer Pfeil geändert wird. Deshalb muss der Weg über B3 gehen. B3 ist von einem Eingang erreichbar (A1 – A2 – B2), ohne dass weitere Pfeile geändert werden müssten. Das ist damit die einzige Möglichkeit.

Dies ist Informatik!

Bei dieser Aufgabe wurde ein Weg durch das Pfeil-Labyrinth gesucht, bei dem nur ein Pfeil geändert werden musste. Wie findet man eigentlich die Lösung zu solch einem Problem? Wie könnte das ein Computer machen? Bei der Beweisführung im letzten Abschnitt wurde eine Vorgehensweise verwendet, die man *backtracking* nennt. Das bedeutet in etwa: Man verfolgt eine Spur (engl. *track*) bis es nicht mehr weitergeht (egal ob vom Start oder vom Ziel aus). Dann nimmt man den letzten Schritt zurück und prüft von dieser Stelle aus eine andere Spur. Unmögliche Wege kann man dabei von vornherein ausschliessen und findet garantiert eine Lösung, wenn es sie gibt, weil man ja alle Möglichkeiten ausprobiert.

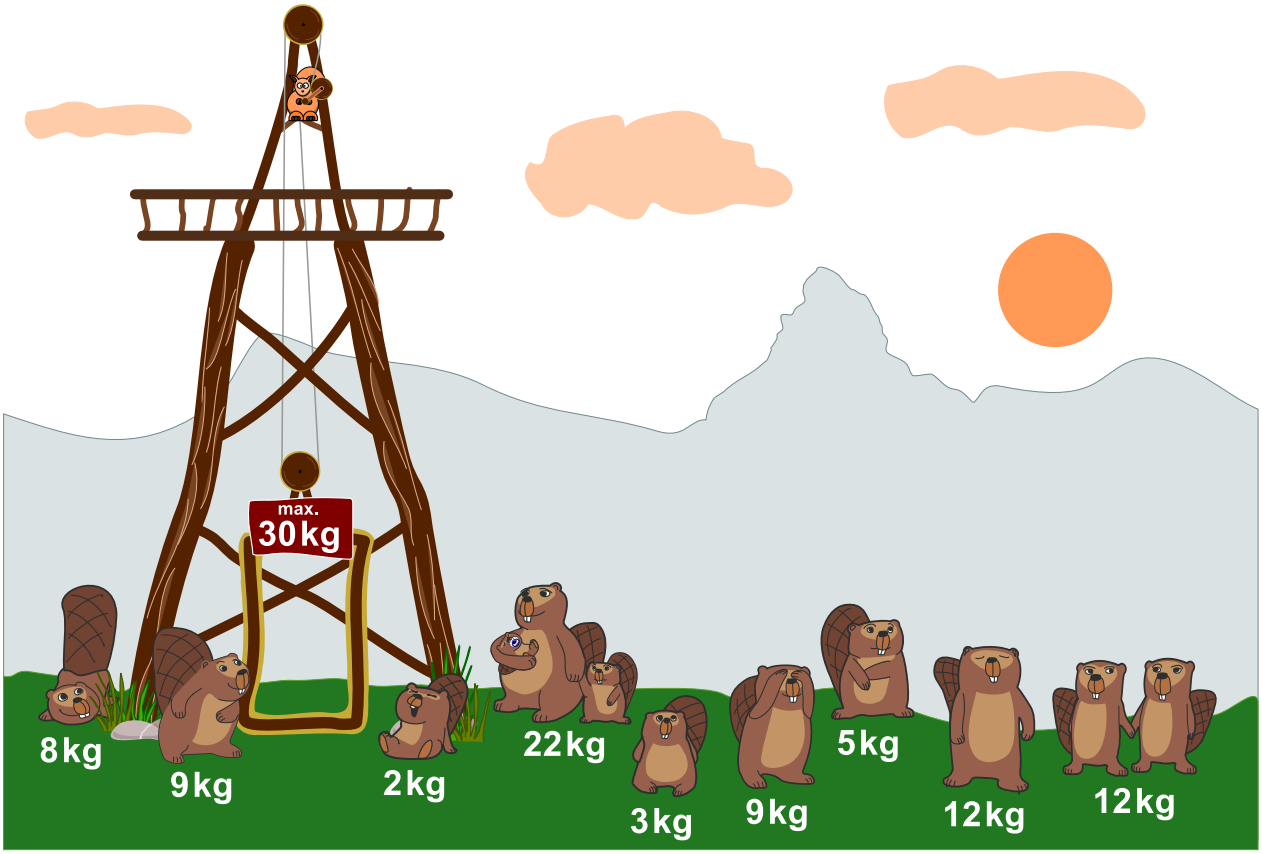
Stichwörter und Webseiten

Labyrinth, Backtracking

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

14. Ausflug mit Aussicht

Eine Biberfamilie macht einen Ausflug zu einem Aussichtsturm. Sie sind spät dran. Der Lift fährt nur noch zweimal hoch. Der Lift darf nicht mit mehr als 30 kg pro Mal beladen werden. Die Zwillinge gehen nur gemeinsam auf den Turm. Die Bibermama hält das Baby im Arm und ein Biberkind an der Hand. Es sollen aber möglichst viele Biber auf den Aussichtsturm.



Es muss schnell entschieden werden und nur die folgenden fünf Optionen sind möglich. Wer muss unten bleiben, damit möglichst viele Biber auf den Aussichtsturm können?

A) Alle können mitfahren.

B) Die Bibermama mit dem Baby und dem Biberkind.

C) Die Zwillinge und der 5 kg-Biber.

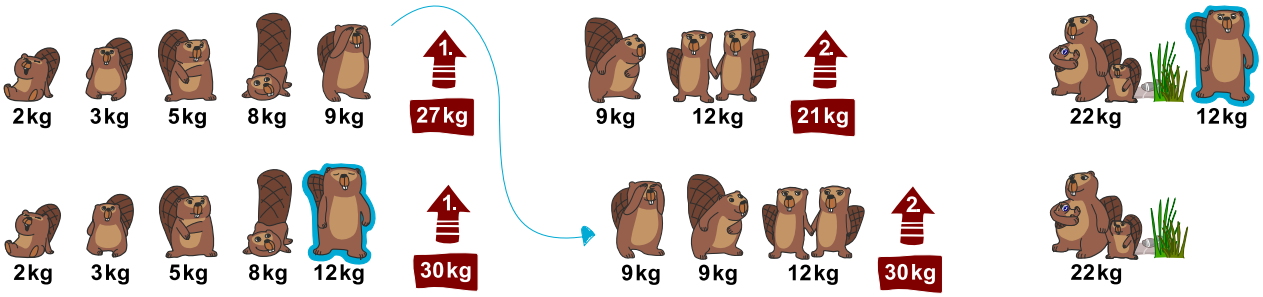
D) Die Zwillinge und die Bibermama mit dem Baby und dem Biberkind.

E) Die Bibermama mit dem Baby und dem Biberkind und der 12 kg-Biber.

Lösung

Eine richtige Antwort ist B):

Ein Lösungsweg könnte sein, die Kabinen mit möglichst vielen von den leichten Bibern zu beladen: $2\,\text{kg} + 3\,\text{kg} + 5\,\text{kg} + 8\,\text{kg} + 9\,\text{kg} = 27\,\text{kg}$ in die erste Kabine und in die zweite $9\,\text{kg} + 12\,\text{kg} = 21\,\text{kg}$ (das sind 8 Biber). Aber es passt noch ein weiterer Biber in den Lift:



Mit dieser Strategie können die beiden Fahrten optimal genutzt werden: Der 9 kg schwere Biber in der ersten Kabine wird durch den 12 kg schweren Biber ersetzt. Das Maximalgewicht von 30 kg ist erreicht. Der 9 kg schwere Biber kann in die zweite Kabine einsteigen, die dann ebenfalls ihr Maximalgewicht von 30 kg erreicht hat.
Die anderen Lösungen sind nicht möglich, weil sie entweder das Maximalgewicht überschreiten (alle Biber zusammen sind mehr als 60 kg und selbst wenn die Zwillinge und der 5 kg-Biber nicht dabei sind, sind es noch 65 kg), oder weil sie schlechter sind (wenn die Mutter mit den Kindern dabei ist, können die Zwillinge oder der 12 kg-Biber mitfahren).

Dies ist Informatik!

Die optimale Kombination als Lösung für ein Problem zu finden ist eines der klassischen Probleme der Informatik. Häufig lassen sie sich nicht schnell genug oder gar nicht innert nützlicher Zeit lösen, da es zu viele mögliche Lösungen gibt, die überprüft werden müssten. Die Informatik nennt so etwas ein praktisch unlösbares Problem.
Dieses Problem nennt man ein *Rucksack-Problem*, bei dem möglichst viele Objekte eingepackt werden müssen ohne ein Maximalgewicht zu überschreiten. Dieses und vergleichbare Probleme heissen *NP-vollständig*. Sie können nur näherungsweise gelöst werden, das bedeutet, dass man eine mögliche und gute Lösung finden kann, aber nicht notwendigerweise die optimale Lösung. Dies erreicht man, indem man sich eine sinnvolle Strategie („Heuristik“) überlegt.

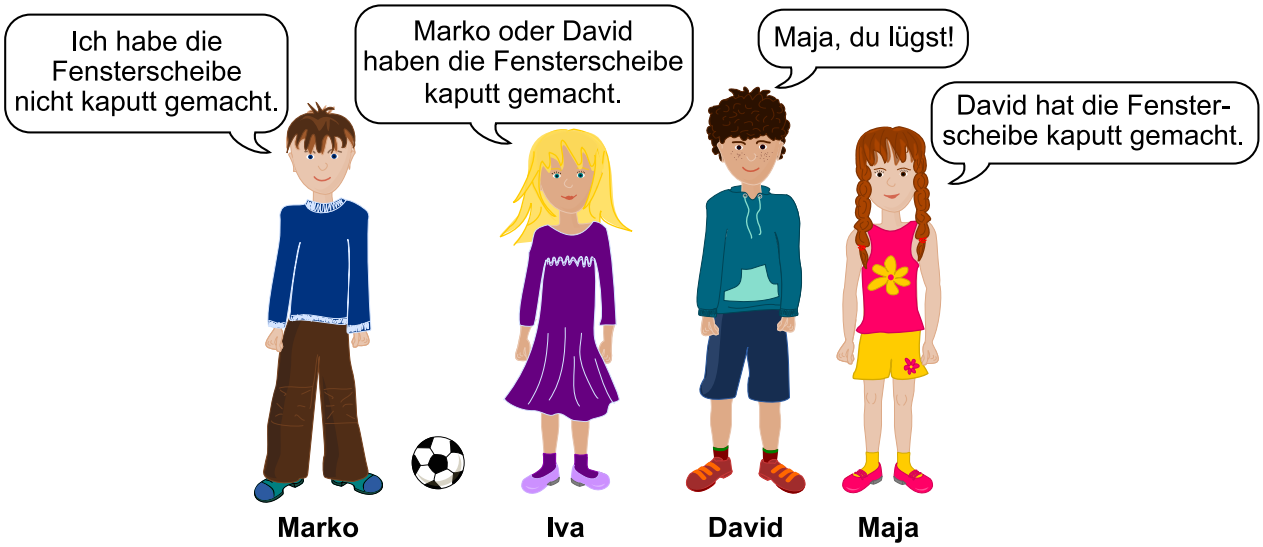
Stichwörter und Webseiten

Optimierung, Rucksack-Problem

- `https://de.wikipedia.org/wiki/Kombinatorische_Optimierung`
- `https://de.wikipedia.org/wiki/Rucksackproblem`

15. Lügen haben kurze Beine

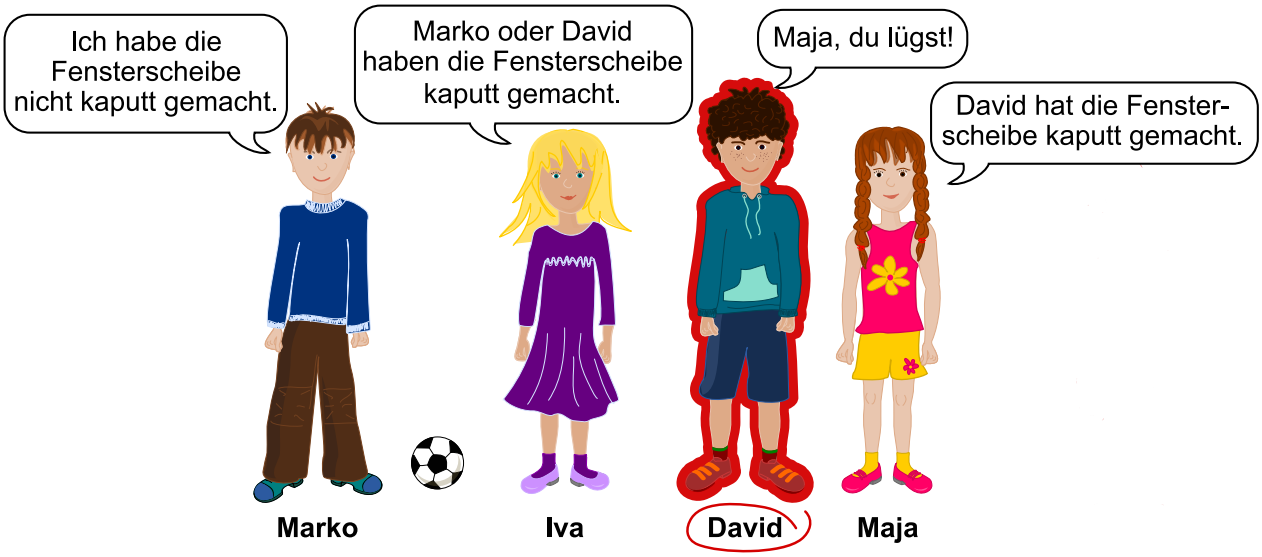
Eines schönen Tages spielen die Kinder Maja, David, Iva und Marko Fussball in der Nähe von Anna's Haus. Auf einmal zerbricht eine Fensterscheibe und Anna möchte wissen, wer es war. Anna kennt die vier Kinder und weiss, dass drei von ihnen immer die Wahrheit sagen, bei dem vierten Kind weiss sie es nicht.
Die vier Kinder sagen:



Welches Kind hat die Fensterschreibe kaputt gemacht?

Lösung

David hat die Fensterscheibe kaputt gemacht.



Die Aussagen von Maja und David können nicht beide wahr sein, einer von den beiden muss also lügen. Wenn Maja die Wahrheit sagen würde, würde also David lügen. Die Aussagen von Iva und Marko könnten ebenso korrekt sein. Wenn David aber die Wahrheit sagen würde, müsste also Maja lügen; zusätzlich müsste aber auch Iva oder Marko lügen! Das wäre aber nicht möglich, da drei Kinder immer die Wahrheit sagen.

Dies ist Informatik!

Um diese Aufgabe zu lösen musst du logisch denken. Die zugrunde liegende Logik wurde 1854 von George Boole (1815 – 1864) formuliert, der logische Aussagen auf ihre Grundlagen zurückgeführt hat und beschrieben hat.
Nach ihm ist eine Aussage entweder *wahr* oder *falsch* (Prinzip *tertium non datur*). Aussagen können mit Hilfe von *Operatoren* kombiniert werden. Einfache logische Operatoren wie *UND* oder *ODER* verbinden zwei Aussagen zu einer neuen Aussage. Es gibt auch Operatoren (wie das *NICHT*), die lediglich eine Aussage nehmen und sie verändern. Wann so kombinierte Aussagen letztlich wahr sind, kann man mit Hilfe von *Wahrheitstabellen* einfach herausfinden.
Einer der Operatoren, die das Schliessen „WENN“ → „DANN“ modelliert, ist die Subjunktion. Man spricht dann davon, dass man „logische Schlüsse zieht“. Diese wird zum Lösen dieser Aufgabe benötigt.
Computer basieren ebenfalls auf boole'schen Aussagen und einfachen logischen Operatoren, da man diese sehr einfach und in grosser Stückzahl bauen kann. Es gibt zwar auch einige Computer, die auf anderen Systemen basieren (beispielsweise ternäre Computer aus den späten 50er Jahren in Russland), diese sind aber entweder nur experimentell geblieben oder haben nie grosse Stückzahlen erreicht.

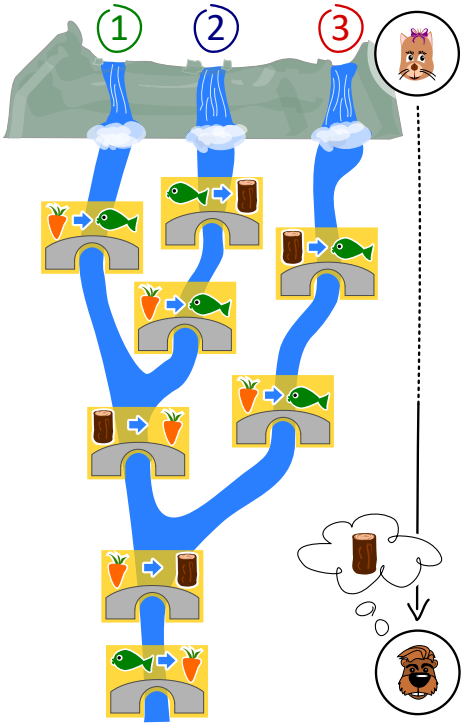
Stichwörter und Webseiten

Logisches Schliessen

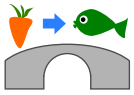
- `https://de.wikipedia.org/wiki/Satz_vom_ausgeschlossenen_Dritten`

- `https://de.wikipedia.org/wiki/George_Boole`
- `https://de.wikipedia.org/wiki/Boolesche_Algebra`
- `https://de.wikipedia.org/wiki/Subjunktion`
- `https://de.wikipedia.org/wiki/Ternärer_Computer`

16. Wasserfälle



Katja sitzt oben auf dem Berg. Der Berg hat drei Wasserfälle. Die Wasserfälle fliessen in einen Fluss.
Katja kann entweder ein Rüebli oder einen Fisch in einen der Wasserfälle fallen lassen. Der Fluss hat mehrere Brücken mit Trollen. Die Trolle ersetzen Gegenstände, die unter den Brücken durchgehen.



