INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Aufgaben und Lösungen 2022

Schuljahre 5/6

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

Herausgeber:

Susanne Datzko, Nora A. Escherle,
Jean-Philippe Pellet

SVIA

www.svia-ssie-ssii.ch
schweizerischer verein für informatik in der ausbildung // société suisse pour l'informatique dans l'enseignement // società svizzera per l'informatica nell'insegnamento

HASLERSTIFTUNG ETH zürich | ABZ AUSBILDUNGS- UND BERATUNGSZENTRUM FÜR INFORMATIKUNTERRICHT hep/ Scuola universitaria professionale della Svizzera italiana SUPSI

Mitarbeit Informatik-Biber 2022

Masiar Babazadeh, Susanne Datzko, Jean-Philippe Pellet, Giovanni Serafini, Bernadette Spieler

Projektleitung: Nora A. Escherle

Herzlichen Dank für die Aufgabenentwicklung für den Schweizer-Wettbewerb an:
Juraj Hromkovič, Christian Datzko, Jens Gallenbacher, Regula Lacher: ETH Zürich, Ausbildungges- und Beratungszentrum für Informatikunterricht
Tobias Berner: Pädagogische Hoschschule Zürich
Waël Almoman: Collège Voltaire

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

Die Online-Version des Wettbewerbs wurde auf cuttle.org realisiert. Für die gute Zusammenarbeit danken wir:
Eljakim Schrijvers, Justina Dauksaite, Dave Oostendorp, Alieke Stijf, Kyra Willekes, Jo-Ann Bolten: cuttle.org, Niederlande
Chris Roffey: UK Bebras Administrator, Vereinigtes Königreich

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

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

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

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

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Der Informatik-Biber 2022 wurde vom Schweizerischen Verein für Informatik in der Ausbildung (SVIA) durchgeführt und massgeblich von der Hasler Stiftung unterstützt. Wettbewerbssponsoren sind das Amt für Wirtschaft und Arbeit des Kantons Zürich sowie die UBS.

Dieses Aufgabenheft wurde am 22. November 2023 mit dem Textsatzsystem LaTeX erstellt. Wir bedanken uns bei Christian Datzko für die Entwicklung und langjährige Pflege des Systems zum Generieren der 36 Versionen dieser Broschüre (nach Sprachen und Schulstufen). Das System wurde analog zum Vorgänger-System neu programmiert, welches ab 2014 gemeinsam mit Ivo Blöchliger entwickelt wurde. Jean-Philippe Pellet danken wir für die Entwicklung der `bebras` Toolchain, die seit 2020 für die automatisierte Konvertierung der Markdown- und YAML-Quelldokumente verwendet wird.

Hinweis: Alle Links wurden am 1. Dezember 2022 geprüft.

Vorwort

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

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

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

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

Der Informatik-Biber 2022 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

Jede Altersgruppe erhält Aufgaben in drei Schwierigkeitsstufen: leicht, mittel und schwierig. In den Altersgruppen 3 und 4 waren 9 Aufgaben zu lösen, mit je drei Aufgaben in jeder der drei Schwierigkeitsstufen. Für die Altersklassen 5 und 6 waren es je vier Aufgaben aus jeder Schwierigkeitsstufe, also 12 insgesamt. Für die restlichen Altersklassen waren es 15 Aufgaben, also fünf Aufgaben pro Schwierigkeitsstufe.

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

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

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

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

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

Bei vielen Aufgaben wurden die Antwortalternativen am Bildschirm in zufälliger Reihenfolge angezeigt. Manche Aufgaben wurden in mehreren Altersgruppen gestellt. Diese Aufgaben hatten folglich in den verschiedenen Altersgruppen unterschiedliche Schwierigkeitsstufen.

Einige Aufgaben werden für bestimmte Altersgruppen als «Bonus» angegeben: sie haben keinen Einfluss auf die Berechnung der Gesamtpunktzahl. Diese Übungen dienen vielmehr dazu, bei mehreren TeilnehmerInnen mit identischer Punktzahl zu entscheiden, wer sich für eine mögliche nächste Runde qualifiziert.

Für weitere Informationen:

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

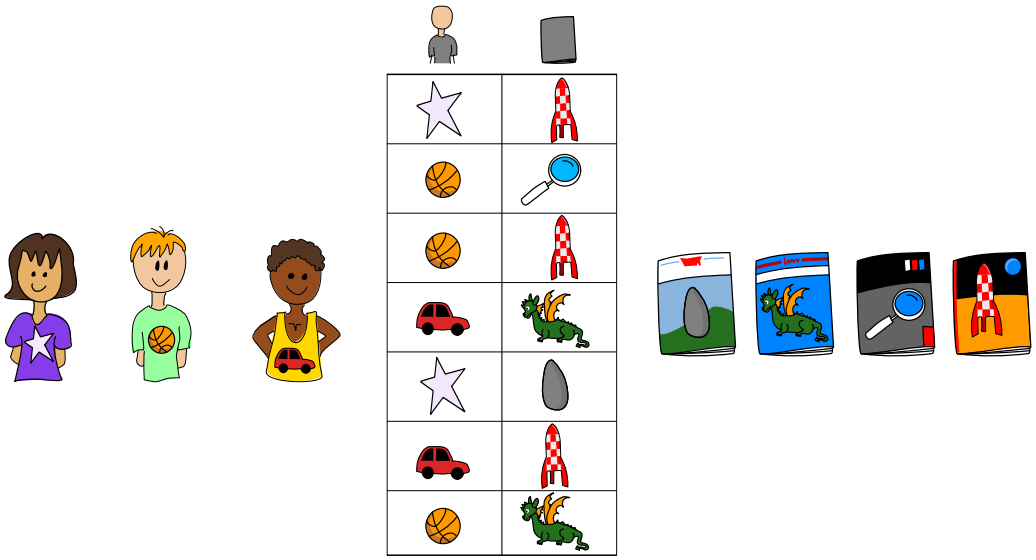
`https://www.informatik-biber.ch/de/kontaktieren/`
`https://www.informatik-biber.ch/`

Inhaltsverzeichnis

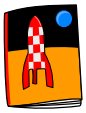
1. Kinder lieben Bücher

Die Kinder leihen in der Bibliothek Bücher aus. Die Bibliothek schreibt in einer Tabelle auf, wer welches Buch ausgeliehen hat.

Welches Buch haben die Kinder am häufigsten ausgeliehen?

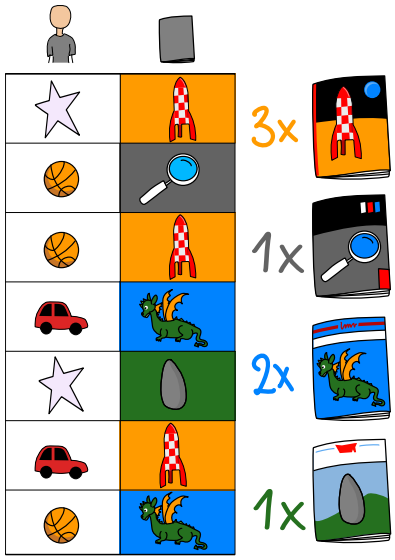


Lösung

Die richtige Antwort ist: 

Folgendes stimmt:

- Drei Kinder haben das Buch mit der Rakete ausgeliehen.
- Ein Kind hat das Buch mit der Lupe ausgeliehen.
- Zwei Kinder haben das Buch mit dem Drachen ausgeliehen.
- Ein Kind hat das Buch mit dem Hinkelstein ausgeliehen.

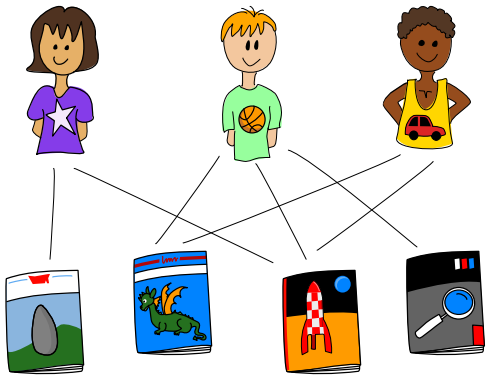


Es ist also das Buch mit der Rakete, das am häufigsten ausgeliehen wurde.

Dies ist Informatik!

Es ist schön, dass die Kinder im Informatik-Biber-Wettbewerb gerne Bücher lesen!

Aber brauchen wir wirklich eine Tabelle mit Kindern und Büchern, um die Wünsche der Kinder darzustellen? Würde es nicht auch gehen, wenn man einfach Linien zeichnen würde?



Das wäre für Menschen einfacher, aber nicht für Computer. Computer können Linien nicht gut lesen. Dafür sind sie sehr gut bei der Arbeit mit Tabellen. Wenn wir wollen, dass Computer uns bei der Arbeit helfen, wie zum Beispiel welches Kind welches Buch ausgeliehen hat, oder welcher Person welches Bankkonto gehört, dann ist es meist eine gute Idee, dies in Tabellen darzustellen.

Tabellen hat man schon vor 4000 Jahren in Babylon eingeführt, um Informationen über *Beziehungen* abzuspeichern. Diese Möglichkeit, Beziehungen abzuspeichern, macht Tabellen zu einem wichtigen Grundkonzept der *relationalen Datenbanken*.

Die Tabellen stellen die Beziehungen zwischen Dingen (oder Menschen) dar. Die Beziehungen bestimmen, wie man die Informationen in den Tabellen darstellt. Wenn es beispielsweise eine Regel gäbe, dass jedes Kind nur ein Buch ausleihen dürfte, hätte die Tabelle nur eine Zeile für jedes Kind. In unserem Beispiel mit der Bibliothek ist es in Ordnung, dass Kinder mehrere Bücher ausleihen dürfen, sie dürfen sogar auch die gleichen Bücher wie andere Kinder ausleihen. In diesem Fall braucht es diese spezielle Tabelle, die die Kinder und Bücher verbindet – und die mehrere Male das gleiche Kind und auch mehrere Male das gleiche Buch auflisten kann.

Die Ausleihtabelle ist praktisch. Wenn ein Buch fehlt, kann der Bibliothekar z. B. nachsehen, ob es verliehen wurde. Die Ausleihtabelle hat zwei Spalten und viele Zeilen. In der ersten Spalte wird das Kind eingetragen, das sich ein Buch ausleiht und in der zweiten Spalte das Buch. So kann die Frage nach dem am häufigsten verliehenen Buch ganz einfach durch Nachzählen in der zweiten Spalte beantwortet werden.

Diese Aufgabe könnte auch ein Computer erledigen. Wenn es sich um eine grosse Bibliothek mit vielen tausend Büchern handelt, dann geht es gar nicht anders! In solch einer grossen Bibliothek würde nicht nur die Ausleihtabelle geführt. Es gäbe auch eine Kundenkartei (Kundentabelle) in der alle Informationen über die Kunden wie Name, Adresse und Telefonnummer hinterlegt wären, und ein Bücherverzeichnis (Büchertabelle) mit Angaben zu den Büchern wie Autor und Titel. So bleibt die Ausleihtabelle ganz schlank weil hier nur die Beziehungen (also wer welches Buch ausgeliehen hat) zwischen den Kunden und den Büchern sind.

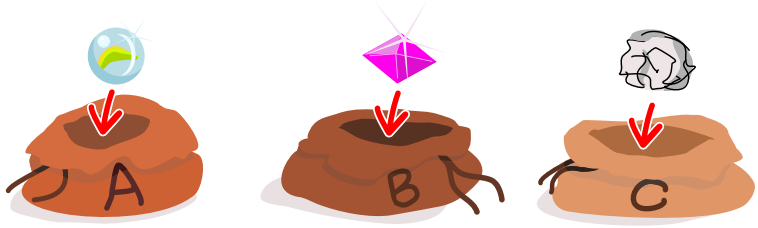
In der Informatik nennt man solche Tabellen relationale Datenbanken.

Stichwörter und Webseiten

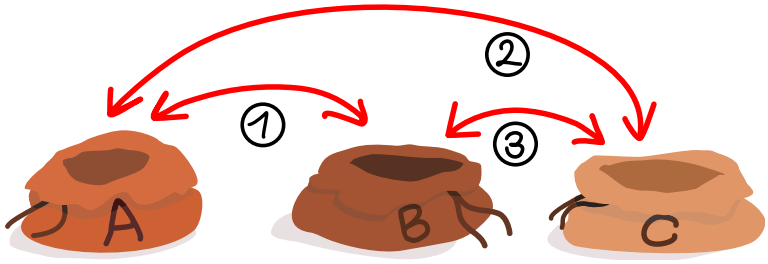
- Beziehungen (Datenbanken): `https://mediencommunity.de/system/files/Beziehungen in Datenbanken.pdf`
- relationale Datenbank: `https://de.wikipedia.org/wiki/Relationale_Datenbank`

2. Vertausche dreimal

Lila legt eine Murmel in Beutel A, einen Edelstein in Beutel B und ein Stück Papier in Beutel C.



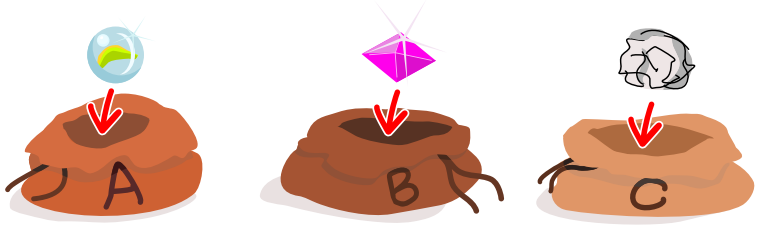
Dann vertauscht sie die Inhalte von Beutel A und Beutel B, danach die Inhalte von A und C und schliesslich vertauscht sie die Inhalte von B und C.



Wo befinden sich dann die 3 Dinge?

Lösung

Am Anfang haben wir diese Anordnung der 3 Dinge in den Beuteln:



Lila vertauscht die Dinge 3 Mal. Nach der ersten Vertauschung (A-B) sehen die Beutel so aus:

Nach der zweiten Vertauschung (A-C) so:

Nach der dritten und letzten Vertauschung (B-C) so:

Daher ist am Ende das Papier in A, der Edelstein in B und die Murmel in C. Dieses Ergebnis hätte man auch einfacher erreichen können, nämlich durch eine einzige Vertauschung der Inhalte von A und C.

Dies ist Informatik!

Hier geht es um Reihenfolgen von Dingen. Eine solche Reihenfolge von Dingen wird auch als Anordnung bezeichnet. Eine andere Reihenfolge stellt eine andere Anordnung dar. Eine Vertauschung ändert die Reihenfolge der Dinge und führt daher zu einer anderen Anordnung. In unserer Aufgabe haben wir am Anfang die Anordnung Murmel-Edelstein-Papier und nach den 3 Vertauschungen die Anordnung Papier-Edelstein-Murmel.

Eine interessante Frage ist, wie viele verschiedenen Anordnungen 3 Dinge haben können. Wir können es uns zunächst einmal etwas einfacher machen und nur alle Anordnungen herstellen, mit einem bestimmten Ding an erster Stelle. Für die beiden restlichen Dinge gibt es dann nur zwei Anordnungen. Wenn die Murmel an erster Stelle ist, dann sind es die beiden Anordnungen:

Murmel-Edelstein-Papier
Murmel-Papier-Edelstein

Es gibt daher auch für die beiden anderen Dinge an erster Stelle jeweils nur zwei verschiedene Anordnungen. Also gibt es noch 4 weitere Anordnungen von den 3 Dingen:

Edelstein-Murmel-Papier
Edelstein-Papier-Murmel
Papier-Murmel-Edelstein
Papier-Edelstein-Murmel

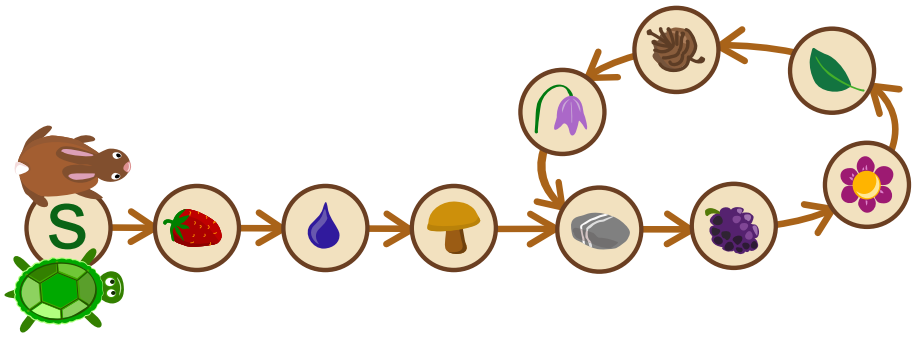
Interessant ist auch die Tatsache, dass man nur mit Vertauschungen jede beliebige Anordnung herstellen kann. Dazu sind bei n Dingen höchstens $n - 1$ Vertauschungen notwendig.

Stichwörter und Webseiten

- Vertauschung:
  - `https://de.wikipedia.org/wiki/Zyklische_Permutation`
  - `https://www.studienkreis.de/mathematik/permutation-definition-aufgaben/`
  - `https://www.lernhelfer.de/schuelerlexikon/mathematik-abitur/artikel/permutationen`

3. Schildkröte und Hase

Eine Schildkröte und ein Hase machen einen Wettlauf. Sie verwenden diese Laufbahn.



Sie starten gleichzeitig auf dem Startfeld. Sie gehen von Feld zu Feld und folgen den Pfeilen.

In einer Minute geht ...

- ... die Schildkröte ein Feld vorwärts.
- ... der Hase zwei Felder vorwärts.