Wenn zum Beispiel ein Rüebli unter einer solchen Brücke durchschwimmt, ersetzen die Trolle das Rüebli durch einen Fisch.
Justus sitzt am Ende des Flusses.
Justus braucht Holz. Welchen Gegenstand muss Katja wählen und in welchen Wasserfall muss sie ihn fallen lassen, damit Justus Holz bekommt?

A) Sie lässt einen Fisch in den Wasserfall 1 fallen.

B) Sie lässt einen Fisch in den Wasserfall 2 fallen.

C) Sie lässt ein Rüebli in den Wasserfall 2 fallen.

D) Sie lässt ein Rüebli in den Wasserfall 3 fallen.

Lösung

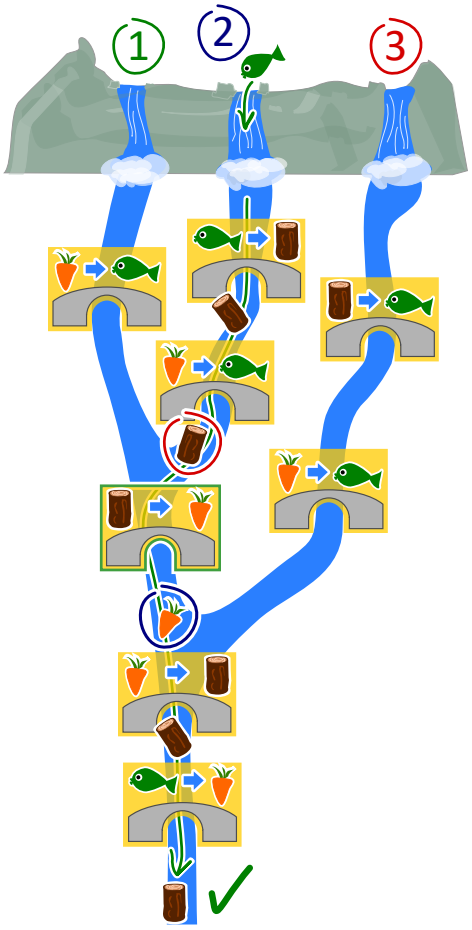
Die richtige Antwort ist: B) Sie lässt einen Fisch in den Wasserfall 2 fallen.
Was passiert bei den verschiedenen Lösungsvorschlägen:

A) Ein in den Wasserfall 1 geworfener Fisch wird nur unter der letzten Brücke gewechselt. So erhält Justus ein Rüebli.

B) Ein in den Wasserfall 2 geworfener Fisch wird in Holz, dann Rüebli und dann wieder in Holz umgewandelt. So erhält Justus Holz. Das ist somit korrekt.

C) Ein Rüebli, die in den Wasserfall 2 fallen gelassen wird, wird in Fisch und dann Rüebli geändert. So bekommt Justus ein Rüebli.

D) Ein Rüebli, die in den Wasserfall 3 fällt, wird in Fisch und dann in Rüebli verwandelt. So bekommt Justus ein Rüebli.



Ein anderer Lösungsansatz für diese Aufgabe besteht darin, rückwärts zu beginnen:
Um am Ende Holz zu bekommen, muss der gefallene Gegenstand ein Rüebli sein, wenn er die vorletzte Brücke passiert. Die einzige Möglichkeit, an diesem Punkt ein Rüebli zu haben, ist, wenn der Gegenstand die einzige gemeinsame Brücke passiert hat, die Wasserfall 1 und 2 gemeinsam haben (und nicht 3). Die einzige von den vier Antwortmöglichkeiten, an diesem Punkt Holz zu haben, ist, einen Fisch in den Wasserfall 2 fallen zu lassen.

Dies ist Informatik!

Man kann sich einen Computer als ein Gerät vorstellen, das Eingaben liest, diese verarbeitet und Ausgaben schreibt. Wie „weiss“ ein Computer aber, was zu tun ist? Die Antwort ist, dass Menschen ihm vorher befohlen haben, was zu tun ist. Das macht man, indem man Programme schreibt. Das Analysieren von Programmen nennt man Testen von Software.
Es gibt viele verschiedene Programmiersprachen, die nach unterschiedlichen Programmierparadigmen funktionieren. Ein solches Programmierparadigma ist die funktionale Programmierung. Dieser Programmierstil ist wie ein kleiner Computer, da er aus vielen Funktionen besteht, die eine Eingabe verarbeiten und daraus eine Ausgabe erzeugen. Die Brücken in dieser Aufgabe sind wie kleine Funktionen und das vollständige System ist wie ein Programm, das mit einer funktionalen Programmiersprache geschrieben wurde.

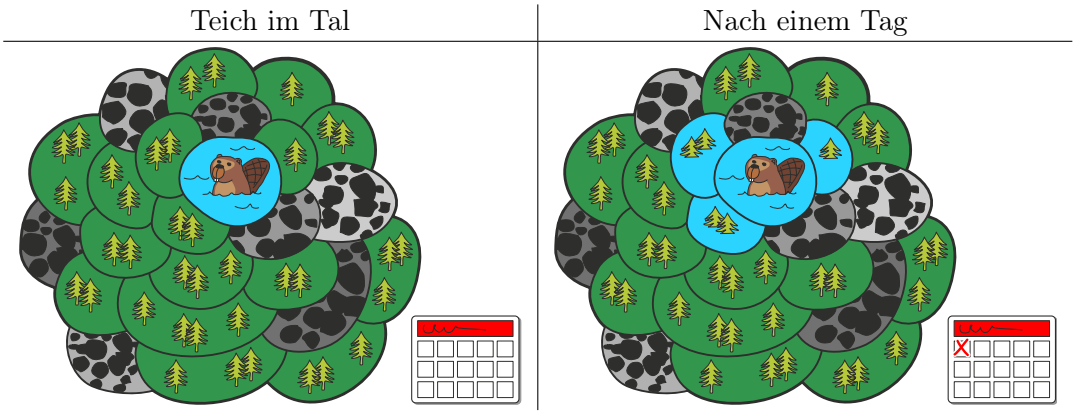
Stichwörter und Webseiten

Test von Software, Programmierparadigma, Funktionale Programmierung, Funktionen und Parameter

- `https://de.wikipedia.org/wiki/Softwaretest`
- `https://de.wikipedia.org/wiki/White-Box-Test`
- `https://de.wikipedia.org/wiki/Black-Box-Test`
- `https://de.wikipedia.org/wiki/Programmierparadigma`
- `https://de.wikipedia.org/wiki/Funktionale_Programmierung`

17. Biber-Teich

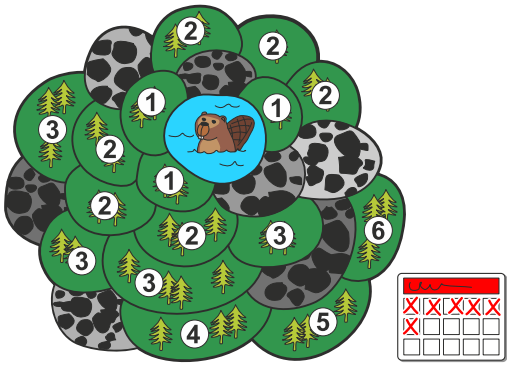
In einem Tal liegt ein kleiner Teich. Er ist umgeben von Landstücken mit Wald oder mit Felsen. Im Teich leben einige Biber.
Eines Tages wird den Bibern der Teich zu klein und nun fluten sie den Wald. An jedem Tag fluten sie alle Waldstücke, die an bereits geflutete Waldstücke angrenzen. Nach einem Tag sind drei Waldstücke geflutet.



Nach wie vielen Tagen insgesamt (also inklusive dem dargestellten ersten Tag) sind alle Waldstücke geflutet?

Lösung

Nach sechs Tagen sind alle Waldstücke geflutet.
Das Bild zeigt für jedes Waldstück, am wievielten Tag es geflutet wird: Die Waldstücke, die an den See angrenzen, sind nach einem Tag geflutet und deshalb mit der Zahl 1 markiert. Die Waldstücke, die an diese Felder angrenzen, sind mit der Zahl 2 markiert; sie sind nach zwei Tagen geflutet, und so weiter. Am sechsten Tag wird ein letztes Waldstück auf diese Weise markiert. Nach sechs Tagen ist also dieses Waldstück geflutet – und damit ist das ganze Tal geflutet.



Dies ist Informatik!

In dieser Biberaufgabe fluten die Biber ein zusammenhängendes Waldgebiet, das neben dem ursprünglichen Teich aus einzelnen benachbarten Waldstücken besteht. Das Gebiet ist zusammenhängend, weil man von jedem Waldstück in dem Gebiet über andere Waldstücke zu jedem anderen Waldstück gehen kann.
Auch ausserhalb des Tals mit dem Teich der Biber gibt es zusammenhängende Gebiete, die geflutet werden müssen. Ein einfarbiger Bereich in einem Bild ist letztlich ein zusammenhängendes Gebiet gleichfarbiger Pixel. Eine Gruppe von Jugendlichen, in denen jeder über beliebig viele andere Jugendliche mit jedem anderen Jugendlichen der Gruppe befreundet ist, ist auch ein „zusammenhängendes Gebiet“, wenn man die direkte Freundschaftsbeziehung zwischen zwei Jugendlichen als Nachbarschaft betrachtet.
Die Informatik kennt Methoden, zusammenhängende Gebiete zu ermitteln, etwa die Breitensuche oder die Tiefensuche. Mit Hilfe dieser Methoden können beispielsweise Bereiche in Bildern umgefärbt oder Gruppierungen in sozialen Netzwerken ermittelt werden.

Stichwörter und Webseiten

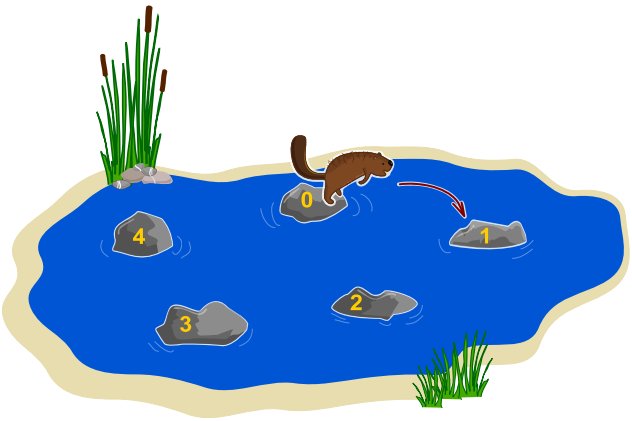
Wavefront Algorithm, Breadth-First Search

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

18. Biber-Wettbewerb

Als Vorbereitung für den alljährlichen Biber-Wettbewerb trainieren einige Biber intensiv. Die heutige Trainingseinheit besteht darin, im Uhrzeigersinn von Fels zu Fels zu springen, wie es der Pfeil zeigt. Wenn der Biber 8 mal springt, landet er am Ende auf Fels Nummer 3:

$$0 \to 1 \to 2 \to 3 \to 4 \to 0 \to 1 \to 2 \to 3.$$



Der stärkste Biber springt heute ganze 129 *mal. Er startet auf Feld* 0*. Auf welchem Fels ist er am Ende gelandet?*

Lösung

Wenn ein Biber 5 mal springt, landet er auf dem Fels von dem er gestartet ist. Wir nennen das eine „Runde“. Um herauszufinden, wo er nach 129 Sprüngen landet, müssen wir herausfinden, wie viele Runden er springt und wie viele Sprünge er danach noch vor sich hat. In diesem Fall sind es $129 = 25 \cdot 5 + 4$ Sprünge (25 Runden plus 4 Sprünge). Wenn er also 129 mal springt, endet er auf demselben Fels wie wenn er 4 mal gesprungen wäre. Daher ist die richtige Antwort 4.

Dies ist Informatik!

Du hast so etwas vielleicht schon im Mathematikunterricht kennen gelernt. Es ist dasselbe wie *der Rest der ganzzahligen Division*, manchmal auch schriftliches Teilen oder Euklidisches Teilen genannt. In dieser Aufgabe muss also der Rest der ganzzahligen Division von $129 : 5 = 25$ Rest 4 ermittelt werden. Da dies in Computern sehr häufig berechnet werden muss, gibt es alleine für das Berechnen des Rests einen eigenen Namen: es ist die Modulo-Operation. Normalerweise wird das Zeichen „%“ oder auch „mod“ als Operator verwendet. Man kann also schreiben: $129\%5 = 4$.
Diese Operation wird beispielsweise in Schleifen (so wie der Biber, der in Runden springt), wenn Variablen überlaufen, oder auch für das weit verbreitete kryptographische Verfahren RSA verwendet.

Stichwörter und Webseiten

Modulo-Operation

- `https://de.wikipedia.org/wiki/Division_mit_Rest#Modulo`
- `https://de.wikipedia.org/wiki/Schriftliche_Division`
- `https://en.wikipedia.org/wiki/Euclidean_division`
- `https://de.wikipedia.org/wiki/RSA-Kryptosystem`

19. Ferienhaus Nr. 29

Milo macht ein Praktikum in einer Ferienhaus-Siedlung. Heute soll er Nummern an den Ferienhäusern anbringen. Einige Häuser sind bereits beschriftet. Er startet bei Haus 50. Von dort aus soll er...

- ... nach links gehen, wenn die neue Nummer kleiner ist als die Nummer des Hauses, bei dem er steht, ...
- ... nach rechts gehen, wenn die neue Nummer grösser ist als die Nummer des Hauses, bei dem er steht, ...
- ... die neue Ferienhaus-Nummer anbringen, wenn das Haus unbeschriftet ist.

An welchem Haus muss Milo die neue Nummer 29 *befestigen?*

Lösung

Das richtige Ferienhaus ist das dritte von links:

Die neue Ferienhausnummer 29 ist kleiner als die Nummer von Ferienhaus 50, also muss er als erstes nach links gehen. Im Vergleich zur Nummer von Ferienhaus 24 hingegen ist 29 grösser, also muss er nach rechts gehen. 29 ist wiederum kleiner als die Nummer von Ferienhaus 34, also muss er noch einmal nach links gehen. Das nächste Ferienhaus hat keine Nummer, also befestigt er dort die Nummer 29.

Dies ist Informatik!

Die Nummerierung der Ferienhäuser entspricht einem *binären Suchbaum*, einer Datenstruktur die in der Informatik häufig genutzt wird. Mit einem binären Suchbaum kann man gespeicherte Daten schnell wiederfinden.
Ein binärer Suchbaum ist so aufgebaut, dass an jeder Kreuzung (*„Knoten“*) ein *„Element“* gespeichert ist. Nach jeder Kreuzung führen maximal zwei Wege (*„Kanten“*) zu weiteren Kreuzungen. Beim Speichern von neuen Elementen wird z. B. stets der linke Weg eingeschlagen, wenn das neue Element einen kleineren Wert hat als das Element an der Kreuzung, sonst der rechte Weg. Das neue Element wird an der ersten freien Kreuzung gespeichert.
Beim Suchen nach einem Element kann man dann an jeder Kreuzung leicht entscheiden, welchen Weg man einschlagen muss. Wenn der binäre Suchbaum *„balanciert“* ist (ein sogenannter *AVL-Baum*), muss man nach einem Schritt nur noch jeweils ungefähr die Hälfte der Kreuzungen durchsuchen. Das führt dazu, dass 1000 Elemente in nur 10 Schritten durchsucht werden können, 1′000′000 Elemente in 20 Schritten oder 1′000′000′000 Elemente in 30 Schritten (also n Elemente in $\log_2(n)$ Schritten).

Stichwörter und Webseiten

Binärer Suchbaum, AVL-Baum

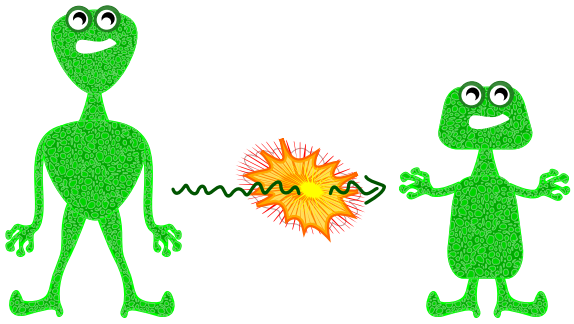
- `https://de.wikipedia.org/wiki/Binärer_Suchbaum`
- `https://de.wikipedia.org/wiki/AVL-Baum`

20. Aliens!

Ein Alien besteht aus einem Kopf, einem Rumpf, zwei Armen und zwei Beinen. Ein Alien kann durch folgende Befehle verändert werden, dabei ist es auch möglich, dass ein Körperteil mehrfach verändert wird.