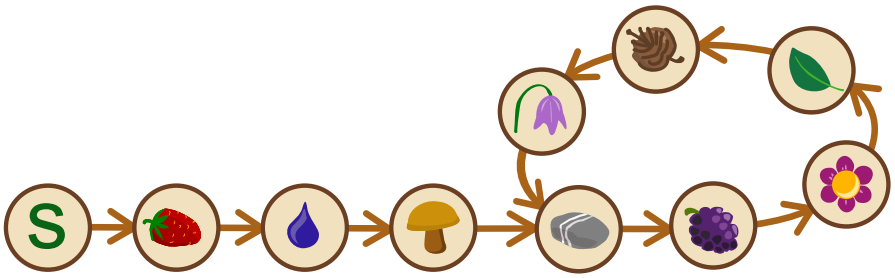
Auf welchem Feld treffen sich Schildkröte und Hase nach dem Start das erste Mal?

Lösung

Schildkröte und Hase treffen einander erstmals auf Feld 8. Man kann das leicht mit 2 Fingern nachvollziehen.

Die folgende Tabelle zeigt im Minutentakt die Felder von Schildkröte und Hase:

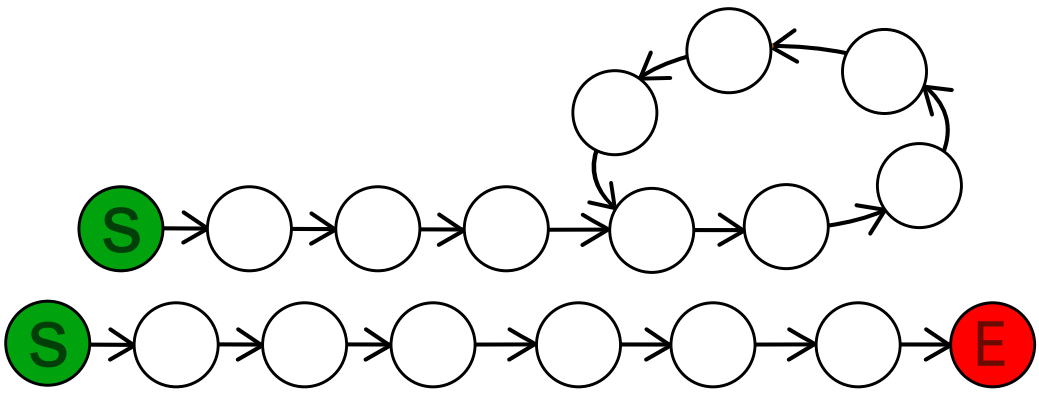
Minuten nach Start	0	1	2	3	4	5	6	7	8	9	10	11	12	13	...
	S														...
	S														...



Dies ist Informatik!

In dieser Aufgabe findet der Wettlauf auf einer besonderen Laufbahn statt. Sie besteht aus einzelnen Feldern und Pfeilen, die zum nächsten Feld zeigen. Das Besondere ist, dass die Laufbahn in einem Kreis endet, in dem die Läufer endlos laufen können. Schildkröte und Hase können sich in dieser Aufgabe nur begegnen, weil diese 6 Felder einen Kreis bzw. einen *Zyklus* bilden.

In der Informatik würde man eine Laufbahn, wie sie in der Aufgabe beschrieben ist, als *Liste* bezeichnen. Einen Kreis aus aufeinander verweisende Feldern wie in der Aufgabe würde man als *Zyklus* bezeichnen. In einer Liste verweist jeder *Knoten* auf höchstens einen anderen Knoten. Es gibt Listen mit einem Zyklus, wie in dieser Aufgabe, und Listen ohne Zyklus.



Hat eine Liste keinen Zyklus, dann besteht die Liste aus einer linearen Kette von Knoten. Dann muss es auch ein Endfeld geben, von dem kein Pfeil mehr ausgeht. Der berühmte Informatiker Robert W.

Floyd (1936−2001) hat einen Algorithmus entworfen, der einfach unterscheiden kann, ob eine Liste einen Zyklus hat oder aus einer linearen Kette besteht. Er lässt ähnlich wie in unserer Aufgabe den Hasen und die Schildkröte am Startfeld loslaufen. Wenn Schildkröte und Hase zur selben Zeit zum selben Feld kommen, gibt es einen Zyklus. In dem Moment, in dem der Hase das Endfeld oder das Feld davor erreicht, ist kein Zyklus vorhanden und der Algorithmus ist beendet.

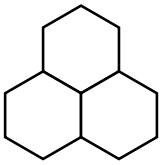
Stichwörter und Webseiten

- Liste: `https://de.wikipedia.org/wiki/Liste_(Datenstruktur)`
- Zyklus: `https://de.wikipedia.org/wiki/Zyklus_(Graphentheorie)`
- Knoten: `https://de.wikipedia.org/wiki/Knoten_(Graphentheorie)`
- Robert W. Floyd: `https://de.wikipedia.org/wiki/Robert_Floyd`
- Algorithmus: `https://de.wikipedia.org/wiki/Algorithmus`

4. Sechsecke ausmalen

Sami legt weisse Sechsecke aneinander. Dann malt er sie aus, mit drei verschiedenen Farben.

Immer, wenn drei Sechsecke genau so zusammen liegen (zwei unten und eines oben in der Mitte), müssen sie am Ende ...

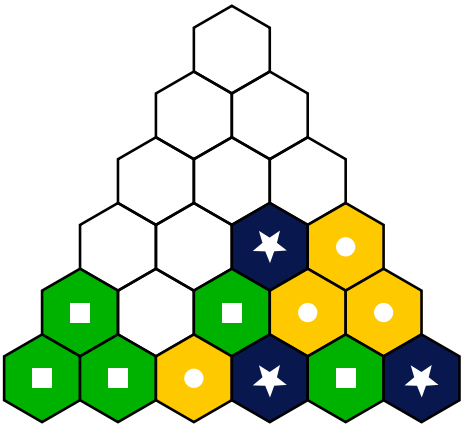


- ... alle drei die gleiche Farbe oder ...
- ... alle drei verschiedene Farben haben.

Das gefällt Sami.

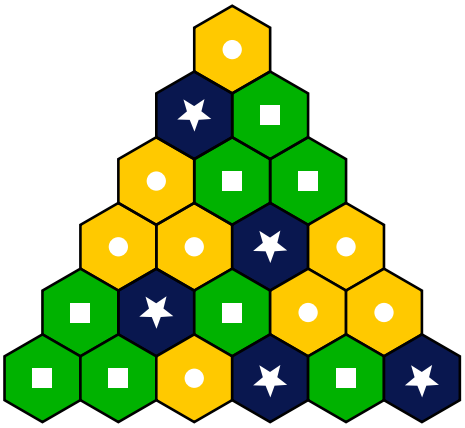
Sami hat viele Sechsecke aneinander gelegt und schon einige ausgemalt.

Male alle übrigen Sechsecke aus, so wie es Sami gefällt.



Lösung

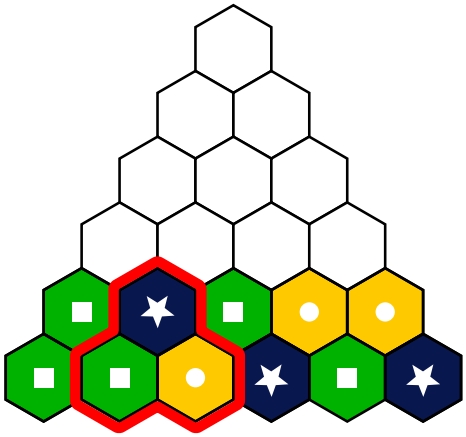
So ist es richtig:

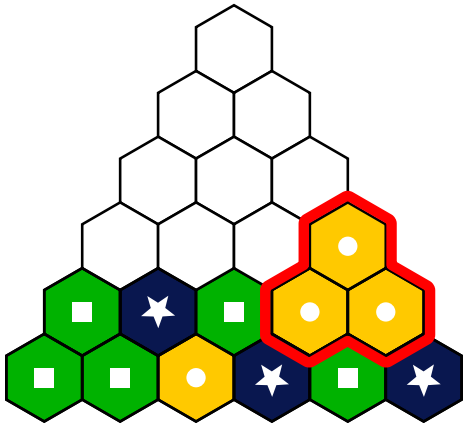


Sobald zwei Sechsecke ausgemalt sind, die in der Sechseck-Pyramide nebeneinander liegen, steht die Farbe für das Sechseck darüber fest:

Haben beide verschiedene Farben, bekommt das Sechseck darüber die dritte Farbe. So wird zum Beispiel das unterste weisse Sechseck blau ausgemalt.

Haben beide die gleiche Farbe, bekommt das Sechseck darüber ebenfalls diese Farbe. So wird das Sechseck über den beiden gelben ebenfalls gelb ausgemalt.





So kann man die übrigen Sechsecke reihenweise, von unten nach oben, nacheinander ausmalen, so wie es Sami gefällt.

Dies ist Informatik!

Wie löst man diese Biberaufgabe? Wenn man ein Sechseck ausmalt, führt man eine Aktion aus. Um die richtige Aktion (mit der richtigen Farbe) auszuwählen, muss man sich die Sechsecke darunter anschauen und prüfen, welche *Bedingung* sie erfüllen: haben sie die gleiche Farbe oder unterschiedliche Farben. Diese Prüfung, mit anschliessender Aktion, wird *wiederholt*, nämlich für jedes noch weisse Sechseck, das über zwei bereits ausgemalten Sechsecken liegt.

Aktionen, Bedingungen, Wiederholungen: Das sind die Grundbausteine eines jeden *Algorithmus*, also eines präzise beschriebenen Verfahrens, das auch als Programm für einen Computer realisiert werden kann. Beim Lösen dieser Biberaufgabe hast du also einen Algorithmus erfunden. Das ist eine der wichtigsten Aufgaben von Informatikerinnen und Informatikern: Algorithmen zu erfinden oder bereits erfundene Algorithmen nutzen und in Programme für Computer umzusetzen, um Aufgaben und Probleme mit automatischer Informationsverarbeitung zu lösen.

Stichwörter und Webseiten

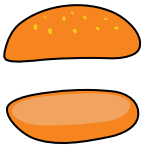
- Algorithmus: `https://de.wikipedia.org/wiki/Algorithmus`
- bedingte Anweisung: `https://de.wikipedia.org/wiki/Bedingte_Anweisung_und_Verzweigung`
- Schleife: `https://de.wikipedia.org/wiki/Kontrollstruktur#Schleife_-(Wiederholung,_Iterationsanweisung)`

5. Biberburger

Biber Jess macht Biberburger. Er folgt dazu drei Regeln:

1. Die Sauce ist direkt auf dem Fleisch.
2. Das Fleisch und der Käse liegen unter den Gurken, dem Salat und den Zwiebeln.
3. Die Zwiebeln berühren nicht die Brötchenhälften.

Biberburger-Zutaten:

Brötchen	**Fleisch**	**Sauce**	**Gurken**	**Salat**	**Zwiebeln**	**Käse**
						

Welcher Biberburger ist nach den drei Regeln zusammengestellt?

A) 

B) 

C) 

D) 

Lösung

Die richtige Antwort ist D. 

Um die Lösung zu finden, muss man bei jedem Burger prüfen, ob er so zusammengestellt ist, dass er allen drei Regeln folgt.

A) Dieser Biberburger folgt den Regeln 1 und 2. Aber die Zwiebeln berühren die obere Brötchenhälfte, also folgt er der Regel 3 nicht.

B) Dieser Biberburger folgt der Regel 1. Aber der Salat ist unter dem Fleisch und dem Käse, also wurde Regel 2 nicht befolgt.

C) Dieser Biberburger folgt Regel 2, denn das Fleisch und der Käse liegen unter den Gurken, dem Salat und den Zwiebeln. Auch folgt dieser Biberburger der Regel 3, da die Zwiebeln die Brötchenhälften nicht berühren. Allerdings ist die Sauce nicht direkt auf dem Fleisch. Somit wurde Regel 1 nicht befolgt.

D) Dieser Biberburger erfüllt alle Regeln. Biberburger D ist somit ein echter Biberburger.

Dies ist Informatik!

Die Biberburger in dieser Aufgabe sind nach drei Regeln zusammengestellt. Bei jedem der Burger, die er macht, muss Biber Jess jede der drei Regeln befolgen. Erfüllt er nur eine der Regeln nicht, ist der Burger kein Biberburger. Jede der drei Regeln ist eine Bedingung, die bei jedem Burger erfüllt sein muss, damit er ein Biberburger ist.

In der Informatik überprüft man Bedingungen (*Constraint Checking*) häufig, um herauszufinden, ob eine Lösung alle gegebenen Regeln befolgt. In dieser Überprüfung verknüpft man sämtliche Regeln (Bedingungen) mit *und*. Das bedeutet, dass alle Regeln (Bedingungen) gleichzeitig erfüllt werden müssen.

Die Prüfung, ob eine gegebene Lösung sämtliche Beschränkungen erfüllt, ist eine grundlegend andere Aufgabe als die, eine mögliche Lösung zu finden. Dies wird als *Constraint Satisfaction Problem* (dt.: Bedingungserfüllungsproblem) bezeichnet. Meistens ist es viel schwieriger, eine Lösung, die alle Bedingungen erfüllt, zu finden, als zu überprüfen, ob eine Lösung sämtliche Bedingungen erfüllt. Dies gilt sogar für einen Computer.

Stichwörter und Webseiten

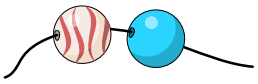
- Constraintprogrammierung: `https://de.wikipedia.org/wiki/Constraintprogrammierung`

- Constraint Satisfaction Problem:
  `https://de.wikipedia.org/wiki/Constraint-Satisfaction-Problem`
- und (im Zusammenhang mit der Logik):
  `https://de.wikipedia.org/wiki/Konjunktion_(Logik)`
- NP (Komplexitätsklasse): `https://de.wikipedia.org/wiki/NP_(Komplexitätsklasse)`

6. Matrosenkette

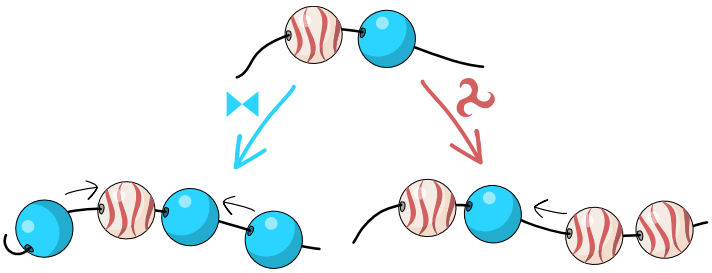
Hier ist die Anleitung für Monikas Matrosenkette mit weiss-roten Wellenperlen und einfarbigen blauen Perlen:

Du beginnst immer mit einer Wellenperle und einer blauen Perle in dieser Reihenfolge:



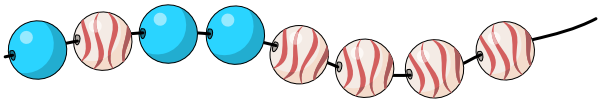
Dann kannst du die Matrosenkette verlängern, indem du

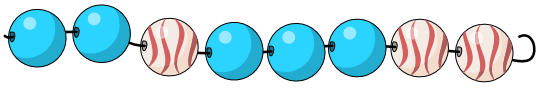
- an beiden Enden der Schnur jeweils eine blaue Perle hinzufügst (⋈)
- oder zwei Wellenperlen am rechten Ende der Schnur hinzufügst (ᔓ)

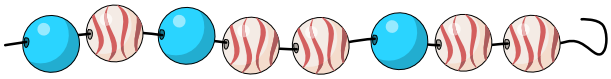


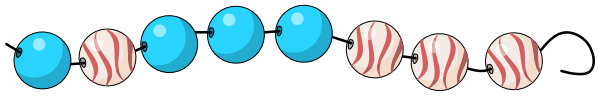
Diese Aktionen kannst du mehrfach durchführen, um immer längere Ketten aufzufädeln.

*Welche der folgenden Ketten ist **keine** von Monikas Matrosenketten?*

A) 

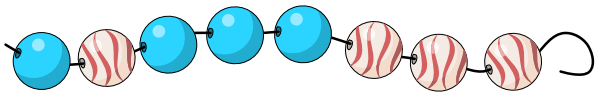
B) 

C) 

D) 

Lösung

D ist die richtige Antwort.



Die Aufgabe kannst du auf verschiedene Arten lösen.

Zum Beispiel, indem du in jeder Kette zuerst die beiden Startperlen suchst und dann eine Reihe von ⋈ - und ∿ -Aktionen ausführst.

- Bei Kette A kannst du mit der zweiten und dritten Perle beginnen und dann die Aktionen ⋈ - ∿ - ∿ ausführen.
- Bei Kette B kannst du mit der dritten und vierten Perle beginnen und dann die Aktionen ⋈ - ⋈ - ∿ ausführen.
- Bei Kette C kannst du mit der zweiten und dritten Perle beginnen und dann die Aktionen ∿ - ⋈ - ∿ ausführen.
- Wenn man jedoch die Kette D betrachtet, müssen die zweite und dritte Perle den Anfang bilden. Dann kann einmal die Aktion ⋈ ausgeführt werden, aber danach gibt es keine Aktionen mehr, um die restliche Kette zu erhalten.

Dieser Ansatz funktioniert nicht gut, wenn die Kette sehr lang ist und viele mögliche Startperlen hat. In diesem Fall kann ein dekonstruktiver Ansatz eher zum Ziel führen. Dabei entfernst du wiederholt Perlen, indem du Aktion ⋈ oder Aktion ∿ umgekehrt ausführst, solange bis nur noch zwei Perlen übrig sind.