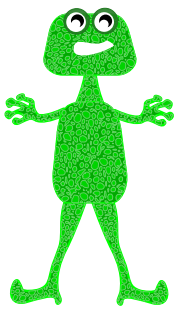
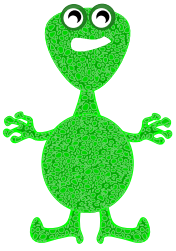
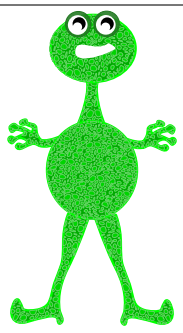
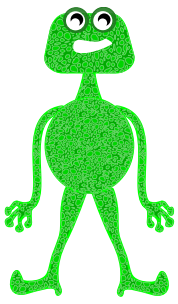
$K(r)$	Der Kopf wird rund.	
$K(4)$	Der Kopf wird viereckig.	
$K(3)$	Der Kopf wird dreieckig.	
$R(r)$	Der Rumpf wird rund.	
$R(4)$	Der Rumpf wird viereckig.	
$R(3)$	Der Rumpf wird dreieckig.	
$A(+)$	Die Arme werden lang.	
$A(-)$	Die Arme werden kurz.	
$B(+)$	Die Beine werden lang.	
$B(-)$	Die Beine werden kurz.	

Die einzelnen Befehle werden von links nach rechts ausgeführt. Zum Beispiel ergibt $K(r)$, $R(4)$, $K(4)$, $A(-)$, $B(-)$ das folgende Alien:

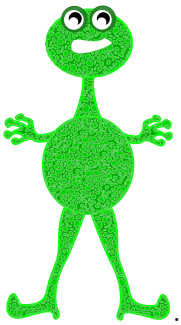


Wie schaut das Alien nach den folgenden Befehlen aus?

$K(3)$, $B(+)$, $R(3)$, $A(+)$, $K(r)$, $A(-)$, $R(r)$

A)	B)	C)	D)
			

Lösung

Die richtige Antwort ist C)  .
Für jeden Körperteil gilt nur der letzte Befehl. Ein vorhergehender Befehl für den selben Körperteil ist in der Endform des Aliens nicht erkennbar, da ein folgender Befehl die Veränderung wirkungslos macht.
Daher ist das Ergebnis ein Alien mit einem runden Rumpf ($R(r)$), kurzen Armen ($A(-)$), einem runden Kopf ($K(r)$) und langen Beinen ($B(+)$). Die anderen Aliens unterscheiden sich von der Lösung C) in mindestens zwei Eigenschaften, so dass sie offensichtlich falsch sind.

Dies ist Informatik!

Beim Ausführen eines Programms werden die Befehle der Reihe nach abgearbeitet. Nachfolgende Befehle können dabei die Wirkung vorhergehender Befehle wieder verändern.
In Computerprogrammen passiert das häufig bei Variablen, in denen Werte gespeichert werden, die sich während der Laufzeit des Programms mehrfach verändern. Man kann daher die Form der vier Körperteile jeweils wie eine Variable ansehen, in der die aktuelle Form gespeichert ist. Der Befehl „$K(r)$“ bewirkt dann, dass in der Variablen für die Kopfform der neue Wert „r“ gespeichert wird.
Die Notation „$K(r)$“ ist funktional gedacht. Man ruft die Funktion „$K()$“ auf und gibt ihr das Argument „r“ mit. Dies wird häufig verwendet, da die Funktion „$K()$“ nebenbei noch prüfen kann, ob das mitgegebene Argument überhaupt gültig ist. Wenn das nicht nötig ist oder wenn die Variable nur lokal verwendet wird, kann man sie auch direkt überschreiben, hierfür verwendet man häufig den Zuweisungsoperator „=“. Man würde dann beispielsweise „$K = r$“ im Programm schreiben.

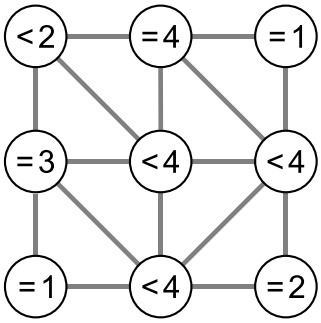
Stichwörter und Webseiten

Variable, Sequenz

- `https://de.wikipedia.org/wiki/Variable_(Programmierung)`
- `https://de.wikipedia.org/wiki/Strukturierte_Programmierung`

21. Nachbarn

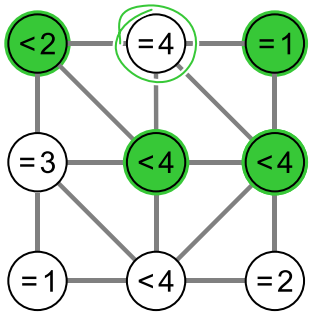
Im Bild unten sind neun Kreise zu sehen, die teilweise miteinander verbunden sind. Eine Verbindung macht sie zu Nachbarn. Durch Anklicken können Kreise ausgewählt werden. Ein ausgewählter Kreis ist grün dargestellt, ein nicht ausgewählter Kreis weiss.
In jedem Kreis steht ein Ausdruck, der anzeigt, wie viele seiner Nachbarn ausgewählt werden sollen. „$= 3$“ bedeutet, dass genau drei der Nachbarn ausgewählt sein sollen. „< 4“ bedeutet, dass maximal drei der Nachbarn ausgewählt sein sollen.



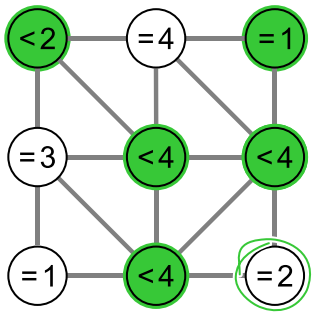
Wähle die Kreise so aus, dass die Bedingungen in allen neun Kreisen gleichzeitig erfüllt sind.

Lösung

Der Kreis oben in der Mitte hat ein „= 4“ und somit genau vier Nachbarn, also müssen alle vier ausgewählt sein:



Ebenso hat der Kreis unten rechts ein „= 2“ und genau zwei Nachbarn, also müssen beide ausgewählt sein (der Kreis in der Mitte rechts ist bereits ausgewählt):



Damit sind bereits alle Bedingungen in allen neun Kreisen erfüllt.
Wenn man einen weiteren Kreis auswählen würde, würden Bedingungen verletzt:

- Wenn der „= 4“-Kreis oben in der Mitte ausgewählt würde, wäre die Bedingung des „= 1“-Kreises oben rechts nicht mehr erfüllt.
- Wenn der „= 3“-Kreis in der Mitte links ausgewählt würde, wäre die Bedingung des „< 2“-Kreises oben links nicht mehr erfüllt.
- Wenn der „= 1“-Kreis unten links ausgewählt würde, wäre die Bedingung des „= 3“-Kreises in der Mitte links nicht mehr erfüllt.
- Wenn der „= 2“-Kreis unten rechts ausgewählt würde, wäre die Bedingung des „< 4“-Kreises in der Mitte rechts nicht mehr erfüllt.

Dies ist Informatik!

Wie viele Versuche braucht man, um das Problem zu lösen? Wenn man einfach alle möglichen Lösungen ausprobiert, hat man für jeden der 9 Kreise unabhängig 2 verschiedene Einstellungen, also $2^9 = 512$ verschiedene Möglichkeiten. Einfach alle Möglichkeiten auszuprobieren, nennt man einen *brute-force-Ansatz*. Für jede dieser Möglichkeiten müsste dann überprüft werden, ob die Bedingungen erfüllt sind.
Sinnvoller ist es in diesem Fall jedoch, logisch und konsequent vorzugehen. Man sucht zuerst Kreise, deren Bedingungen eindeutig erfüllt werden können. Das sind zum Beispiel alle Kreise, die die

Bedingung „$= n$“ haben und genau n Verbindungen, also n Nachbarn haben. Von da an könnte man mit logischem Folgern weitermachen: schauen, ob es unerfüllte Bedingungen gibt, die man nur auf eine Art und Weise erfüllen kann. So kann man mit viel weniger Aufwand zur richtigen Lösung kommen. Im allgemeinen Fall kann man so auch feststellen, dass es keine Lösung gibt oder zumindest eine von mehreren Möglichkeiten findet. Eine analytische Vorgehensweise, bei der von allen Lösungsmöglichkeiten nur die vielversprechenden betrachtet werden, wird als *heuristisch* bezeichnet.

Stichwörter und Webseiten

Nachbarschaft in Graphen, logisches Vorgehen

- `https://de.wikipedia.org/wiki/Nachbarschaft_(Graphentheorie)`
- `https://de.wikipedia.org/wiki/Heuristik#Informatik`
- `https://de.wikipedia.org/wiki/Brute-Force-Methode`

22. Computerspiel

Andrea hat ein Computerspiel in der Schule programmiert. Die Spielregeln sind ganz einfach:
Das Spiel besteht aus mehreren Spielrunden. In jeder Spielrunde fällt ein Blatt. Der Biber versucht das Blatt zu fangen, bevor es den Boden erreicht. Um zu gewinnen, muss der Biber 15 Blätter fangen, bevor 4 Blätter den Boden berühren.
Die Länge des Spiels wird in der Anzahl der Spielrunden gemessen.
Im folgenden Beispiel verliert der Biber nach 6 Spielrunden, weil das Maximum von 4 nicht gefangenen Blättern erreicht ist. Die Länge dieses Beispiels beträgt 6 Spielrunden.

Spielrunde	Resultat	Spielstand – Total Anzahl Blätter	
		Gefangen	Nicht gefangen
1	gefangen	1	0
2	nicht gefangen	1	1
3	gefangen	2	1
4	nicht gefangen	2	2
5	nicht gefangen	2	3
6	nicht gefangen	2	4

Wie lange kann ein Spiel maximal dauern?

A) 4 Spielrunden

B) 15 Spielrunden

C) 18 Spielrunden

D) 19 Spielrunden

E) 20 Spielrunden

F) Die Spiellänge ist unbegrenzt.

Lösung

Um das längstmögliche Spiel zu finden, müssen wir alle Situationen kombinieren, in denen das Spiel weitergeht. Dazu kombinieren wir die maximal gefangenen Blätter vor Spielende (14 Spielrunden) mit den maximal nicht gefangenen Blättern vor Spielende (3 Spielrunden). Danach wird entweder ein 15. Blatt gefangen oder ein 4. Blatt verloren. Daher ist die maximale Länge $15+3 = 14+4 = 18$ Runden und die richtige Antwort ist C).
Die Antwort A) „4 Runden“ wäre die Mindestlänge des Spiels (wenn alle Blätter nicht gefangen werden).
Die Antwort B) wäre die Mindestlänge, um das Spiel zu gewinnen (wenn alle Blätter gefangen werden).
Die Antworten D), E) und F) sind falsch, da das Maximum der gefangenen oder das Maximum der nicht gefangenen Blätter vorher erreicht würde.

Dies ist Informatik!

Bei der Programmierung eines Spiels müssen die Regeln klar definiert sein. Die Auswirkungen der Regeln müssen vollständig verstanden werden, so dass das Spiel so aufgebaut werden kann, dass Gewinnen oder Verlieren möglich ist (ausreichende Blätter sind verfügbar), und dass das Spiel weder zu kurz noch zu lang dauert.
Ein Spiel, das aus mehreren Runden besteht, ist ein Prozess. Informatiker sind Spezialisten in der Modellierung und Beschreibung von Prozessen. Eine der Hauptaufgaben besteht darin herauszufinden, was alles passieren kann, und wie lange ein Prozess laufen kann.

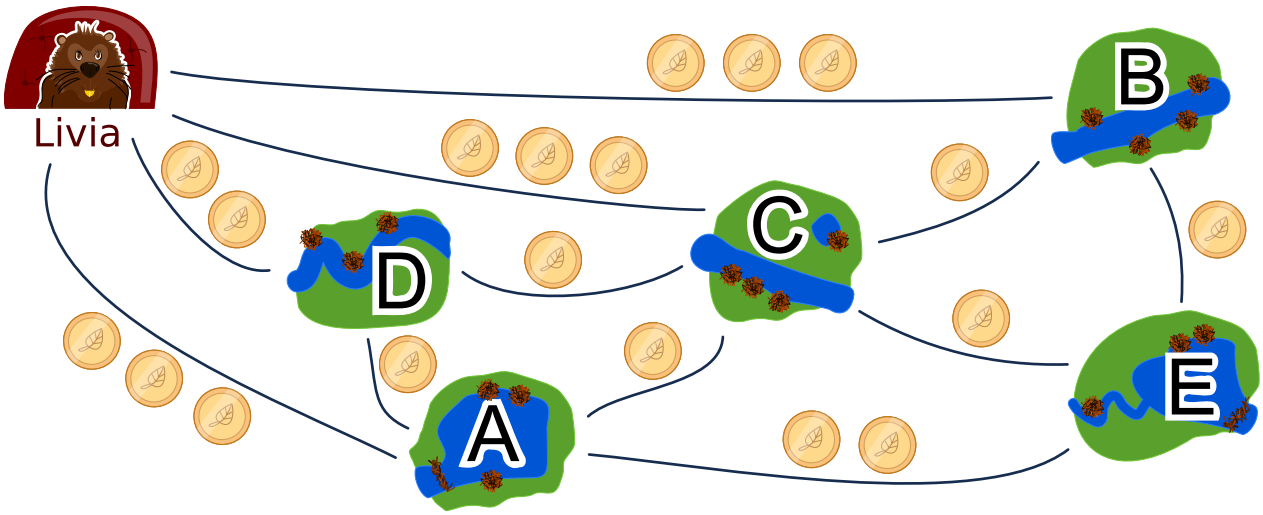
Stichwörter und Webseiten

Analyse, Verifizierung und Validierung von Software

- `https://en.wikipedia.org/wiki/Software_verification`
- `https://de.wikipedia.org/wiki/Verifizierung_und_Validierung`

23. Biberbesuch

Livia möchte alle ihre Freunde in den Dörfern A, B, C, D und E mit öffentlichen Verkehrsmitteln besuchen. Sie besucht alle ihre Freunde auf einer einzigen Reise, ohne ein Dorf mehr als einmal zu besuchen. Am Ende ihrer Reise kehrt sie nach Hause zurück. Der Fahrpreis jeder Linie ist unten angezeigt.



Ein möglicher Weg, ihre Freunde zu besuchen ist:

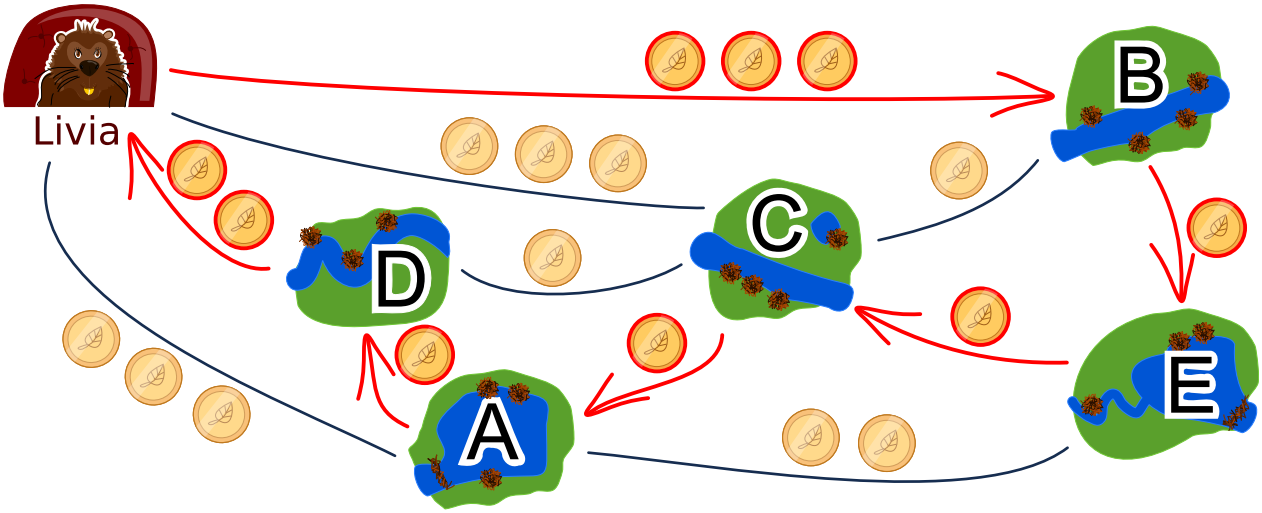
$$\text{Start} \to \text{B} \to \text{E} \to \text{A} \to \text{D} \to \text{C} \to \text{Start}.$$

Dieser Weg kostet $3 + 1 + 2 + 1 + 1 + 3 = 11$ Bibermünzen.
In welcher Reihenfolge muss Livia die Freunde besuchen, damit sie möglichst wenige Münzen bezahlen muss?