Eine dritte Strategie macht sich die *Parität* zunutze. Nach der Anleitung für die Matrosenkette gibt es immer eine ungerade Anzahl von einfarbigen blauen Perlen und eine ungerade Anzahl von rot-weissen Wellenperlen («ungerade Parität»). Siehst du, warum das so ist?

Kette D hat eine gerade Anzahl von beiden Arten von Perlen und kann daher keine von Monikas Matrosenketten sein.

Dies ist Informatik!

Bei dieser Aufgabe kannst du nur Perlen an den Enden der Kette auffädeln. Du kannst keine Perle in der Mitte einfügen. Du kannst auch keine Perle aus der Mitte herausnehmen, ohne zuerst die Perlen vom Ende der Kette her abzufädeln.

Diese Art von Speicherstruktur, bei der man leicht Elemente an den Enden hinzufügen und entfernen kann, aber nicht in der Mitte, wird in der Informatik *double-ended queue* oder *deque-Warteschlange* genannt (deque wird wie «Deck» ausgesprochen).

Deque-Warteschlangen können verwendet werden, um den Verlauf eines Browsers zu speichern, um Druckaufträge zu planen und auch um die Gültigkeit mathematischer Ausdrücke zu überprüfen.

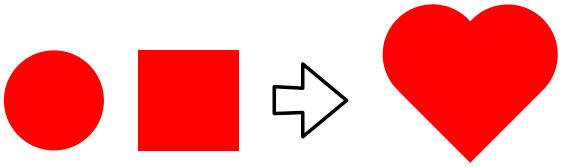
Dabei kann zum Beispiel die Überprüfung auf übereinstimmende Klammern auf ganz ähnliche Weise erfolgen wie bei der Überprüfung, ob es sich bei einer Kette um eine von Monikas Matrosenketten handelt.

Stichwörter und Webseiten

- double-ended queue: `https://de.wikipedia.org/wiki/Deque`

7. Härzige Bildli

Tina hat zwei Formen: einen Kreis und ein Quadrat. Sie wandelt diese zu einem Herz um.



Dazu verwendet sie diese drei Umwandlungen:

- *drehe*: Eine Form beliebig weit drehen.
- *verschiebe*: Eine Form beliebig verschieben.
- *verdopple*: Eine Form verdoppeln, so dass beide an derselben Stelle bleiben.

Was hat sie in welcher Reihenfolge gemacht?

A) *verdopple* Kreis, *drehe* Quadrat, *verschiebe* Kreis, *verschiebe* Kreis

B) *verdopple* Quadrat, *drehe* Quadrat, *verschiebe* Quadrat, *verschiebe* Kreis

C) *verdopple* Kreis, *drehe* Kreis, *verschiebe* Kreis, *verschiebe* Quadrat

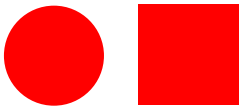
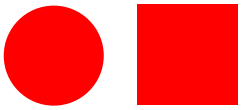
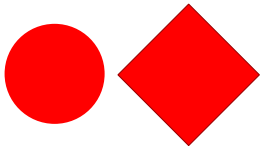
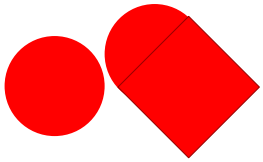
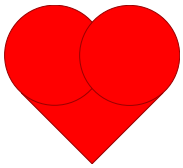
D) *verschiebe* Kreis, *verschiebe* Kreis, *verdopple* Kreis, *verschiebe* Quadrat

Lösung

Wenn man sich das Herz genau anschaut, stellt man fest, dass es aus zwei Kreisen und einem um 1/8 gedrehten Quadrat besteht. Es braucht in den Umwandlungen also ein «*verdopple* Kreis», damit man zwei Kreise hat, und ein «*drehe* Quadrat», damit das Quadrat um 1/8 gedreht werden kann. Damit fallen die Antworten B), C) und D) weg, denn:

- In der Antwort B) wird ein Quadrat verdoppelt und kein Kreis.
- In der Antwort C) wird ein Kreis gedreht, aber nicht das Quadrat.
- In der Antwort D) wird gar keine Form gedreht, also insbesondere nicht das Quadrat.

Ist aber die Antwort A) überhaupt richtig? Die Formen müssen ja noch verschoben werden! Die folgenden Umwandlungen sind vorgegeben:

- Das  ...
- ... wird durch *verdopple* Kreis zu  ...
- ... wird durch *drehe* Quadrat zu  ...
- ... wird durch *verschiebe* Kreis zu  ...
- ... wird durch *verschiebe* Kreis zu 

Deshalb ist die Antwort A) *verdopple* Kreis, *drehe* Quadrat, *verschiebe* Kreis, *verschiebe* Kreis richtig.

Dies ist Informatik!

In Bildbearbeitungsprogrammen kann man viele verschiedene Umwandlungen mit einem Bild machen. In dieser Aufgabe sind es Umwandlungen wie Drehen, Verschieben oder Verdoppeln. Das alleine genügt jedoch noch nicht: man muss dem Computer beispielsweise auch mitteilen, wie weit eine Form gedreht werden soll, oder wohin sie verschoben werden soll.

Du könntest den Weg wie man aus einem Kreis und einem Quadrat ein Herz zeichnet, natürlich auch als längeren Text beschreiben. In der Informatik ist es jedoch besser, möglichst wenige Grundumwandlungen zu verwenden, die man dann wiederholt oder unterschiedlich ausführt. Dies nennt man

Generalisieren, wenn allgemeine Lösungen aus speziellen Beispielen entwickelt werden. Solche Befehle könnten beispielsweise lauten:

- Eine Form drehen: drehe Form, wie weit
- Eine Form verschieben: verschiebe Form, wohin
- Eine Form verdoppeln: verdoppele Form

Das Bildbearbeitungsprogramm von Tina kommt einem vielleicht ungewöhnlich vor: statt dass das Bild wie bei Fotos in Form von *Pixeln* gespeichert wird, wird eine Beschreibung der Form (z. B. «Kreis, Radius 2 cm, Füllfarbe rot») gespeichert. So ist es möglich, dass zwei Formen übereinander liegen, wie die beiden Kreise, und dass nachher eines davon bewegt werden kann, ohne dass das untere überschrieben wurde. Solche Graphiken nennt man *Vektorgraphiken*. Sie werden häufig verwendet, wenn abstrakte Formen in hoher Qualität gezeichnet werden sollen. Die anderen Graphiken nennt man *Pixelgraphiken*, sie sind häufig Fotos oder fotorealistische Zeichnungen.

Stichwörter und Webseiten

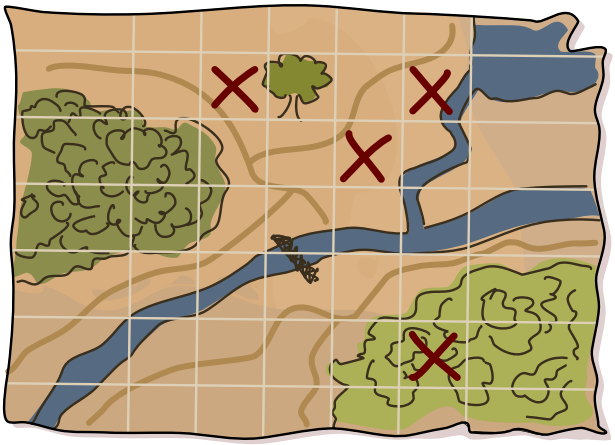
- Pixel: `https://de.wikipedia.org/wiki/Pixel`
- Rastergraphik oder Pixelgraphik: `https://de.wikipedia.org/wiki/Rastergrafik`
- Vektorgraphik: `https://de.wikipedia.org/wiki/Vektorgrafik`

8. Futter verstecken

Biber Bilbo hat zwei gute Verstecke für sein Futter. Auf einer Karte markiert er die beiden Felder, in denen die Verstecke liegen, mit ✕. Aber was ist, wenn andere Biber die Karte und damit die Verstecke finden?

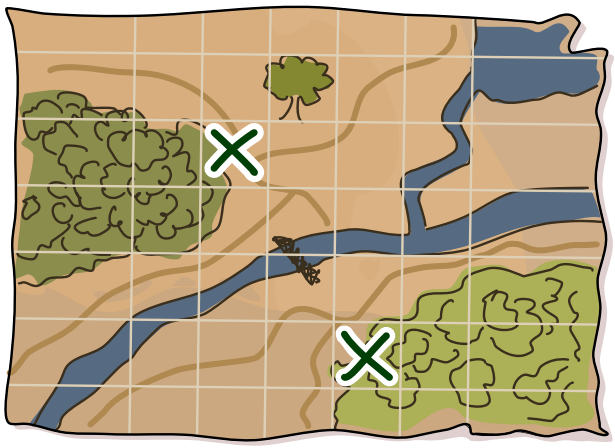
Zur Verwirrung markiert Bilbo weitere Felder mit ✕. Das macht er so, dass in jeder Zeile und Spalte der Karte eine gerade Anzahl an Feldern markiert ist. Danach entfernt er die beiden ✕ von den Feldern mit seinen Verstecken. Unten siehst du das Ergebnis.

In welchen Feldern liegen Bilbos Verstecke?

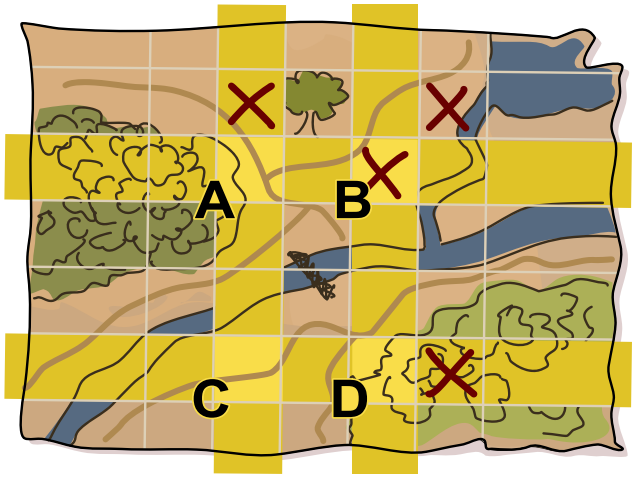


Lösung

Hier sind die beiden Verstecke:



Um sie zu finden, schauen wir uns die ursprüngliche Karte an und stellen fest, dass es zwei Zeilen und zwei Spalten gibt, in denen die Anzahl der ✕ nicht gerade ist: Zeilen 3 und 6 und Spalten 3 und 5.



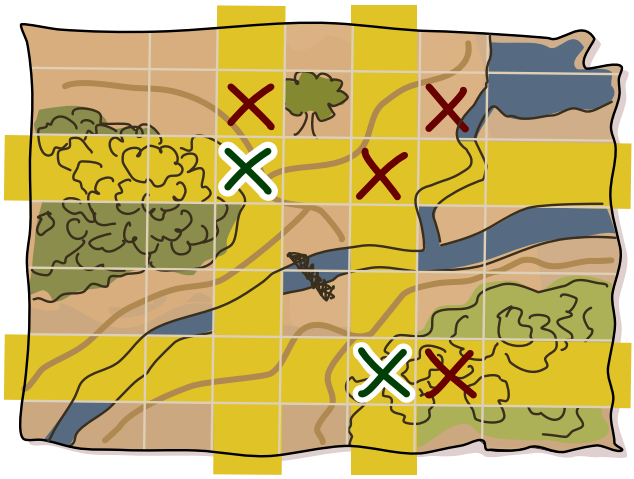
Die ✕, welche die Verstecke markieren, wurden ja entfernt. Wir wissen, dass in allen Zeilen und Spalten eine gerade Anzahl von ✕ vorhanden sein muss, nachdem die gelöschten ✕ wieder eingezeichnet sind.

Die betroffenen Zeilen und Spalten überschneiden sich und haben vier gemeinsame Felder (A, B, C und D). Diese «Schnittfelder» sind für uns besonders interessant. Wenn wir Felder ausserhalb eines Schnittfeldes mit ✕ markieren, könnten wir in einer Spalte eine gerade ✕-Anzahl erreichen, gleichzeitig wird die Anzahl in der jeweiligen Zeile ungerade und umgekehrt. Daher müssen die ✕ der beiden Verstecke auf den Schnittfeldern liegen.

Das Schnittfeld B ist bereits mit einem ✕ markiert: Es kann kein Versteck sein, da wir wissen, dass Bilbo die ✕ der Verstecke gelöscht hat.

Um also eine gerade Anzahl von ✕ in der Zeile 2 wiederherzustellen, müssen das Schnittfeld A mit einem ✕ markieren. Dort ist ein Versteck. Das andere Versteck kann nicht bei Schnittfeld C liegen,

denn dann wären drei ✕ in dieser Spalte. Das andere Versteck liegt also bei Schnittfeld D. Hier ist die Karte, bevor Bilbo die ✕ gelöscht hat, mit einer geraden Anzahl von ✕ in jeder Zeile und Spalte:



Dies ist Informatik!

Bilbo verwendet hier einen Trick, der in der Informatik häufig verwendet wird: *Paritätsbits*. Diese sind Teil einer Reihe von Techniken, die als *Fehlererkennungs-* und *Fehlerkorrekturcodes* bekannt sind. Die Idee dabei ist, dass wir immer dann, wenn wir Daten als eine Reihe von *Bits* (können entweder 0 oder 1 sein) speichern oder übertragen, zusätzliche Bits hinzuzufügen, die uns helfen, zu erkennen, ob Übertragungs- oder Speicherfehler aufgetreten sind – typischerweise, wenn ein Bit verdreht wurde, also wenn ein Bit als 1 gesendet und fälschlicherweise als 0 empfangen wurde oder umgekehrt.

Wenn wir zum Beispiel einen einfachen Fehlererkennungscode verwenden, würde ein Paritätsbit hinzugefügt, sodass die Anzahl der Einsen immer gerade ist. 0110101 würde eine 0 hinzugefügt werden, um 01101010 zu werden (die Anzahl der 1er-Bits bleibt gerade). Wenn das zweite Bit umgedreht wurde und die Nachricht jetzt 00101010 gesendet wird, dann erfüllt diese empfangene Nachricht nicht die Forderung nach gerader Parität (drei Bits sind 1er-Bits). Beachte, dass diese Methode kein Problem erkennen kann, wenn mehr als 1 Bit fehlerhaft ist.

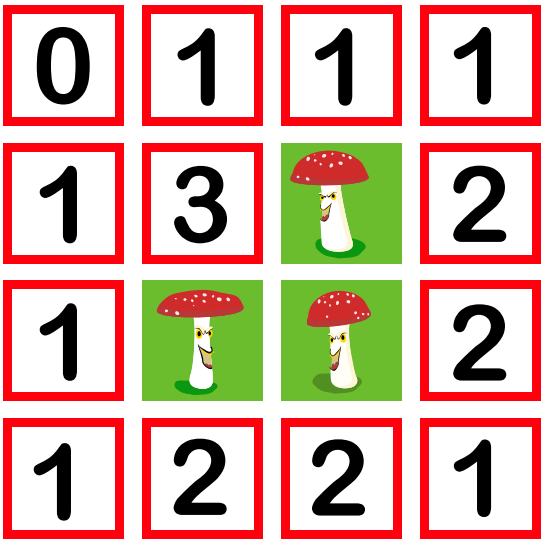
Stichwörter und Webseiten

- Bit: `https://de.wikipedia.org/wiki/Bit`
- Paritätsbits: `https://de.wikipedia.org/wiki/Paritätsbit`
- Fehlerkorrekturverfahren: `https://de.wikipedia.org/wiki/Fehlerkorrekturverfahren`

9. Achtung Fliegenpilz

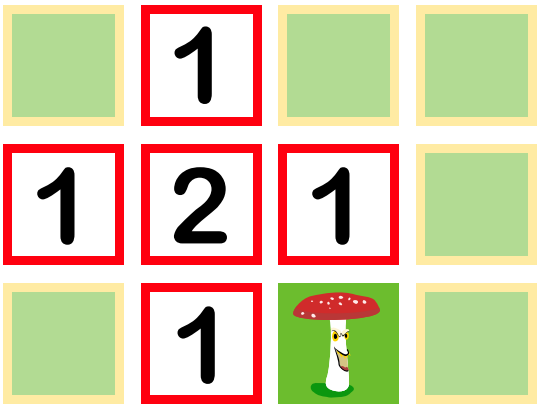
Beim Spiel «Achtung Fliegenpilz» ist zu Beginn genau ein Fliegenpilz zu sehen. Alle anderen Felder des Spielbretts sind zugedeckt. Deckst du ein Feld auf, erscheint entweder ein weiterer Fliegenpilz oder die Anzahl der Fliegenpilze auf den Nachbarfeldern. Wenn du alle Felder aufdeckst, auf denen kein Fliegenpilz versteckt ist, hast du gewonnen.

Hier ist ein Beispiel für ein vollständig aufgedecktes Spielbrett:



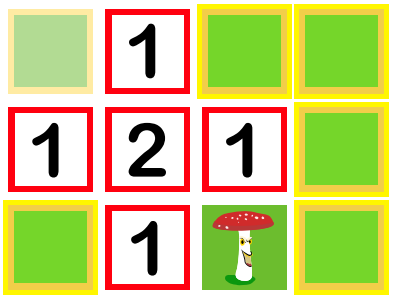
Du hast ein neues Spiel begonnen und bereits einige Felder aufgedeckt.

Auf welchen der übrigen Feldern ist sicher kein Fliegenpilz?