Lösung

Es gibt zwei optimale Lösungen:

- Start → B → E → C → A → D → Start
- Start → D → A → C → E → B → Start



Die zwei Lösungen sind bis auf die Richtung gleich und kosten 9 Bibermünzen. Es gibt keine bessere Lösung, denn von Livias Zuhause aus kannst du einmal den Zwei-Bibermünzen-Weg und dann einen Drei-Bibermünzen-Weg gehen. Zu den vier weiteren Knoten gehören vier Wege, die jeweils mindestens eine Bibermünze kosten, was bereits 9 Bibermünzen ergibt.
Alle anderen Lösungen kosten mehr:

- Kosten von 10 Bibermünzen: Start → A → D → C → E → B → Start
- Kosten von 10 Bibermünzen: Start → A → E → B → C → D → Start
- Kosten von 10 Bibermünzen: Start → B → C → E → A → D → Start
- Kosten von 10 Bibermünzen: Start → B → E → A → C → D → Start
- Kosten von 10 Bibermünzen: Start → B → E → C → D → A → Start
- Kosten von 10 Bibermünzen: Start → C → B → E → A → D → Start
- Kosten von 10 Bibermünzen: Start → D → A → E → B → C → Start
- Kosten von 10 Bibermünzen: Start → D → A → E → C → B → Start
- Kosten von 10 Bibermünzen: Start → D → C → A → E → B → Start
- Kosten von 10 Bibermünzen: Start → D → C → B → E → A → Start
- Kosten von 11 Bibermünzen: Start → B → E → A → D → C → Start
- Kosten von 11 Bibermünzen: Start → C → D → A → E → B → Start

Eine Methode, den günstigsten Rundweg zu finden, besteht darin, einen Weg zu gehen, der die minimale Menge an Bibermünzen kostet, und dann von dort aus eine Lösung zu finden.

Dies ist Informatik!

Nach guten oder sogar optimalen Lösungen zu suchen, ist eine der grundlegenden Aufgaben der Informatik. Wir können die Beschreibung dieser Optimierungsaufgabe in einem Graphen visualisieren, in dem Freunde Knoten und die Strassen Kanten sind. Die Aufgabe besteht darin, alle Knoten genau einmal so zu besuchen, dass die Summe der Kantengewichte (die Kosten in Bibermünzen) minimal sind. Dies ist ähnlich dem berühmten Travelling Salesman Problem (TSP).
Diese Art von Problemen sind normalerweise sehr schwierig mit einem Computer zu lösen. Um zu vermeiden, dass jede einzelne Lösung ausprobiert werden muss, kann man eine gute Heuristik verwenden (eine Heuristik ist zum Beispiel, zuerst den kürzesten Weg zu nehmen) und alle Lösungen, die schlechter werden, zu streichen. In diesem Fall erlauben wir, dass jeder Knoten nur einmal besucht werden darf. Wenn wir einen Knoten mehr als einmal besuchen dürfen, wird das Problem tatsächlich schwieriger, weil wir viel mehr Alternativen in Betracht ziehen müssen.

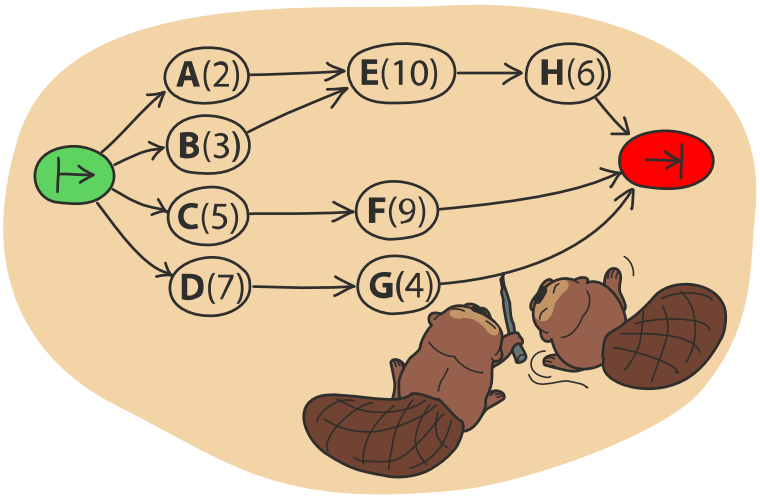
Stichwörter und Webseiten

Optimierung, Problem eines Handlungsreisenden

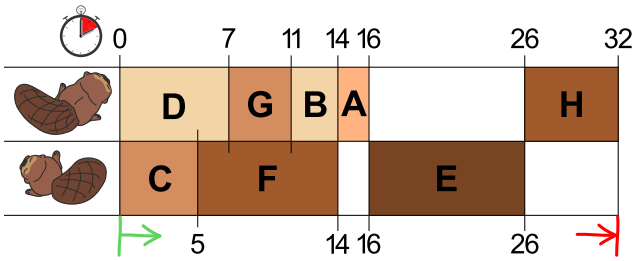
- `https://de.wikipedia.org/wiki/Problem_des_Handlungsreisenden`
- `https://de.wikipedia.org/wiki/Optimierungsproblem`

24. Zwei Biber bei der Arbeit

Zwei Biber bauen einen Damm und müssen dazu acht Aufgaben lösen: Bäume fällen, von den Stämmen die Äste entfernen, Stämme ins Wasser bringen, und so weiter. Für jede Aufgabe gibt es einen Buchstaben als Namen und eine Zahl in Klammern, die die nötige Anzahl der Arbeitsstunden angibt.
Einige Aufgaben können erst dann begonnen werden, wenn bestimmte andere Aufgaben bereits vollständig gelöst worden sind. Diese Abfolge wird durch die Pfeile dargestellt. Die Biber können parallel verschiedene Aufgaben bearbeiten, es kann aber immer nur einer an einer Aufgabe arbeiten.



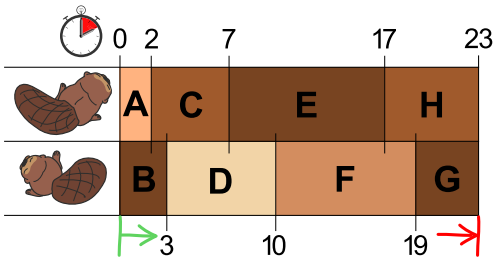
Die Abbildung unten zeigt einen möglichen Arbeitsplan der beiden Biber, der 32 Stunden benötigt. Es geht aber schneller!



Was ist die kürzeste Zeit, in der die Biber einen Damm bauen können?

Lösung

Es sind mindestens 23 Stunden erforderlich.
Das Bild in der Aufgabe zeigt einen möglichen Arbeitsplan der beiden Biber. Dort hat der erste Biber eine lange Pause von 10 Stunden und der zweite Biber zwei Pausen über insgesamt 8 Stunden. Wenn beide die ganze Zeit arbeiten würden, wären sie schneller fertig.



Wenn man darauf achtet, dass die beiden grössten Aufgaben E(10) und F(9) nicht von demselben Biber ausgeführt werden, findet man leicht einen Arbeitsplan, der mit 23 Stunden auskommt. Schneller geht es nicht, denn die beiden Biber arbeiten ohne Pause.

Dies ist Informatik!

Um einen kürzesten Arbeitsplan zu finden, wäre eine Möglichkeit, sich an die folgende Regel zu halten: „Wähle unter den noch verfügbaren Aufgaben immer die mit den meisten Arbeitsstunden“.
In der Informatik nennt man eine solche Strategie “greedy” (engl. für „gierig“). Man löst zuerst die Teilaufgaben, die einen möglichst grossen Fortschritt im Hinblick auf die Gesamtlösung des Problems bedeuten.
In vielen Fällen ist “greedy” eine gute Strategie, aber manchmal – wie bei dieser Aufgabe – funktioniert sie nicht so gut. Diese Aufgabe wurde absichtlich so konstruiert, dass die “greedy”-Strategie nicht funktioniert. Das Finden solcher ungünstigen Problemstellungen ist jedoch auch wichtig: In der theoretischen Informatik beispielsweise sucht man für Computerprogramme gezielt nach dem ungünstigsten Fall (“worst case”), um den Zeitbedarf von Algorithmen besser abschätzen zu können.
Eigentlich gäbe es nur einen sicheren Weg, die beste Lösung zu finden: Man probiert alle denkbaren Arbeitspläne aus, die den vorgegebenen Regeln entsprechen. Bei grösseren Projekten kann aber die Anzahl der Möglichkeiten so gross sein, dass die Entscheidung zu viel Zeit benötigt. Da kommt dann eine Strategie wie “greedy” ins Spiel, denn mit ihr kann man einfach Lösungen finden, die zumindest hinreichend gut sind.

Stichwörter und Webseiten

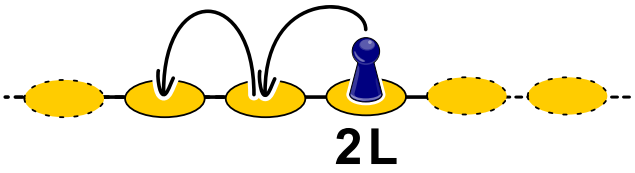
Scheduling, Greedy-Algorithmus

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

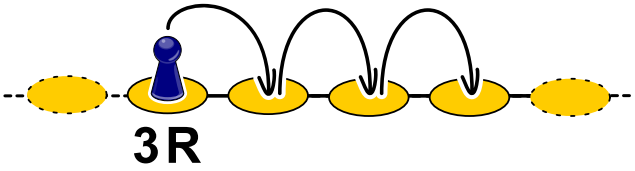
25. Hüpfspiel

Wie bei jedem Hüpfspiel muss man auch hier Felder nach bestimmten Regeln abhüpfen. Bei diesem Hüpfspiel gehört zu jedem Feld eine Regel. Es gibt drei Arten von Regeln:

- nL: n Felder nach links hüpfen, 2L bedeutet also, zwei Felder nach links zu hüpfen:

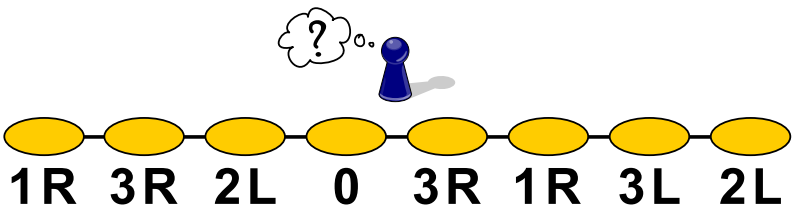


- nR: n Felder nach rechts hüpfen, 3R bedeutet also, drei Felder nach zu rechts hüpfen:



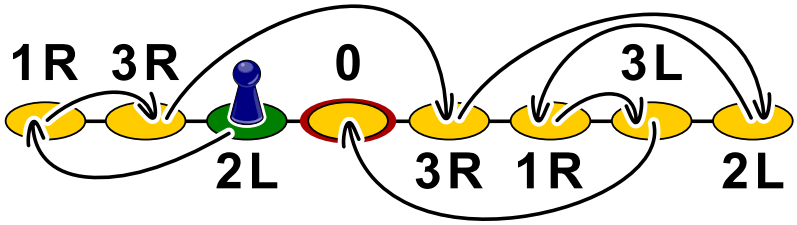
- 0: nicht mehr weiter hüpfen.

Auf welchem Feld muss man starten, damit man nach dem Spiel auf jedem Feld einmal gewesen ist?



Lösung

Wenn man auf dem dritten Feld von links („2L“) startet, wird man am Ende auf jedem Feld einmal gewesen sein.
Man kommt darauf, indem man für das Feld „0“ das Feld sucht, von welchem aus man es erreichen kann. Das ist in diesem Fall das zweite Feld von rechts („3L“). Dieses wiederum erreicht man vom dritten Feld von rechts („1R“), dieses wiederum vom Feld ganz rechts („2L“), dieses vom vierten Feld von rechts („3R“), das vom zweiten Feld von links („3R“), dieses nun vom Feld ganz links („1R“) und zuletzt bleibt nur noch das dritte Feld von links („2L“), das tatsächlich wie gewünscht das Feld ganz links erreichen lässt.



Dieses Markieren der Hüpfwege mit Pfeilen macht aus den Feldern einen gerichteten Graphen, den man nun lediglich von 0 an rückwärts durchgehen muss, um beim dritten Feld von links als Startfeld zu landen.

Dies ist Informatik!

In der Informatik funktioniert das Speicherprinzip *„Verkettete Liste“* ähnlich wie die Hüpffelder: im Arbeitsspeicher werden Objekte mit Verweisen auf die Speicheradresse des nächsten Objekts gespeichert. So kann die Speicherverwaltung ein einzelnes Objekt beliebig im Arbeitsspeicher hinterlegen und muss nicht zeitaufwendig für alle Objekte einen zusammenhängenden Platz im Arbeitsspeicher erstellen. Der Programmierer wiederum muss sich um nichts kümmern, ausser dass er bei der Speicherverwaltung einen Speicher in der entsprechenden Grösse reserviert.
Was passiert aber, wenn Objekte im Speicher nicht mehr gebraucht werden? Während sich früher die Programmierer darum kümmern mussten, dass der Speicher wieder freigegeben wird (was leider häufig schief ging, so dass Programme mit der Zeit immer mehr Speicher verschwendeten und dann irgendwann aus Speichermangel abstürzten), haben moderne Programmiersprachen hierfür eine Art Müllabfuhr, die *„Garbage Collection“*, die regelmässig überprüft, ob Objekte im Speicher noch von anderen Objekten referenziert werden (ob es also noch Verweise auf sie gibt). Da manchmal grosse Strukturen von Objekten nicht mehr referenziert werden, muss genau wie im Beispiel zurückverfolgt werden, von welchem „Startfeld“ aus auf die anderen Objekte verwiesen wird.

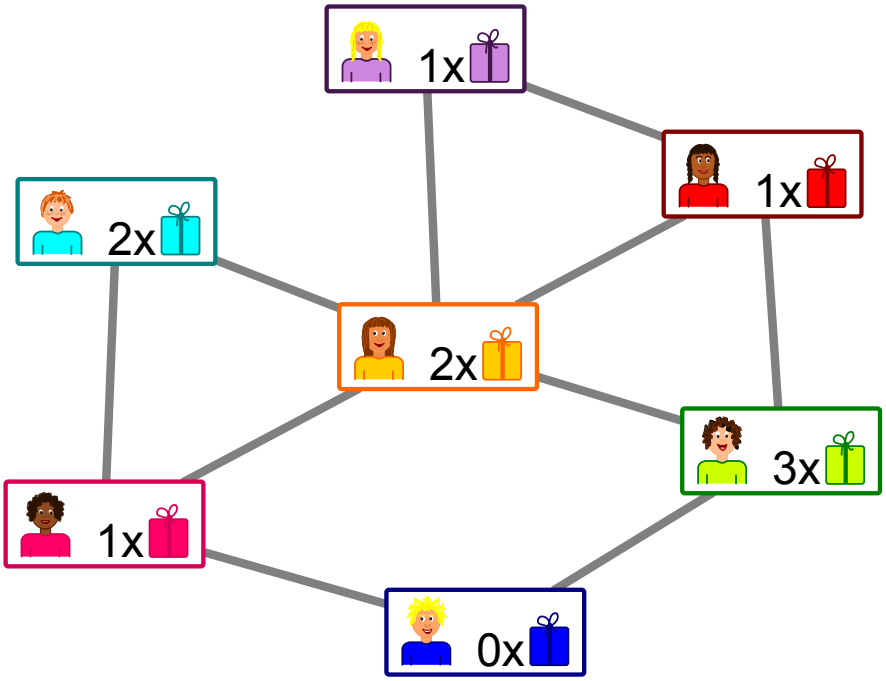
Stichwörter und Webseiten

Schlange, Speicherverwaltung, GOTO

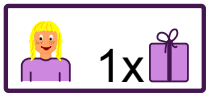
- `https://de.wikipedia.org/wiki/Garbage_Collection`
- `https://de.wikipedia.org/wiki/Erreichbarkeitsproblem_in_Graphen`

26. Geschenke

Das Bild zeigt die Freundschaften zwischen den Kindern in einem Haus. Eine Linie zwischen zwei Freunden bedeutet: Diese Kinder sind Freunde.



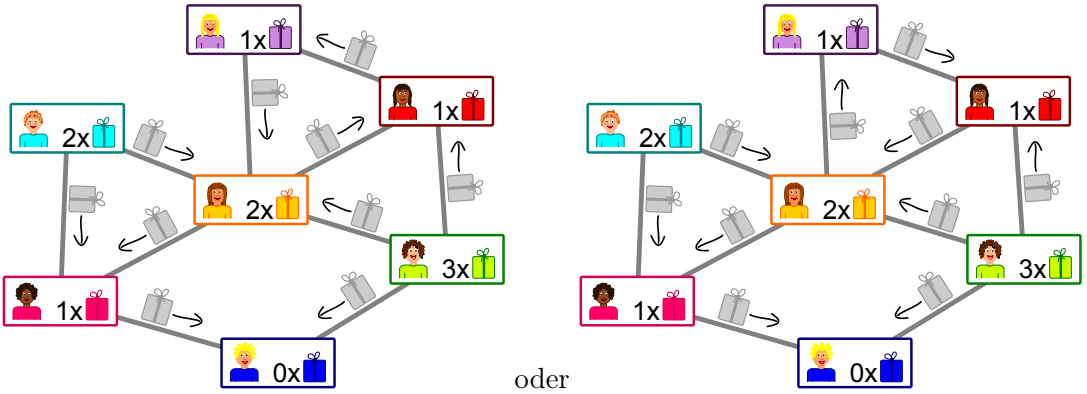
Die Hausbewohner planen ein Kinderfest mit Geschenken. Bei allen Paaren von Freunden soll ein Kind dem anderen Kind ein Geschenk besorgen.

Im Bild steht, wie viele Geschenke das Kind besorgen kann: 

bedeutet, dass das Kind ein Geschenk besorgen kann.
Entscheide für jedes Freundespaar, wer das Geschenk besorgt. Dabei soll kein Kind mehr Geschenke besorgen müssen, als es besorgen kann.

Lösung

Es gibt zwei Möglichkeiten, wie man für alle Paare von Freunden die „Schenkrichtung“ angeben kann, ohne dass ein Kind mehr Geschenke besorgen muss, als es besorgen kann.