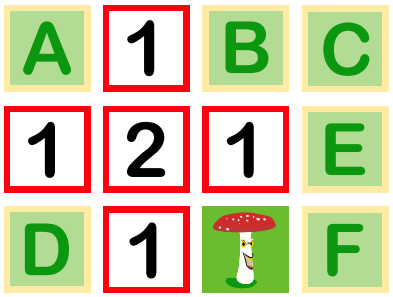


Lösung

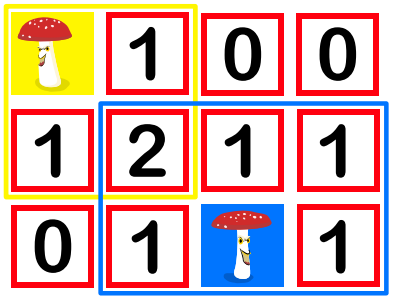
So ist es richtig:



Um die richtige Antwort zu erklären, versehen wir die zugedeckten Felder mit Buchstaben. Ausserdem sagen wir, dass eine Zahl N auf einem Feld «verbraucht» ist, wenn bereits auf N Nachbarfeldern dieser Zahl je ein Fliegenpilz aufgedeckt ist; auf anderen Nachbarfeldern kann dann kein Fliegenpilz mehr sein.



- Auf Feld D ist kein Fliegenpilz, weil die Zahl 1 rechts daneben verbraucht ist.
- Auf den Feldern B, C, E und F ist kein Fliegenpilz, weil die gemeinsame Nachbarzahl 1 dieser Felder verbraucht ist.
- Auf Feld A ist ein Fliegenpilz, weil sonst die Nachbarzahlen 1, 2 und 1 die Anzahl der Fliegenpilze auf ihren Nachbarfeldern nicht korrekt angeben würden.



Also ist auf Feld A ein Fliegenpilz versteckt. Die Felder B, C, D, E und F dürfen aufgedeckt werden.

Dies ist Informatik!

Wie sind wir vorgegangen? Manchmal muss man mit einer Vermutung beginnen und logisch weiter denken. Wenn man einen Widerspruch findet, geht man zurück und folgt der nächstliegenden Vermutung. Dabei handelt es sich um ein «zielgerichtetes» Suchen und nicht um ein Ausprobieren.

Wie würde ein Computer dieses Beispiel lösen? Wenn mindestens ein Feld mit einem Fliegenpilz aufgedeckt ist, können einfache Regeln aufgestellt werden. Zum Beispiel, wenn das Feld mit der Zahl 1 bereits ein Nachbarfeld mit einem aufgedeckten Fliegenpilz abdeckt, dann kann es keinen weiteren Fliegenpilz als Nachbarn geben. Wenn diese Regeln für jede Zahl genau formuliert sind, könnte ein Computer sie Schritt für Schritt als *Anweisungen* ausführen. Dann hätten wir letztlich einen *Algorithmus*, den man «nur» ausführen müsste, um im Spiel (mit mindesten einem aufgedeckten Fliegenpilz) erfolgreich zu sein.

Stichwörter und Webseiten

- Minesweeper: `https://de.wikipedia.org/wiki/Minesweeper`
- Anweisung (Informatik): `https://de.wikipedia.org/wiki/Anweisung_(Programmierung)`
- Algorithmus: `https://de.wikipedia.org/wiki/Algorithmus`

10. Schrauben und Muttern

Ben steht am Fliessband und verarbeitet Bauteile: Muttern  und Schrauben  .

Ben geht strikt nach folgendem Verfahren vor:

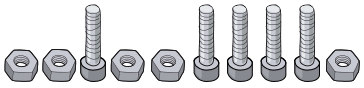
- Ben nimmt das nächste Bauteil vom Fliessband herunter.
- Wenn Ben eine Mutter vom Fliessband genommen hat, legt er sie in den Eimer.
- Wenn Ben eine Schraube vom Fliessband genommen hat, nimmt er eine Mutter aus dem Eimer, schraubt sie auf die Schraube und legt das fertige Teil in den Kasten.

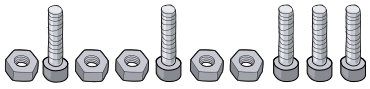
Bei diesem Verfahren können zwei Fehler auftreten:

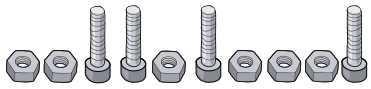
1. Ben nimmt eine Schraube vom Fliessband, aber es ist keine Mutter im Eimer, die er aufschrauben könnte.
2. Ben hat alle Bauteile vom Fliessband verarbeitet, aber es sind immer noch Muttern im Eimer.

Der Eimer für die Muttern ist ausreichend gross und zu Beginn leer. Welche der Folgen von Muttern und Schrauben kann Ben ohne Fehler von links nach rechts verarbeiten?

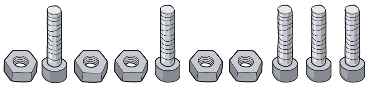
A) 

B) 

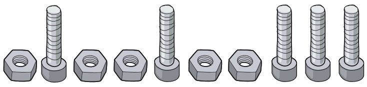
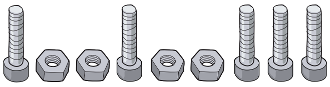
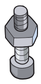
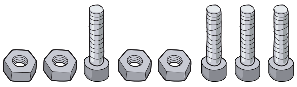
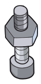
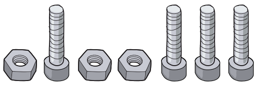
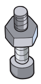
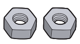
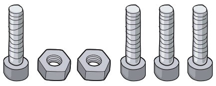
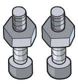
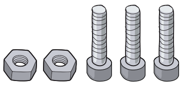
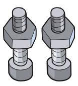
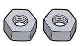
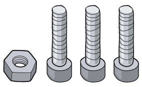
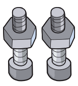
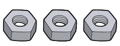
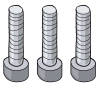
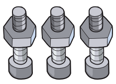
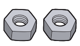
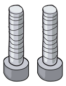
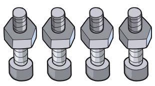
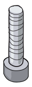
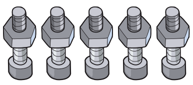
C) 

D) 

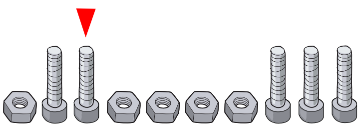
Lösung

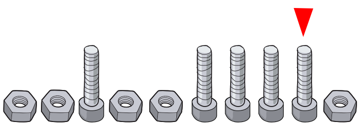
Die richtige Antwort ist C): 

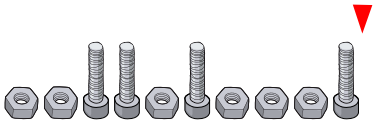
Die Tabelle zeigt den Zustand des Kastens für die fertigen Teile, des Eimers für die Muttern und des Fliessbands.

Kasten	Eimer	Fliessband
leer	*leer*	
leer		
	*leer*	
		
		
		
		
		
		
		
	*leer*	*leer*

Warum sind die anderen Antworten falsch?

A)  führt zu einem Fehler an der markierten Stelle. Dann hat Ben eine Schraube aufgenommen, aber es ist keine Mutter mehr im Eimer.

B)  führt zu einem Fehler an der markierten Stelle. Ben hat bisher 4 Muttern auf vier Schrauben geschraubt. Der Eimer ist also leer. Nun hat er aber eine fünfte Schraube aufgenommen, für die er keine Mutter mehr hat.

D)  führt zu einem Fehler, nachdem die gesamte Folge verarbeitet worden ist. Denn es wurden 4 Muttern auf 4 Schrauben geschraubt und 2 Muttern bleiben übrig.

Dies ist Informatik!

Ben verarbeitet Bauteile, die eins nach dem anderen von dem Fliessband geliefert werden. Dabei verwendet er einen grossen Eimer zum Zwischenspeichern der Muttern. Eine ähnliche Anordnung wird in der *theoretischen Informatik* als Modell für *Algorithmen* verwendet, die eine bestimmte Klasse von Problemen lösen können: *Kellerautomaten*.

Ein Kellerautomat verarbeitet Daten (Zahlen oder Zeichen), die er nach und nach als Eingabe erhält. Er besitzt einen einzigen unendlich grossen Speicher, einen Keller. Im Unterschied zum Eimer in der Aufgabe haben die Elemente im Keller eine bestimmte Reihenfolge und man kann aus einem Keller nur das Element herausnehmen, das man als letztes hineingegeben hat («last in first out», LIFO). Ein Kellerautomat kann verwendet werden, um eine *kontextfreie Sprache* zu erkennen.

In der Informatik versteht man unter einer Sprache eine Menge von Zeichenketten, die nach bestimmten Regeln geformt worden sind. Ein einfacher Typ von Sprachen sind kontextfreie Sprachen. Ein Beispiel für eine kontextfreie Sprache sind alle wohlgeformten Klammerausdrücke. Bei einem wohlgeformten Klammerausdruck wird jede öffnene Klammer wieder geschlossen. Wohlgeformt sind z.B. ((( ))) und (()()). Nicht wohlgeformt sind dagegen dagegen (((() und ())(. Man kann sich die Muttern und Schrauben in der Aufgabe als öffnende und schliessende Klammern vorstellen. Dann verarbeitet Ben eine Folge von Bauteilen auf dem Fliessband nur dann ohne Fehler, wenn sie einen wohlgeformten Klammerausdruck darstellt. Das Prüfen von Klammerausdrücken ist eine wichtige Aufgabe eines Compilers, der Programmtexte in ausführbare Programme übersetzt. Denn in Programmtexten der meisten Programmiersprachen kommen geschachtelte Funktionsaufrufe und arithmetische Ausdrücke mit Klammern vor.