Es lohnt sich bei dem Kind unten anzufangen: Es kann kein Geschenk besorgen und erhält deshalb je ein Geschenk von seinen Freundinnen links und rechts. Damit hat die Freundin links ihr Geschenk vergeben und erhält selbst je ein Geschenk von den anderen beiden Kindern, mit denen sie befreundet ist. Für die anderen Freundespaare sind die „Schenkrichtungen“ also klar.
Die einzige Auswahlmöglichkeit, die es noch gibt, ist, ob bei den drei Kindern oben rechts im Uhrzeigersinn oder gegen den Uhrzeigersinn geschenkt wird.

Dies ist Informatik!

Die Freundschaften zwischen den Kindern bilden ein Netzwerk aus Knoten (die Kinder) und Kanten (die Freundschaftsbeziehungen). Ähnlich sind „soziale Netzwerke“ aufgebaut mit Millionen von Benutzenden. Es gibt jedoch einen Punkt, in dem sich diese Systeme grundlegend unterscheiden können: In manchen gibt es wechselseitige „Freundschaften“, in denen die Verbindungen keine Richtung haben, so wie in dieser Aufgabe. In anderen gibt es „Anhänger“ (engl. “follower”), so dass die Verbindungen eine Richtung haben: Wenn du beispielsweise einer berühmten anderen Nutzerin „folgst“, muss sie nicht unbedingt auch dir folgen.
In dieser Aufgabe sollen die Freundschafts-Verbindungen „Schenkrichtungen“ bekommen. Das ist ein neuer Aspekt, denn die „Schenk-Kapazitäten“ der Kinder sind begrenzt und setzen so indirekt der Wahl der Richtungen Grenzen. Das Ziel ist, in jeder Freundschaft ein Geschenk zu schenken, ohne dass die Schenk-Kapazität eines Kindes überschritten wird. In der Informatik gibt es ähnliche Probleme: In einem Netzwerk (etwa die Kabel, die das Internet bilden) sind die Kapazitäten der Verbindungen begrenzt. Innerhalb dieser Grenzen aber ist es am besten, wenn diese Kapazitäten voll ausgenutzt werden.
Das Problem des maximalen Flusses in Netzwerken lässt sich effizient lösen. Da seine Struktur der Struktur unserer Aufgabe gleicht, lassen sich die Lösungsmethoden auch hier anwenden. So etwas kommt in der Informatik häufig vor: Ein Problem wird in ein anderes Problem mit der gleichen Struktur umgewandelt, in der man das Problem bereits einfach lösen konnte.

Stichwörter und Webseiten

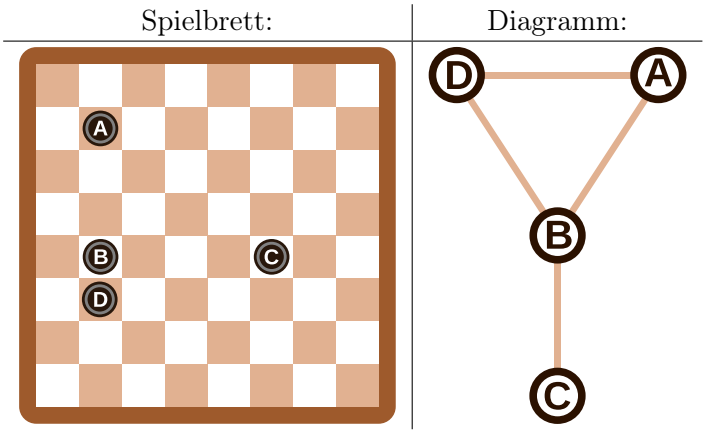
Netzwerkfluss, Problemreduktion

- `https://en.wikipedia.org/wiki/Maximum_flow_problem`

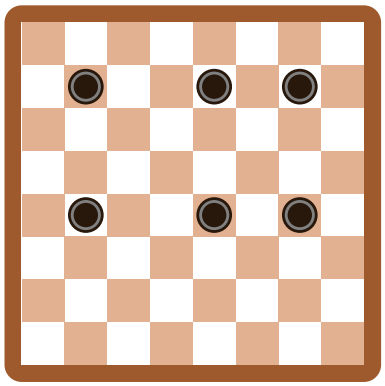
27. Zeilen und Spalten

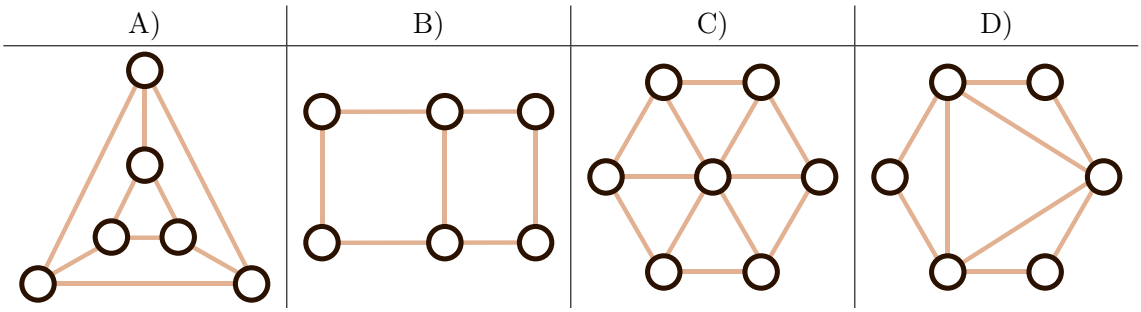
Aus den Spielsteinen auf dem Spielbrett wurde das Diagramm rechts vom Spielbrett so konstruiert, dass...

- ...jeder Spielstein durch einen Kreis dargestellt wird, und...
- ...2 Spielsteine im Diagramm durch eine Linie verbunden sind, wenn sie auf dem Brett in derselben Zeile oder in derselben Spalte liegen.



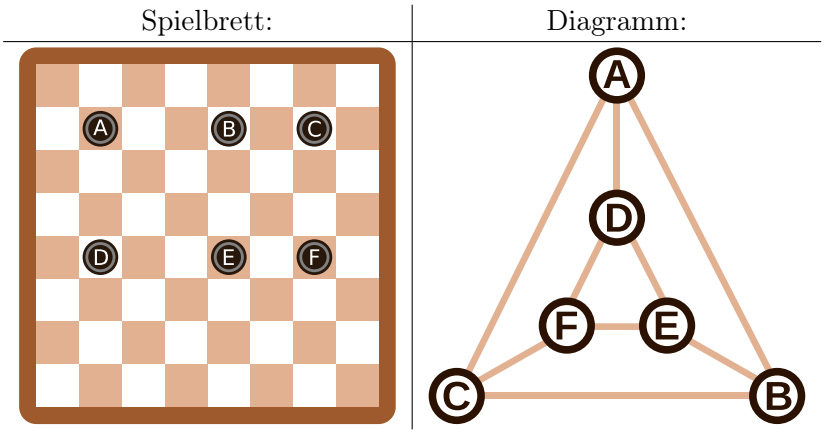
Die Spielsteine auf dem Spielbrett und die Kreise im Diagramm sind in diesem Beispiel mit Buchstaben bezeichnet, damit der Zusammenhang deutlich wird.
Welches Diagramm entspricht dem folgenden Spielbrett mit 6 Spielsteinen?



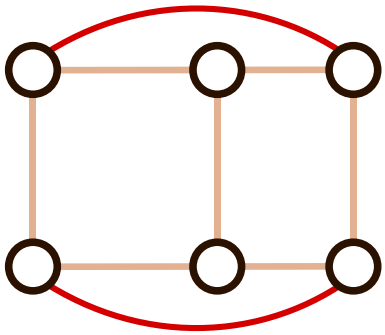


Lösung

Das Diagramm A) ist richtig. Das erkennt man in folgender Grafik, in der die Spielsteine mit Buchstaben markiert sind:



Man kann die Diagramme B), C) und D) mit folgender Beobachtung ausschliessen: Jeder Spielstein hat 2 andere Spielsteine in der selben Reihe und einen anderen Spielstein in der selben Spalte. Das heisst, dass jeder Spielstein im Diagramm mit genau $2 + 1 = 3$ anderen Spielsteinen verbunden sein muss. Das ist bei den anderen Diagrammen nicht der Fall. Zudem hat das Diagramm C) mit 7 Kreisen einen Kreis zu viel.
Das Diagramm B) ist nicht richtig, obwohl es dem Spielbrett ähnelt. Die äusseren 4 Kreise sind nur mit 2 anderen Kreisen verbunden. Um dieses Diagramm richtig zu machen, müsste man 2 zusätzliche Linien wie im folgenden Diagramm einzeichnen:



Dies ist Informatik!

In der Informatik werden solche Diagramme oft verwendet um das Wesentliche von Problemstellungen darzustellen. Solche Diagramme werden *Graphen* genannt. Die Kreise heissen *Knoten* und die Linien werden *Kanten* genannt.
Bei Graphen kommt es nur darauf an, welche Knoten mit welchen anderen Knoten durch Kanten verbunden sind. Die Anordnung der Knoten und auch die Form der Kanten spielt keine Rolle. Der selbe Graph kann daher auf unterschiedliche Weise dargestellt werden, wie wir bereits oben gesehen haben: Sowohl der Graph in Antwort A) als auch das letzte Bild in der Antworterklärung sind korrekte Lösungen und repräsentieren denselben Graph.
Graphen sind eine Form der Abstraktion. Sie repräsentieren das Wesentliche eines Problems. In unserem Fall kann man mit Hilfe des Graphen zum Beispiel das Problem „Was ist die kleinste Anzahl von Spielsteinen, die man entfernen muss, damit keine 2 Spielsteine in derselben Reihe oder derselben Spalte liegen?“ lösen. Eine wesentlicher Teil der Arbeit eines Informatikers liegt darin eine gute Repräsentation der Problemstellung zu finden, die zur Lösung des Problems hilfreich ist.

Stichwörter und Webseiten

Graph

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

28. Büchertausch

Jeder der drei Biber hat einen Tisch mit zwei Büchern. Sie wollen die Bücher durch Vertauschen benachbarter Bücher sortieren. Das machen die Biber gemeinsam in Runden. In jeder Runde darf ein Buch höchstens einmal bewegt werden.
Es gibt zwei verschiedene Typen von Runden, die immer abwechselnd durchgeführt werden:

A. Alle Biber dürfen (aber müssen nicht) die beiden Bücher auf seinem Tisch vertauschen (Beispiel A).

B. Die beiden linken Biber dürfen (aber müssen nicht) das rechte ihrer beiden Bücher mit linken Buch auf dem rechten Nachbartisch vertauschen (Beispiel B).

Die Biber beginnen mit folgender Anfangssituation:

Die erste Runde ist vom Typ A.
Wie viele Runden sind insgesamt mindestens notwendig um die Bücher zu sortieren, d.h. in die Reihenfolge 1, 2, 3, 4, 5, 6 *zu bringen?*

A) drei Runden

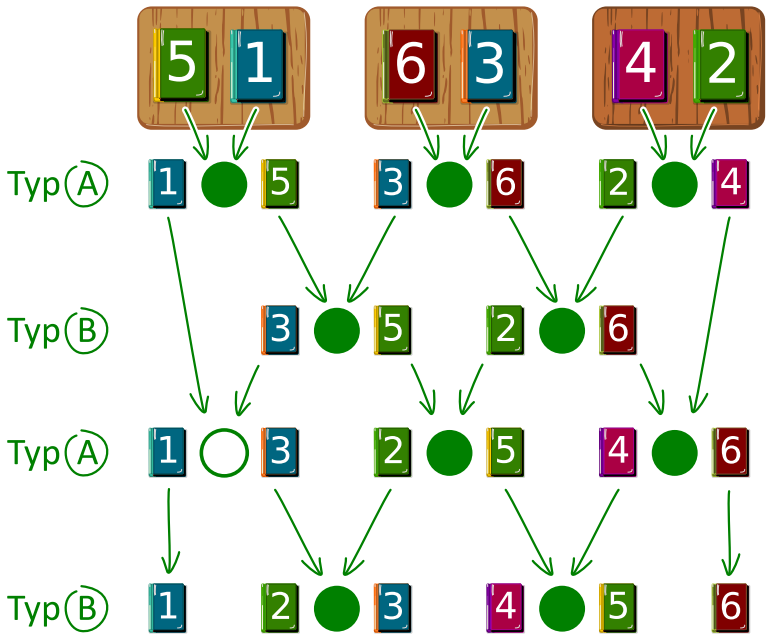
B) vier Runden

C) fünf Runden

D) sechs Runden

Lösung

Die richtige Antwort ist B).
Die Abbildung zeigt, wie die Bücher durch Platztausch sortiert werden können. Die Biber verfolgen eine „greedy-Strategie“ (“greedy” bedeutet „gierig“). Das heisst, sie versuchen bei jedem Schritt der Lösung ein Stückchen näher zu kommen. Sie vergleichen benachbarte Bücher, die gerade getauscht werden dürfen. Wenn diese beiden Bücher schon sortiert sind (das linke Buch hat eine kleinere Nummer als das rechte Buch), dann machen sie nichts. Ansonsten vertauschen sie die Bücher.
In der ersten Runde (Typ A) werden auf jedem Tisch die beiden Bücher vertauscht. In der zweiten Runde (Typ B) werden benachbarte Bücher von Nachbartischen getauscht, in der dritten Runde (Typ A) werden nur auf den beiden rechten Tischen die Bücher getauscht und in der vierten Runde (Typ B) werden alle benachbarten Bücher von Nachbartischen getauscht. Somit sind die Bücher sortiert. Schneller geht es nicht, denn das Buch 5 muss beispielsweise um vier Positionen in vier Runden nach rechts getauscht werden.



Dies ist Informatik!

Das Sortieren in dieser Aufgabe ist ein Beispiel für einen parallelen Algorithmus. Mehrere Akteure arbeiten zur gleichen Zeit an der Lösung eines Problems. Paralleles Sortieren kann durch ein Sortiernetz wie in der Abbildung dargestellt werden. Ein Sortiernetz besteht aus gerichteten Kanten, die durch Pfeile dargestellt werden, und Knoten, die durch die Kreise dargestellt werden.
In jeder Runde werden jeweils die beiden durch einen Kreis markierten Bücher verglichen und bei Bedarf vertauscht. Dabei können Vergleiche von Kreisen nebeneinander zeitgleich stattfinden. Wenn man den Pfeilen eines Buchs von oben nach unten folgt, kann man erkennen, wie es nach einigen Vertauschungen allmählich seine richtige Position in der angestrebten Reihenfolge einnimmt.

Stichwörter und Webseiten

Paralleles Sortieren, Sortiernetz

- `https://en.wikipedia.org/wiki/Sorting_network`

29. Soundex

Donald möchte Wörter nach ihrem Klang codieren. Er macht dazu Folgendes:

- Behalte den ersten Buchstaben bei.
- Streiche von allen anderen Buchstaben A, E, I, O, U, H, W und Y.
- Ersetze die restlichen Buchstaben wie folgt:
  - B, F, P oder V → 1
  - C, G, J, K, Q, S, X oder Z → 2
  - D oder T → 3
  - L → 4
  - M oder N → 5
  - R → 6
- Wenn nun zweimal oder öfters dieselbe Ziffer auftaucht, und die Buchstaben, die zu diesen Ziffern geführt haben, im Original direkt nebeneinander standen, behalte die Ziffer nur einmal. Dies gilt auch, wenn der erste Buchstabe durch diese Ziffer codiert würde, dann wird nur dieser Buchstabe behalten.
- Am Ende werden nur die ersten vier Zeichen (inkl. des ersten Buchstabens) notiert, fülle gegebenenfalls am Ende mit Nullen auf.

Die folgenden Wörter werden so codiert:

Euler → E460
Gauss → G200
Heilbronn → H416
Kant → K530
Lloyd → L300
Lissajous → L222

Welcher Code wird für das Wort „Hilbert“ erstellt?

A) H410

B) B540

C) H041

D) H416

Lösung

Der erste Buchstabe ist ein H, also ist das erste Zeichen des Codes ebenfalls ein H.
Danach werden alle A, E, I, O, U, H, W und Y gelöscht, nun muss also noch Hlbrt übersetzt werden.
Das Ersetzen der Buchstaben durch ihre Entsprechungen ergibt H4163.
Es gibt keine nebeneinanderliegenden Buchstaben mit demselben Code, also muss auch nichts gelöscht werden.
Nun werden die ersten vier Zeichen aufbewahrt, also ist H416 die richtige Antwort.

Dies ist Informatik!

Das Soundex-Verfahren, genauer gesagt der amerikanische Soundex, wurde bereits vor 100 Jahren von Robert C. Russel und Margaret King Odell entwickelt und patentiert. Es wurde dazu verwendet, ähnlich klingende Wörter in der englischen Sprache insbesondere auch ähnliche Namen von Personen zu finden. Das funktioniert, weil die Gruppen von Buchstaben, die demselben Code zugeordnet werden, ähnlich klingen: B, F, P und V sind Lippenlaute, C, G, J, K, Q, S, X und Z sind Gaumenlaute und Zischlaute, D und T sind Zahnlaute, L ist ein langer Fliesslaut, M und N sind Nasenlaute und R ist ein kurzer Fliesslaut.
Da es sehr einfach ist und nicht nur in der englischen Sprache relativ gute Resultate gibt, wird es häufig zur phonetischen Suche, also zur Suche nach ähnlich klingenden Wörtern verwendet. Es ist auch als Standard in vielen Datenbanken eingebaut.
Die Beispiele oben stammen von Donald Knuth, einem der ganz grossen Informatiker des 20. Jahrhunderts, der bis heute an seinem Buch „The Art of Computer Programming“ arbeitet. Im Band 3 „Sorting And Searching“ findet sich das beschriebene Verfahren.

Stichwörter und Webseiten

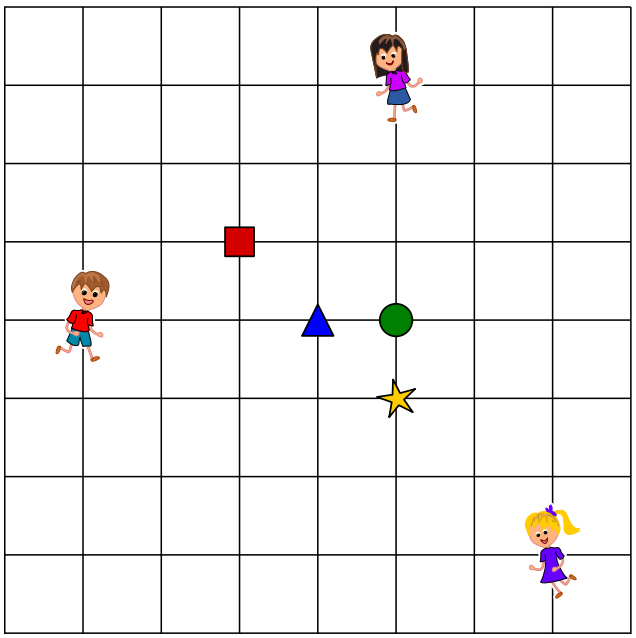
Phonetische Suche, Soundex

- `https://www.functions-online.com/soundex.html`
- `https://de.wikipedia.org/wiki/Soundex`
- `https://de.wikipedia.org/wiki/Laut`
- `https://www-cs-faculty.stanford.edu/~knuth/taocp.html`
- `http://www.highprogrammer.com/alan/numbers/soundex.html`

30. Drei Freunde

 Alice, Bob  und Céline  wohnen in La Chaux-de-Fonds. Sie haben auf dem Plan ihre Wohnorte markiert. Sie möchten einen Treffpunkt festlegen, zu dem die Summe ihrer Weglängen möglichst klein ist. Als Weglänge gilt die Zahl der Teilstrecken von Kreuzung zu Kreuzung.

 ,  ,  und  sind mögliche Treffpunkte. Der kürzeste Weg von Alice  zum Treffpunkt  hat beispielsweise die Länge 4.



Welches ist der Treffpunkt für den die Summe der Weglängen aller drei Freunde möglichst klein ist?

A)	B)	C)	D)
			

Lösung

Die richtige Antwort ist C)  . Die Summe der Weglängen der Freunde zum grünen Kreis beträgt: $3+4+5=12$.

Nicht richtig:  . Die Summe der Weglängen der Freunde zum roten Quadrat beträgt: $4+3+8=15$.

Nicht richtig:  . Die Summe der Weglängen der Freunde zum blauen Dreieck beträgt: $4+3+6=13$.

Nicht richtig:  . Die Summe der Weglängen der Freunde zum gelben Stern beträgt: $4+5+4=13$.

Dies ist Informatik!

Um von den vier Treffpunkten den besten zu bestimmen, muss für jeden von ihnen die Summe der Weglängen der drei Freunde berechnet werden. Der Treffpunkt mit der kleinsten Summe ist der beste Treffpunkt. Das ist einfach und kostet nicht viel Zeit – es geht sogar im Kopf. Bei einer grossen Anzahl von Freunden und einer grossen Zahl möglicher Treffpunkte wird daraus ein Optimierungsproblem, dessen Lösung in der Regel nicht in sinnvoller Zeit zu bestimmen ist.
In der Informatik gibt es Verfahren, die bei solchen Optimierungsproblemen in sinnvoller Zeit zumindest eine annähernd optimale Lösung finden, die beispielsweise nur noch maximal 1% von der optimalen Lösung entfernt ist. Wenn es anstelle vom Treffen von Freunden Zeitungsausträger sind, die an einem Ort ihre zu verteilenden Zeitungen abholen, kommt es bei 10 Minuten Wegzeit auf 6 Sekunden (1% von 10 Minuten) nicht an.
Ein solches Verfahren für eine annähernd optimale Lösung ist die lokale Suche: Um für eine grosse Zahl von Freunden einen annähernd optimalen Treffpunkt zu finden, wird bei der lokalen Suche zunächst irgend ein Vorschlag gesucht, der eventuell sogar zufällig gewählt wird. Davon ausgehend werden die Wegsummen für den ersten Vorschlag und für benachbarte Treffpunkte verglichen und der beste Treffpunkt in dieser Nachbarschaft bestimmt. So kann man sich dem Optimum annähern.
Warum wohnen die drei Freunde eigentlich in La Chaux-de-Fonds? La Chaux-de-Fonds ist zusammen mit Le Locle seit 2009 UNESCO Welterbe, und zwar nicht nur aufgrund der Geschichte der Uhrmacherei, sondern auch weil die Städte im 19. Jahrhundert nach Bränden ähnlich wie ein Schachbrett geplant und wieder aufgebaut wurden.

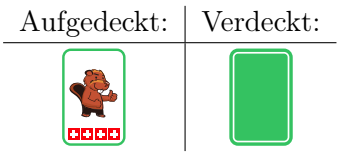
Stichwörter und Webseiten

Optimierungsproblem, lokale Suche

- `https://de.wikipedia.org/wiki/Lokale_Suche`
- `https://whc.unesco.org/en/list/1302`

31. Karten drehen

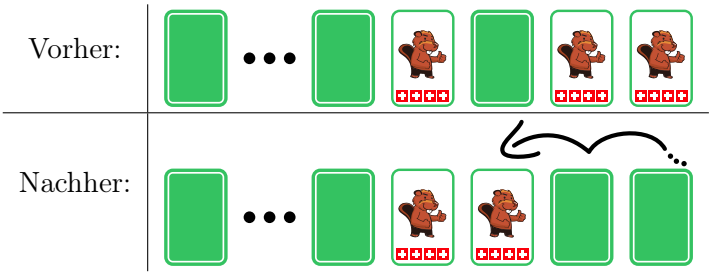
Jemand schenkt dir einen Satz gleicher Spielkarten. Die Karten sehen so aus:



Mit diesen Karten kannst du „Drehen“ spielen. Dafür legst du eine Reihe Karten vor dir aus.
In einem Spielzug gehst du diese Karten von rechts nach links so durch:

- Ist die aktuelle Karte aufgedeckt, drehe sie um.
- Ist die aktuelle Karte verdeckt, drehe sie um. Damit ist der Spielzug beendet, die übrigen Karten bleiben unverändert.

Ein Spielzug könnte zum Beispiel sein:



Die beiden rechten verdeckten Karten drehst du um. Die nächste Karte ist verdeckt. Du deckst sie auf und damit ist der Spielzug beendet.
Diesmal beginnst das Spiel mit 16 verdeckten Karten.

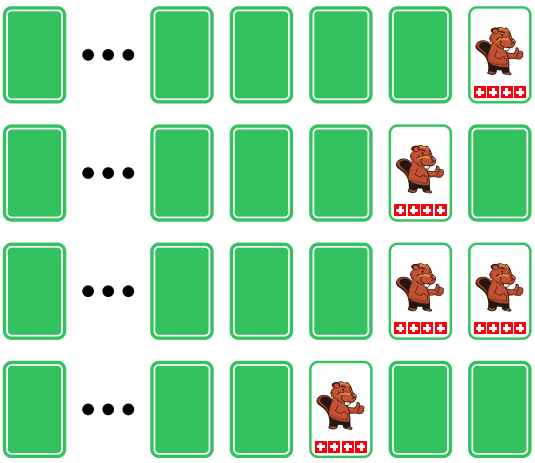
Wie viele Karten sind nach 16 Spielzügen aufgedeckt?

Lösung

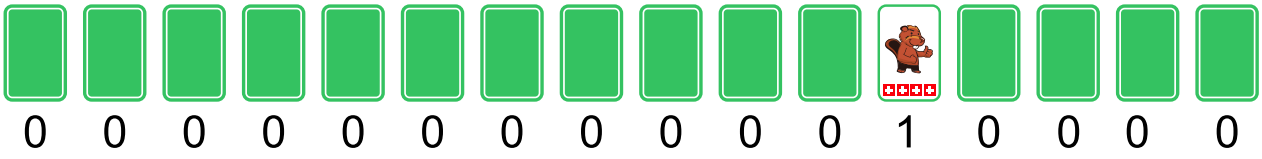
Es ist genau eine Karte aufgedeckt.
Man kann sich die Reihe von gleichen Karten, die entweder verdeckt oder aufgedeckt sein können, als Binärzahl vorstellen. Binärzahlen bestehen nur aus den Ziffern 0 und 1. Eine verdeckte Karte stellt beispielsweise die Ziffer 0 dar und eine aufgedeckte Karte die Ziffer 1.
Analog zum üblichen Zehnersystem gibt jede Stelle einer Binärzahl an, ob die passende Zweierpotenz in den Wert der Zahl einzurechnen ist oder nicht. Ist z. B. die dritte Stelle (von rechts) einer Binärzahl mit einer 1 besetzt, ist die dritte Zweierpotenz zum Wert der Zahl zu addieren – also 2^2, denn $1 = 2^0$.
Binärzahlen werden auf diese Weise um 1 hochgezählt. Man beginnt mit der Ziffer ganz rechts:

- Ist die aktuelle Ziffer eine 0, mache daraus eine 1. Damit ist die gesamte Zahl um 1 hochgezählt.
- Ist die aktuelle Ziffer eine 1, mache daraus eine 0 und gehe zur nächsten Ziffer nach links für den Übertrag.

Das entspricht genau einem Spielzug im Spiel „Drehen“. Ein Spielzug erhöht also den Wert der Binärzahl, die durch die Kartenreihe dargestellt wird, um 1. Die verdeckten Karten zu Beginn stellen die Binärzahl dar, die nur aus Nullen besteht, also den Wert 0 hat.
Das folgende Bild zeigt die Ergebnisse der ersten vier Spielzüge, die also den Zahlen von 1 bis 4 entsprechen. Man kann sehen, dass bei den Zahlen 1, 2 und 4 (also den Zweierpotenzen 2^0, 2^1 und 2^2) genau eine Karte aufgedeckt ist. Das heisst, dass bei einer Binärzahl, die eine Zweierpotenz als Wert hat, nur an der Stelle, die dieser Zweierpotenz entspricht, eine 1 ist.



Nach 16 Spielzügen erhalten wir also die Darstellung einer Binärzahl mit dem Wert 16. Da $16 = 2^4$ ist, hat diese Binärzahl an der fünften Stelle von rechts eine 1 und sonst nur Nullen: 0000000000010000.
In der Darstellung mit den Karten ist also genau diese eine Karte aufgedeckt:



Dies ist Informatik!

Die kleinste Speichereinheit heutiger Computer kann nur zwei Werte unterscheiden: AN oder AUS, WAHR oder FALSCH, 0 oder 1. Alle Daten, die in einem Computer gespeichert und verarbeitet

werden, müssen wir als Reihen binärer Ziffern sehen, letztlich also als Binärzahlen. Deshalb haben Operationen auf Binärzahlen für die Informatik eine grosse Bedeutung.
Seitdem es Computer gibt, wird das Abarbeiten solche Operationen möglichst effizient gebaut. Es gibt Operationen, die zwei Binärzahlen miteinander verknüpfen, wie etwa die Rechenoperationen Addition oder Multiplikation. Es gibt aber auch Operationen, die eine einzelne Binärzahl verändern, etwa das Verschieben aller Ziffern um eine Position nach links oder eben das Hochzählen, also die Addition um 1 wie in dieser Aufgabe.
Gute Prozessoren zeichnen sich dadurch aus, dass sie solche Operationen schnell und energiesparend ausführen, und zwar millionenfach pro Sekunde. Es ist dann die Aufgabe des Programmierers und seiner Werkzeuge, komplizierte Abläufe, auch diese einfachen Operationen, zu reduzieren, so dass der Benutzer beliebige Programme verwenden kann.

Stichwörter und Webseiten

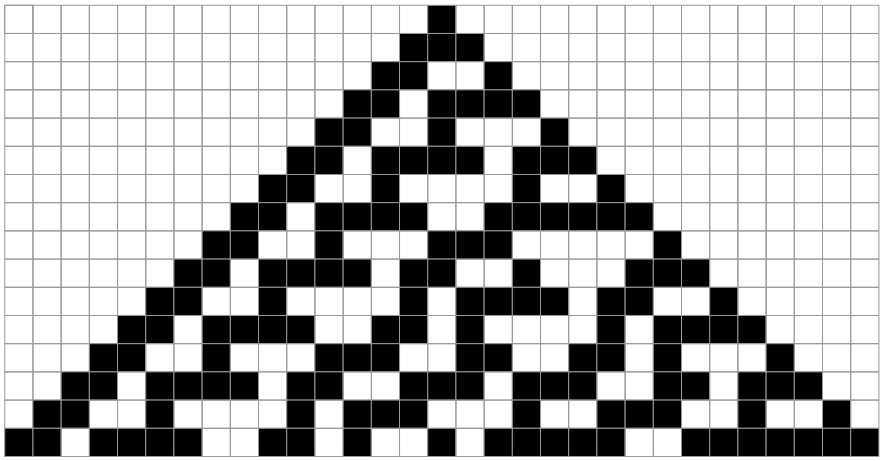
Binärzahlen

- `https://de.wikipedia.org/wiki/Dualsystem`
- `https://de.wikipedia.org/wiki/Synchronzähler`
- `https://de.wikipedia.org/wiki/Asynchronzähler`

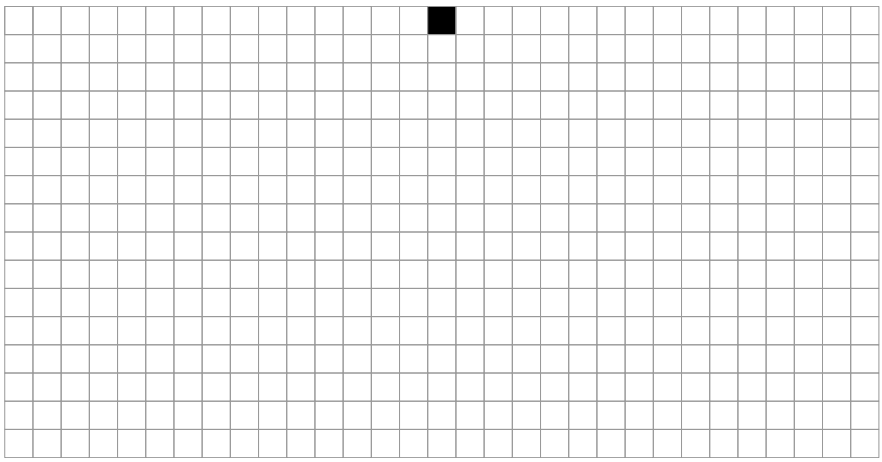
32. Plättlimuster

Tina soll Plättli auf eine Fläche legen, die 31 Plättli breit und 16 Plättli hoch ist. Tina möchte, dass die Plättli nach einem Satz einfacher Regeln gelegt werden.
Eine einfache Regel ist immer so aufgebaut, dass drei Plättli nebeneinander bestimmen, wie das Plättli mittig darunter aussehen soll. Ein Satz einfacher Regeln besteht aus acht einfachen Regeln: für jede mögliche Kombination von Plättli eine einfache Regel (am Rand werden einfach weisse Plättli angenommen):

Tina startet oben in der Mitte mit einem schwarzen Plättli, alle anderen Plättli der ersten Reihe sind weiss. Wenn sie ihre Regeln anwendet, sieht die Fläche so aus:



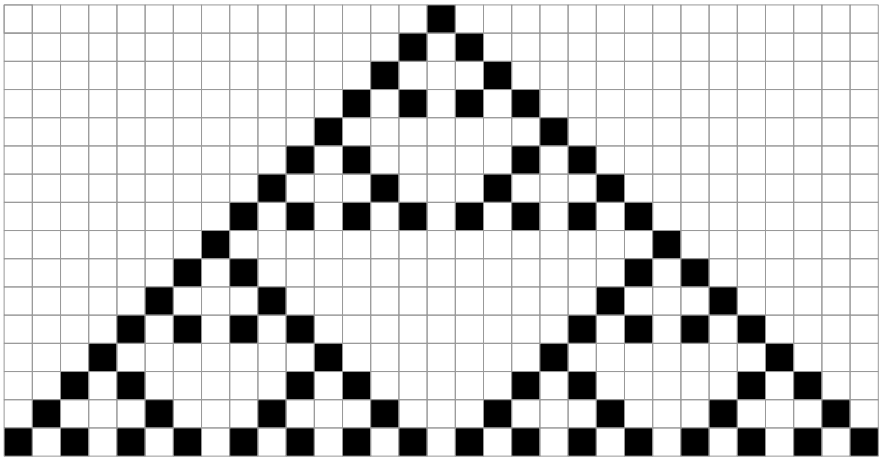
Erstelle einen eigenen Satz einfacher Regeln, so dass in der letzten Reihe immer abwechselnd ein schwarzes und ein weisses Plättli liegt.

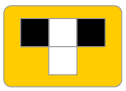


Lösung

Für diese Aufgabe gibt es viele Lösungen. Eine mögliche Lösung ist die folgende:

Sie erzeugt das folgende Muster:



Wenn man sich das Muster genau anschaut, sieht man, dass es eigentlich neun „Dreiecke“ sind: die dritte erzeugte Linie besteht aus abwechselnd schwarzen und weissen Plättli, so dass immer die Regel  angewendet wird ausser am linken und rechten Rand der Gesamtfigur, von wo aus sich neue Dreiecke bilden können.
Wenn man das Ergebnis einer Regel als W für ein weisses Plättli und S für ein schwarzes Plättli abkürzt, kann man jede der 256 möglichen Sätze von Regeln (W oder S für jedes der 8 einzelnen Regeln, also $2^8 = 256$) mit einem Code beschreiben. Das Beispiel in der Aufgabe hätte dann den Code WWWSSSSW.
Die folgenden siebzehn Codes erzeugen in der 16. Zeile ein schwarz-weiss-Muster der Plättli:

SWSSWWSS
SSSSSWSW
WSSSSWSW
SWSSSWSW
WWSSSWSW
SSWSSWSW
WSWSSWSW
SWWSSWSW
WWWSSWSW
SSSSWWSW
WSSSWWSW
SWSSWWSW
WWSSWWSW
SSWSWWSW
WSWSWWSW
SWWSWWSW
WWWSWWSW (die Beispiellösung)

Dies ist Informatik!

Diese Regeln in dieser Aufgabe ähneln Conways Spiel des Lebens (engl. *Conway's Game of Life*), das der englische Mathematiker John Horton Conway im Jahre 1970 veröffentlicht hat. Es basiert auf einem zweidimensionalen zellulären Automaten. Ein zweidimensionaler zellulärer Automat sieht aus wie ein Boden aus Plättli. Jedes Plättli ist eine „Zelle", deren Zustand von den acht Nachbar„zellen" abhängt. Der neue Zustand jeder Zelle wird nach einfachen Regeln aus den (alten) Zuständen der Nachbarzellen berechnet. So kann die z.B. Vermehrung und das Absterben von Lebewesen in einem Gebiet simuliert werden. In diesem Fall ist es ein reduzierter Regelsatz, da eine „Zelle" nur von den drei „Zellen" darüber abhängt.

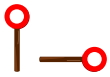
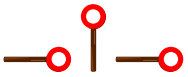
Stichwörter und Webseiten

Muster, Zellulärer Automat

- `http://web.stanford.edu/~cdebs/GameOfLife/`
- `https://en.wikipedia.org/wiki/Rule_90`
- `https://de.wikipedia.org/wiki/Zellulärer_Automat`
- `https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens`

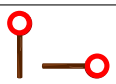
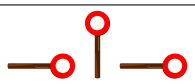
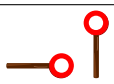
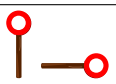
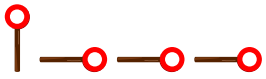
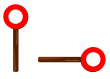
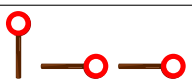
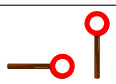
33. Wo ist das Segelflugzeug?

Jana und Robin spielen mit ihrem Segelflugzeug. Einer von ihnen lässt es von einem kleinen Hügel fliegen und der andere holt es nach jeder Landung ab. Leider wurde das Gras der Wiese schon länger nicht mehr gemäht. So sieht man das gelandete Segelflugzeug nur noch vom Hügel und nicht mehr von der Wiese aus. Jana und Robin müssen sich daher gut verständigen können. Dazu haben Sie einen Signalcode vereinbart.

links	rechts	in Richtung Hügel	in Richtung Tal
			

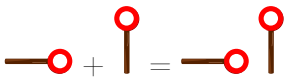
Leider gibt es ein Problem mit diesem Signalcode. Wenn man zum Beispiel die folgenden Befehle sendet, ...

... kann dies „links – in Richtung Hügel – links“ bedeuten, aber auch „links – rechts – links – links“. *Jana und Robin haben sich vier neue Signalcodes überlegt. Welchen Signalcode können Sie widerspruchsfrei nutzen?*

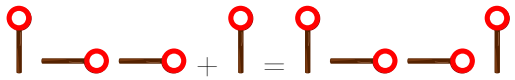
	links	rechts	in Richtung Hügel	in Richtung Tal
A)				
B)				
C)				
D)				

Lösung

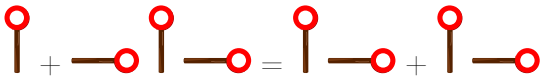
Die richtige Antwort ist C). Man kann das beispielsweise daran erkennen, dass jeder Befehl mit  anfängt und danach 0, 1, 2 oder 3 mal  folgt und  nicht wieder vorkommt. Somit weiss man, dass bei jedem  ein neuer Befehl startet und man muss nur zählen, wie häufig  folgt.
Die Antwort B) ist kein guter Signalcode, da „links“ gefolgt von „rechts“ die gleichen Signale verwendet wie „in Richtung Hügel“:



Die Antwort D) ist kein guter Signalcode, da „links“ gefolgt von „in Richtung Tal“ dasselbe bedeutet wir „in Richtung Hügel“:

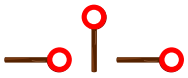


Bei der Antwort A) ist es etwas schwieriger. Wenn man „in Richtung Tal“ und dann „links“ signalisiert, bedeutet das dasselbe wie zweimal „rechts“:



Dies ist Informatik!

Wenn Computer Daten über ein Kabel oder über Funk senden, so tun sie dies durch schnelle Signalfolgen, wobei für jedes Einzelsignal zwei Möglichkeiten bestehen. Man kann sich das als Strom ein oder aus vorstellen (oft sind die Dinge natürlich auch etwas komplexer). Genau dies machen Jana und Robin. Auch sie verwenden zwei Einzelsignale für Ihre Nachrichten.
Diese Art und Weise, in der Nachrichten oder Befehle in Signale umgesetzt werden, wird als (binärer) Code bezeichnet. In diesem Fall ein Code mit variabler Länge, weil die Anzahl der für eine Nachricht oder einen Befehl verwendeten Signale nicht immer gleich sein muss.
Es ist wichtig, dass der Empfänger einer codierten Nachricht die Signale zurück in die ursprüngliche Nachricht übersetzen kann, ohne dass er einen Fehler macht. Mit andere Worten, du musst vorsichtig sein, wenn du einen solchen Code entwirfst. Codes, die wir als „gut“ bezeichnen, werden als eindeutig decodierbare Codes bezeichnet.
Eine spezielle Art von eindeutig decodierbaren Codes ist der Präfix-Code. Dies ist ein Code, bei dem keines der gültigen Codewörter mit der vollständigen Folge von Signalen eines anderen Codeworts beginnt, zum Beispiel:

links	rechts	in Richtung Hügel	in Richtung Tal
			

Präfix-Codes haben schöne Eigenschaften: Sie können ziemlich kurz gehalten werden und sie sind leicht zu entschlüsseln. Sie werden nicht nur für Kommunikationszwecke verwendet, sondern kommen auch in mehreren Komprimierungsalgorithmen.

Stichwörter und Webseiten

Code, Präfix-Code, Signalscheibe

- `https://de.wikipedia.org/wiki/Präfixcode`
- `https://de.wikipedia.org/wiki/Optische_Telegrafie`

34. Probenplan

Fünf Tänzer proben für einen Auftritt: Alex, Bojan, Coco, Deniz und Emil.
Beim Auftritt bilden die Tänzer nacheinander diese Paare:

- Alex – Bojan
- Coco – Alex
- Emil – Deniz
- Alex – Emil
- Coco – Deniz
- Bojan – Coco
- Deniz – Alex
- Coco – Emil

Morgen sollen die Paare nacheinander proben. Dabei soll der Zeitplan so erstellt werden, dass immer ein Mitglied eines Paares zum nächsten Paar gehört und direkt weitermachen kann. Da es sonst zu ermüdend wäre, soll jedoch kein Tänzer dreimal nacheinander proben. Zum Beispiel probt nach dem Paar Alex – Bojan ein anderes Paar, zu dem entweder Alex oder Bojan gehört, also Coco – Alex, Alex – Emil, Bojan – Coco oder Deniz – Alex.
Einer der Tänzer stellt fest, dass er auf jeden Fall später zur Probe kommen kann: Er wird auf keinen Fall zum ersten Paar auf dem Zeitplan gehören.
Welcher Tänzer ist das?

A) Alex

B) Bojan

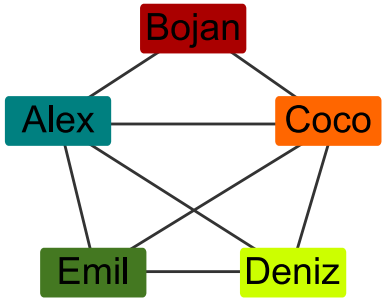
C) Coco

D) Deniz

E) Emil

Lösung

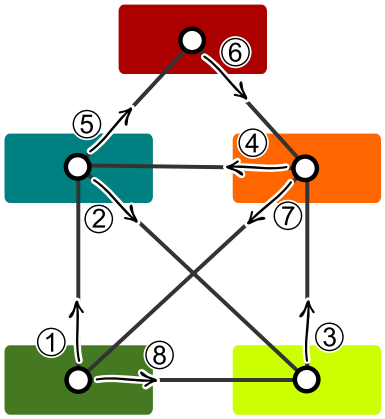
B) Bojan ist die richtige Antwort.
Im folgenden Diagramm ist für jeden Tänzer ein Viereck mit seinem Namen zu sehen. Ausserdem sind zwei Vierecke mit einer Linie verbunden, wenn die beiden Tänzer ein Paar bilden. Ein gültiger Probenplan, ist dann ein „Weg“ durch das Diagramm: Dieser beginnt bei einem Viereck und geht genau einmal an jeder Linie entlang. Ein direktes Zurück gibt es nicht, denn sonst würde ein Tänzer ja dreimal nacheinander proben.



Dabei muss der Weg jedes Viereck, zu dem er zwischendurch kommt, auch wieder verlassen. Bei solchen Vierecken gehen eine gerade Anzahl an Linien aus (die Tänzer sind also in einer geraden Anzahl von Paaren beteiligt). Die Vierecke, bei denen der Weg beginnt bzw. endet, haben eine ungerade Anzahl Linien aus (die Tänzer sind also an einer ungeraden Anzahl von Paaren beteiligt). Von den Vierecken von Deniz und Emil gehen eine ungerade Anzahl Linien aus (nämlich jeweils drei Linien). Nur diese beiden können also zum ersten oder letzten Paar gehören. Bojan ist der einzige Tänzer, der weder mit Deniz noch mit Emil ein Paar bildet. Damit kann nur er auf keinen Fall zum ersten Paar des Probenplans gehören.

Dies ist Informatik!

Das Bild oben zeigt, wie die Tanzpaare als *Graph* dargestellt werden. Ein Graph besteht aus *Knoten* (hier: die Tänzer) und *Kanten* (hier: die Paar-Bildungen der Tänzer). Er ist eine sehr vielseitige Struktur, die bei vielen Informatik-Aufgabenstellungen zur Modellierung verwendet wird, z. B. bei Verkehrs- oder Kommunikationsnetzen. In dieser Aufgabe bilden die Tänzer ein Paar-Netzwerk.
In vielen Netzwerken kann es nötig sein, auf einem Weg von einem Start- zu einem (unterschiedlichen) Zielknoten alle Verbindungen abzugehen. Dabei stellt sich die Frage, z. B. aus Gründen der Effizienz, ob es möglich ist, einen solchen Weg so zu gestalten, dass jede Verbindung nur einmal begangen wird. Ein solcher Weg, der jede Kante genau einmal durchläuft, wird zu Ehren von Leonhard Euler, dem Erfinder der Graphentheorie, als Eulerweg bezeichnet. Euler hatte bewiesen, dass in einem zusammenhängenden Graphen ein Eulerweg existiert, wenn genau zwei Knoten eine ungerade Anzahl Verbindungen mit anderen Knoten haben (und alle anderen Knoten demnach eine gerade Anzahl von Verbindungen haben). Nur diese beiden Knoten können Anfangs- oder Endpunkt des Eulerwegs sein.



Das Paar-Netzwerk dieser Biberaufgabe hast Du übrigens wahrscheinlich schon einmal gesehen. Wenn man es ein wenig dreht und die Knoten verkleinert, erkennt man das *Haus des Nikolaus*. Wer das Haus vom Nikolaus in einem Zug zeichnet, geht also einen Eulerweg entlang der Linien des Hauses.

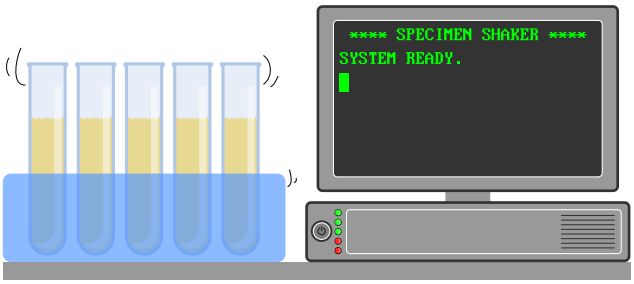
Stichwörter und Webseiten

Graph, Eulerweg

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

35. Labor

In einem medizinischen Labor muss eine Probe eines Patienten regelmässig geschüttelt werden. Dafür benutzt das medizinische Labor eine Maschine, die ein Programm ausführt. Das Programm wird Zeile für Zeile ausgeführt.
Die Maschine wird mit dem folgenden Programm programmiert:




```
1 SPEICHERE 0 ALS N
2 ERHÖHE N UM 1
3 GEHE ZU ZEILE 6
4 WENN N GLEICH 60 IST, DANN GEHE ZU ZEILE 8
5 SPEICHERE 0 ALS N
6 ERHÖHE N UM 1
7 GEHE ZU ZEILE 2
8 WIEDERHOLE N MAL SCHÜTTLE
9 ENDE
```

Die Befehle sind:

- `SPEICHERE` *`Zahl`* `ALS` *`Name`*: Speichert die Zahl *`Zahl`* unter dem Namen *`Name`*.
- `ERHÖHE` *`Name`* `UM 1`: Liest die unter *`Name`* gespeicherte Zahl, addiert 1 und speichert die erhöhte Zahl unter *`Name`*.
- `GEHE ZU ZEILE` *`Zeilennummer`*: Führt das Programm ab der Zeile *`Zeilennummer`* weiter.
- `WENN` *`Name`* `GLEICH` *`Zahl`* `IST, DANN` *`Befehl`*: Vergleicht die Zahl, die unter *`Name`* gespeichert ist, mit der Zahl *`Zahl`*. Wenn beide gleich sind, führt es den Befehl *`Befehl`* aus, sonst nicht.
- `WIEDERHOLE` *`Name`* `MAL` *`Befehl`*: Führt den Befehl *`Befehl`* so häufig aus wie die Zahl, die unter *`Name`* gespeichert ist.
- `SCHÜTTLE`: Schüttelt die Probe einmal.
- `ENDE`: Beendet das Programm.

Wie häufig wird die Maschine die Probe schütteln?

A) Die Probe wird nie geschüttelt.

B) Die Probe wird einmal geschüttelt.

C) Die Probe wird 60 mal geschüttelt.

D) Die Maschine wird nicht aufhören, die Probe zu schütteln.

Lösung

Das Programm springt immer wieder von der Zeile 3 zur Zeile 6 und von der Zeile 7 zur Zeile 2. Ausser ganz am Anfang die Zeile 1 werden also nur die Zeilen 2, 3, 6 und 7 ausgeführt. Die Probe würde in der Zeile 8 geschüttelt werden, die aber nie ausgeführt wird. Das heisst, dass das Programm die Probe letztlich nie schütteln wird. Da auch die Zeile 9 nie ausgeführt wird, beendet die Maschine das Programm auch nie.

Dies ist Informatik!

Das Programm benutzt als Kontrollstruktur den Befehl „`GEHE ZU ZEILE`“ (engl. “`GO TO`”), um zu anderen Programmteilen zu springen. Diese Kontrollstruktur ist sehr hardwarenah und wurde in frühen Programmiersprachen (bis in die 1980er-Jahre) häufig geschrieben, um in Abhängigkeit von Benutzereingaben oder anderen Bedingungen zu reagieren. Es ist jedoch manchmal nicht einfach, als Mensch einen solchen Programmcode zu lesen und zu verstehen, häufig passierten so Fehler. Moderne Programmiersprachen, wie sie ab den 1950er-Jahren entwickelt wurden und sich langsam durchsetzten, benutzen deshalb diese Kontrollstruktur nicht mehr. Anstelle dessen benutzt man Schleifen (wie das `WIEDERHOLE` im Beispiel), Verzweigungen mit Programmblöcken oder Unterprogramme.

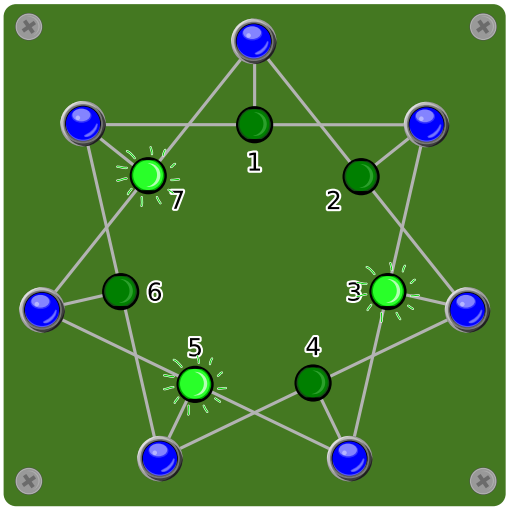
Stichwörter und Webseiten

Kontrollstruktur, Programmanalyse, `GOTO`

- `https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf`
- `https://de.wikipedia.org/wiki/Sprunganweisung`

36. Licht an!

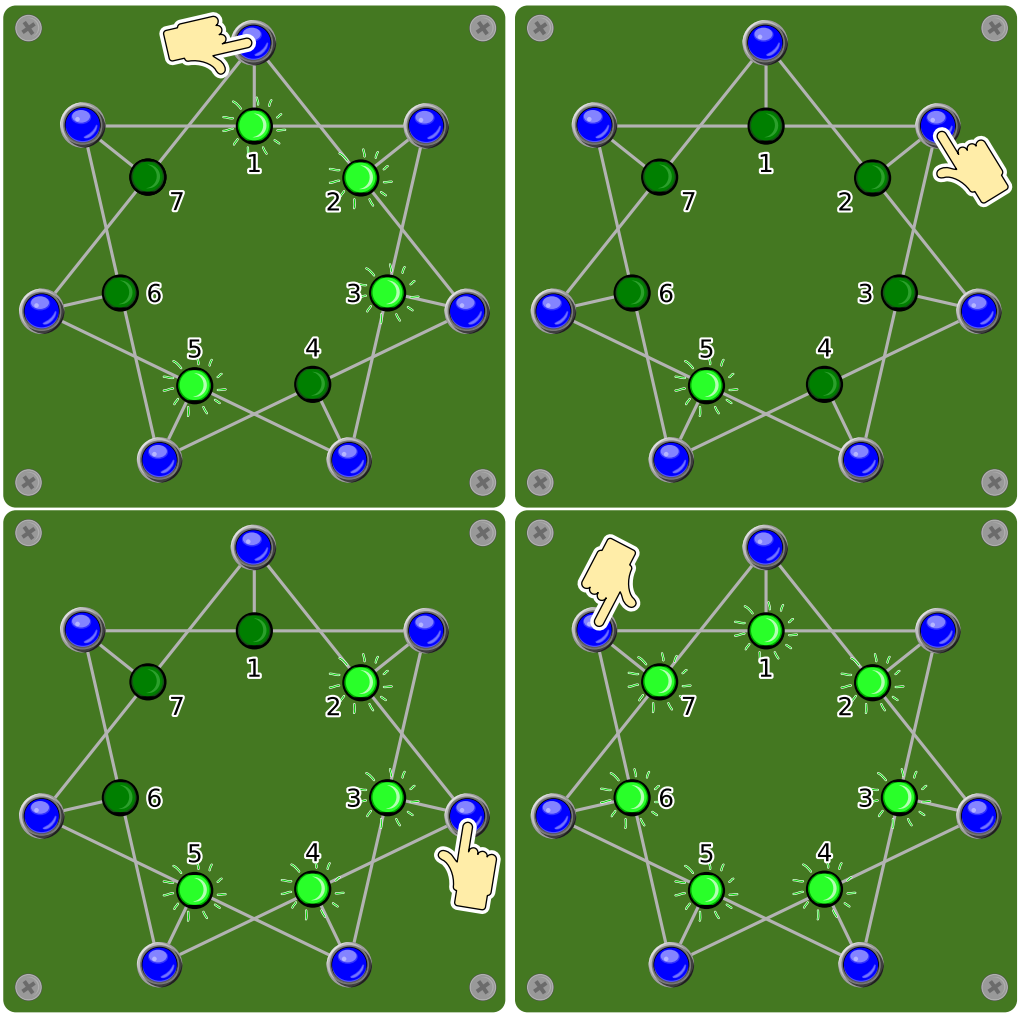
Sieben Schalter sind mit sieben Lampen verbunden und zwar so, dass ein Schalter immer drei Lampen kontrolliert. Wird ein Schalter gedrückt, wird eine von ihm kontrollierte Lampe, die eingeschaltet war, ausgeschaltet und eine von ihm kontrollierte Lampe, die ausgeschaltet war, eingeschaltet.



Welche Schalter muss man drücken, dass am Ende alle Lampen leuchten?

Lösung

Wenn man den Schalter bei der Lampe 1 (oder bei der Lampe 2) drückt, werden die beiden Lampen 1 und 2 eingeschaltet und die danebenliegende Lampe 7 (oder 3) ausgeschaltet. Damit sind genau drei nebeneinanderliegende Lampen eingeschaltet. Indem man auf den Schalter bei der mittleren dieser Lampen drückt (Schalter 2 oder 1, je nachdem was man vorher gedrückt hat), werden diese drei ausgeschaltet. Damit sind alle Lampen ausser der Lampe 5 ausgeschaltet. Da dies sechs Lampen sind, können diese mit zwei Schaltern jeweils in der Mitte eines der beiden Dreierblöcke (Schalter 3 und 7) einschalten. Damit sind alle Lampen eingeschaltet.



Jede mögliche Reihenfolge dieser Schalter (1, 2, 3, 7) führt zum richtigen Resultat. Das zweimalige Drücken desselben Schalters macht das Resultat des ersten Drückens rückgängig. Damit stellt sich letztlich nur die Frage, welcher Schalter gedrückt werden muss und welcher nicht. Bei sieben Schaltern gibt es immerhin noch $2^7 - 1 = 127$ verschiedene Möglichkeiten, die Schalter zu drücken oder nicht (keinen Schalter zu drücken ist offensichtlich falsch).

Dies ist Informatik!

Diese Aufgabe besteht darin, ein System in einem bestimmten bekannten Zustand (Lampen 3, 5 und 7 sind eingeschalten) in einen anderen bekannten Zustand (Lampen 1, 2, 3, 4, 5, 6 und 7 sind

eingeschaltet) zu überführen. Dabei müssen bestimmte Regeln beachtet werden. Das Finden des Weges steht im Vordergrund.
In der Informatik stellen sich solche Fragen öfters. Naiv könnte man natürlich anfangen, einfach alle Kombinationen durchzuprobieren (was zu den oben erwähnten 127 verschiedenen Möglichkeiten führt). Will man schneller auf das Ergebnis kommen, lohnt es sich, das Problem von zwei Seiten her anzugehen: zum einen aus Richtung des Startzustandes und parallel dazu aus Richtung des Endzustandes. Je grösser das System ist, desto mehr Zeit kann man mit dieser Methode der bidirektionalen Suche sparen.

Stichwörter und Webseiten

Bidirektionale Suche, Endlicher Automat

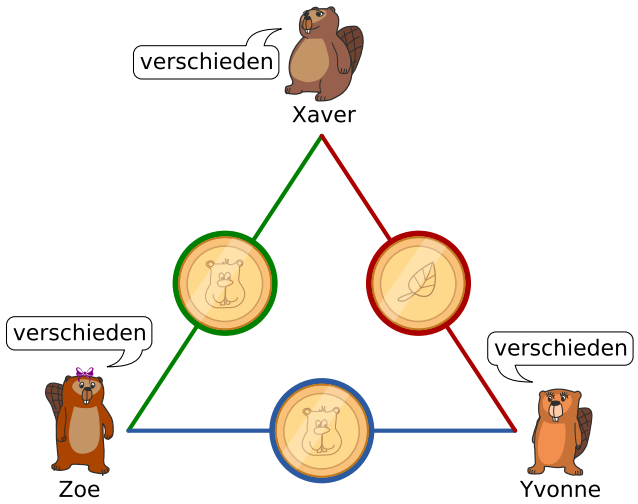
- `https://de.wikipedia.org/wiki/Bidirektionale_Suche`

37. Streng geheim

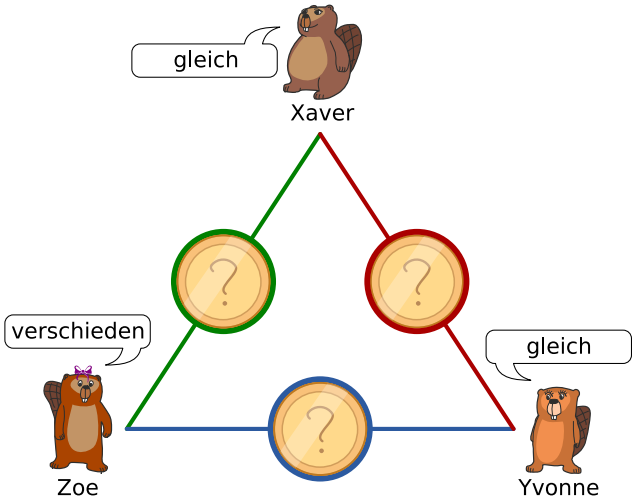
Xaver, Yvonne und Zoe spielen gelegentlich bei einer Tombola mit. Jetzt wurde bekannt, dass der einzige Haupttreffer von jemandem aus ihrer Stadt gewonnen wurde. Gerne würden sie wissen, ob einer von ihnen zufällig den Hauptgewinn gewonnen hat, aber andererseits sollte geheim bleiben, wer es ist. So gehen sie vor:

1. Xaver und Yvonne werfen eine Münze, nur sie kennen das Ergebnis.
2. Xaver und Zoe werfen eine Münze, nur sie kennen das Ergebnis.
3. Yvonne und Zoe werfen eine Münze, nur sie kennen das Ergebnis.
4. Jeder von ihnen gibt danach bekannt, ob ihre beiden jeweiligen Münzwürfe „gleich“ oder „verschieden“ ausgegangen sind.
   - Jemand, der nicht den Hauptgewinn gewonnen hat, soll wahrheitsgemäss antworten.
   - Jemand, der den Hauptgewinn gewonnen hat, soll den Wahrheitswert seiner Aussage umdrehen (also „gleich“ sagen, auch wenn seine beiden Ergebnisse verschieden waren und umgekehrt).

Unten ein Beispiel mit erfolgten Münzwürfen und mit der Annahme, dass Zoe den Hauptpreis gewonnen hat.



Betrachte folgende Situation, bei der die Münzwürfe erfolgen, dir aber nicht bekannt sind.



Welche der folgenden Aussagen ist wahr?

A) Keiner von den drei Freunden hat den Hauptgewinn gewonnen.

B) Einer von den drei Freunden hat den Hauptgewinn gewonnen, aber wir wissen nicht wer.

C) Einer von den drei Freunden hat den Hauptgewinn gewonnen, und wir wissen genau wer.

D) Wir wissen nicht, ob jemand den Hauptgewinn gewonnen hat.

Lösung

Die richtige Antwort ist B). Einer von den drei Freunden hat den Hauptgewinn gewonnen, aber wir wissen nicht wer.
Für diese Frage gibt es verschiedene Lösungsansätze. Einer ist: Bei drei Münzwürfen gibt es genau diese zwei Möglichkeiten:

- Alle drei Münzen zeigen das gleiche Ergebnis.
- Eine Münze zeigt ein anderes Ergebnis als die beiden anderen.

Unter der Annahme, dass niemand den Hauptgewinn gewonnen hat, gilt folgendes:

- Sind alle Münzen gleich, dann sagen alle drei Freunde „gleich“.
- Zeigt eine Münze ein anderes Ergebnis, dann sagt einer der Freunde „gleich“ und die beiden anderen sagen „verschieden“.

Da aber zwei Freunde „gleich“ sagen und einer „verschieden“, muss einer von ihnen lügen. Also hat einer den Hauptpreis gewonnen.
Wir können aber nicht sagen, wer. Denn wenn einer der beiden Freunde, die „gleich“ sagen, eigentlich „verschieden“ hätten sagen müssen, wenn sie die Wahrheit sagen würden, wissen wir nicht, welcher der beiden lügt. Und auch der dritte könnte lügen: dann würden alle eigentlich „gleich“ sagen müssen, was auch sein könnte.

Dies ist Informatik!

Die von Xaver, Yvonne und Zoe gewählte Vorgangsweise wurde als sogenanntes *Dining Cryptographers Problem* zuerst beschrieben.
Diese Methode ist für Informatiker deshalb besonders interessant, weil sowohl der Sender als auch der Empfänger der Nachricht anonym bleiben: Wenn das Verfahren richtig ausgeführt wird, wissen am Ende alle verlässlich, ob einer von ihnen der Gewinner ist. Ausser dem Gewinner selbst weiss aber keiner, von wem die Information kommt. Auch der Empfänger der Information bleibt anonym, weil alle Beteiligten die Information erhalten haben.

Stichwörter und Webseiten

Anonymität, Dining Cryptographers Problem

- `https://en.wikipedia.org/wiki/Dining_cryptographers_problem`

A. Aufgabenautoren

Andrea Adamoli
Jared Asuncion
Wilfried Baumann
Carlo Bellettini
Javier Bilbao
Daphne Blokhuis
Laura Briviba
Lucia Budinská
Špela Cerar
William Chan
Kessarapan Charoensueksa
Anton Chukhnov
Kris Coolsaet
Valentina Dagienė
Darija Dasović Rakijašić
Christian Datzko
Susanne Datzko
Dilek Doğan
Marissa Engels
Hanspeter Erni
Veerle Fack
Georgios Fessakis
Gerald Futschek
Ionuț Gorgos
Shuchi Grover
Yasemin Gülbahar
Martin Guggisberg
Bent Halden
Urs Hauser
Wei-fu Hou
Juraj Hromkovič
Takeharu Ishizuka
Svetlana Jakšić
Zhang Jinbao
Emil Kelevedjiev
Dong Yoon Kim
Vaidotas Kinčius
Iryna Kirynovich
Jia-Ling Koh
Regula Lacher
Anh Vinh Le
Dan Lessner
Judith Lin
Violetta Lonati
Nils Mak
Dimitris Mavrovouniotis
Karolína Mayerová
Mattia Monga
Samart Moodleah
Anna Morpurgo
Tom Naughton
Henry Ong
Sanja Pavlovic Šijanović
Péter Piltmann
Zsuzsa Pluhár
Wolfgang Pohl
Ilya Posov
Stavroula Prantsoudi
Nol Premasathian
J.P. Pretti
Doris Reck
Alei Reyes
Chris Roffey
Kirsten Schlüter
Andrea Maria Schmid
Victor Schmidt
Andrea Schrijvers
Eljakim Schrijvers
Vipul Shah
Mohamed El-Sherif
Jacqueline Staub
Allira Storey
Gabrielė Stupurienė
Faisal Al-Sudani
Márta Szabó
Aliaksei Tolstsikau
Peter Tomcsányi
Ahto Truu
Willem van der Vegt
Jiří Vaníček
Troy Vasiga
Rechilda Villame
Eslam Wageed
Pieter Waker
Michael Weigend
Khairul A. Mohamad Zaki
Magdalena Zarach

B. Sponsoring: Wettbewerb 2018

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.roborobo.ch/
Die RoboRobo Produkte fördern logisches Denken, Vorstellungsvermögen, Fähigkeiten Abläufe und Kombinationen auszudenken und diese systematisch aufzuzeichnen.
Diese Produkte gehören in innovative Schulen und fortschrittliche Familien. Kinder und Jugendliche können in einer Lektion geniale Roboter bauen und programmieren. Die Erwachsenen werden durch die Erfolgserlebnisse der „Erbauer“ miteinbezogen.
RoboRobo ist genial und ermöglicht ein gemeinsames Lern-Erlebnis!

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.

UBS

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

bbv Software Services

`http://www.bbv.ch/`
bbv Software Services AG ist ein Schweizer Software- und Beratungsunternehmen. Wir stehen für Top-Qualität im Software Engineering und für viel Erfahrung in der Umsetzung. Wir haben uns zum Ziel gesetzt, unsere Expertise in die bedeutendsten Visionen, Projekte und Herausforderungen unserer Kunden einzubringen. Wir sind dabei als Experte oder ganzes Entwicklungsteam im Einsatz und entwickeln individuelle Softwarelösungen.
Im Bereich der Informatik-Nachwuchsförderung engagiert sich die bbv Software Services AG sowohl über Sponsoring als auch über die Ausbildung von Lehrlingen. Wir bieten Schnupperlehrtage an und bilden Informatiklehrlinge in der Richtung Applikationsentwicklung aus. Mehr dazu erfahren Sie auf unserer Website in der Rubrik Nachwuchsförderung.

PRESENTEX
Das Geschenk – die gute Werbung

`http://www.presentex.ch/`
Beratung ist keine Nebensache
Wir interessieren uns, warum, wann und wie die Werbeartikel eingesetzt werden sollen – vor allem aber, wer angesprochen werden soll.

Bits with Bite
ZUBLER & PARTNER AG
Informatik

`http://www.zubler.ch/`
Zubler & Partner AG Informatik
Umfassendes Angebot an Dienstleistungen.

OXOCARD

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

http://www.diartis.ch/
Diartis AG
Diartis entwickelt und vertreibt Softwarelösungen für das Fallmanagement.

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

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

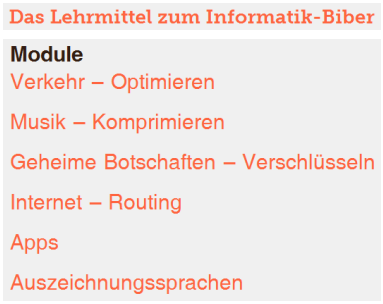
http://www.hepl.ch/
Haute école pédagogique du canton de Vaud

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

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

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.

Der Informatik-Biber auf Facebook:
https://www.facebook.com/informatikbiberch

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.