Stichwörter und Webseiten

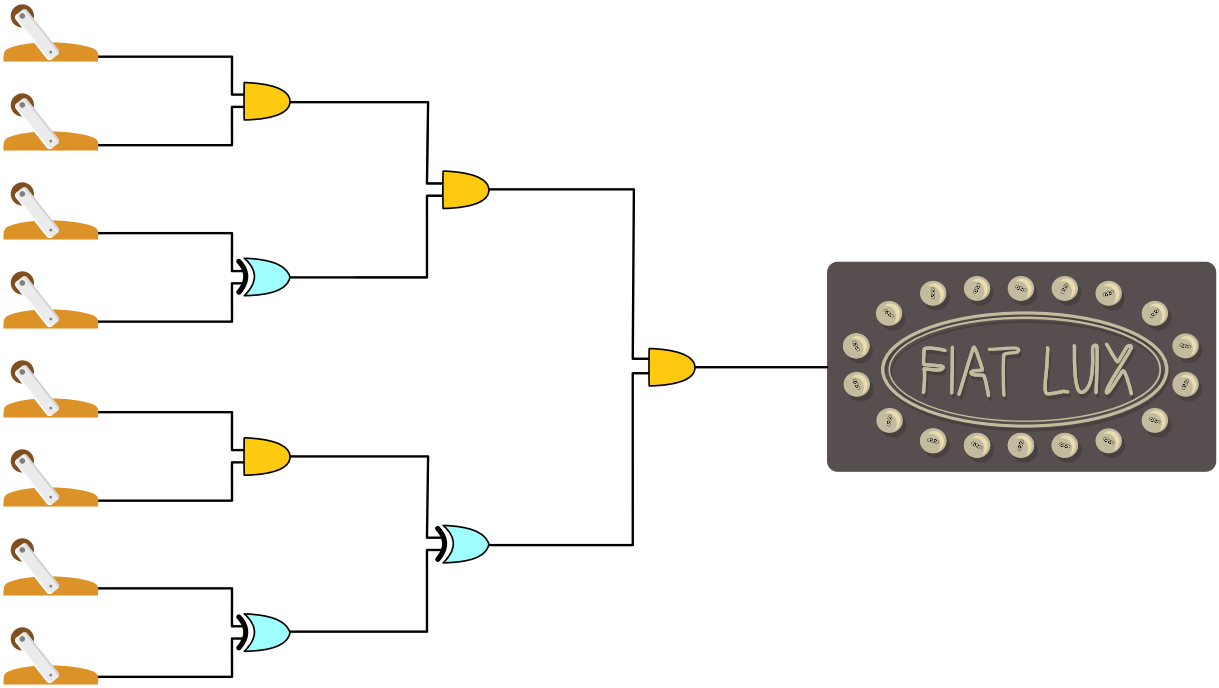
- Theoretische Informatik: `https://de.wikipedia.org/wiki/Theoretische_Informatik`
- Kellerautomat: `https://de.wikipedia.org/wiki/Kellerautomat`
- Kontextfreie Sprache: `https://de.wikipedia.org/wiki/Kontextfreie_Sprache`

11. FIAT LUX!

Das Spiel «FIAT LUX!» hat 8 Schalter, die an  oder aus  sein können. Aus diesen Schaltern führen Drähte, die durch einige Bauteile und schliesslich zu einer Leuchtreklame führen.

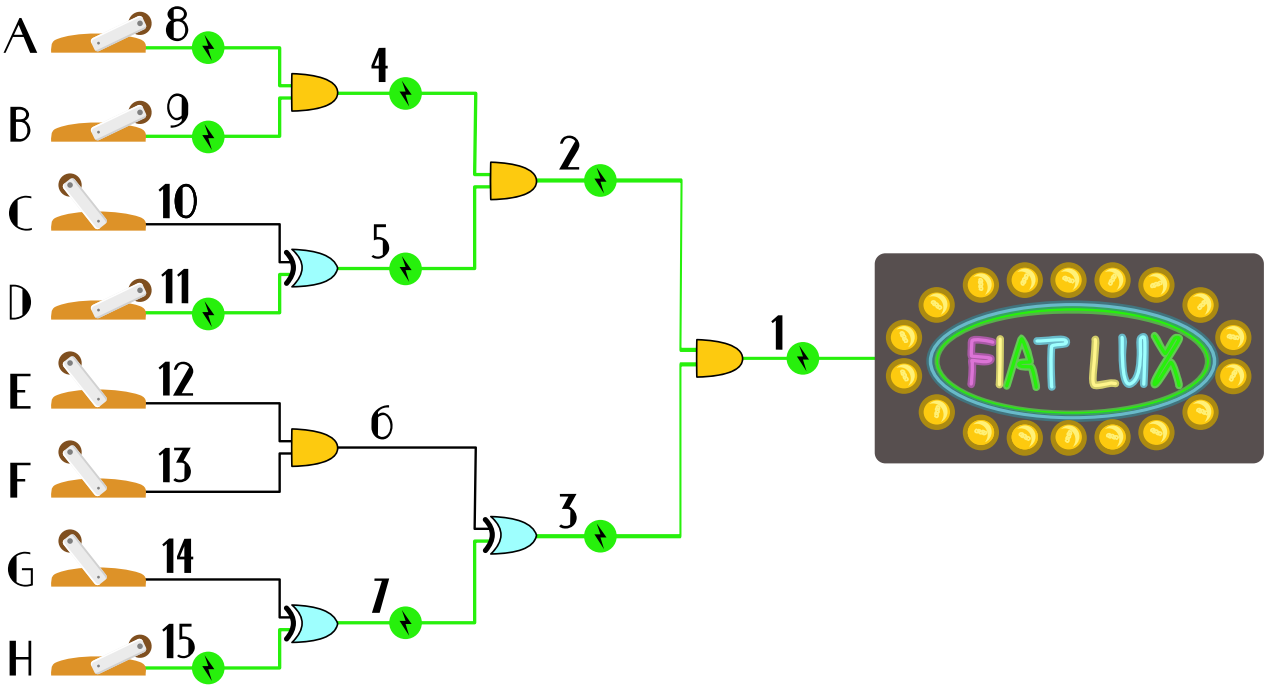
Der Ausgang vom  -Bauteil ist nur dann an, wenn beide eingehenden Drähte an sind. Der Ausgang vom  -Bauteil ist dann an, wenn genau einer der eingehenden Drähte an ist.

Welche Schalter müssen an  *sein, um am Ende die Leuchtreklame einzuschalten?*



Lösung

Eine mögliche Lösung ist diese:



Man kann sie sich einfach erarbeiten, indem man von hinten das Problem löst. Der angeschlossene Draht 1 ist mit einem  -Bauteil verbunden. Damit dies am Ausgang *an* ist, müssen beide eingehenden Drähte 2 und 3 *an* sein.

- Draht 2 ist mit einem  -Bauteil verbunden. Damit dies am Ausgang *an* ist, müssen beide eingehenden Drähte 4 und 5 *an* sein.
- Draht 3 ist mit einem  -Bauteil verbunden. Damit dies am Ausgang *an* ist, muss genau einer der beiden eingehenden Drähte *an* sein, zum Beispiel Draht 7. Dann muss Draht 6 *aus* sein.
- Draht 4 ist mit einem  -Bauteil verbunden. Damit dies am Ausgang *an* ist, müssen beide eingehenden Drähte 8 und 9 *an* sein, also die beiden Schalter A und B ebenfalls *an* sein:  .
- Draht 5 ist mit einem  -Bauteil verbunden. Damit dies am Ausgang *an* ist, muss genau einer der beiden eingehenden Drähte *an* sein, zum Beispiel der Draht 11. Dann muss Draht 10 *aus* sein. Also muss Schalter C *aus*  und Schalter D *an*  sein.
- Draht 6 ist mit einem  -Bauteil verbunden. Damit dies am Ausgang *aus* ist, muss mindestens einer der eingehenden Drähte 12 und 13 *aus* sein, also können sogar beide Schalter E und F *aus* sein:  .
- Draht 7 ist mit einem  -Bauteil verbunden. Damit dies am Ausgang *an* ist, muss genau einer der beiden eingehenden Drähte *an* sein, zum Beispiel Draht 15. Dann muss Draht 14 *aus* sein. Also muss Schalter G *aus*  und Schalter H *an*  sein.

Alternativen gibt es bei den  –Bauteilen, denn hier kann man entscheiden, welcher der beiden eingehenden Drähte *an* ist. Zudem kann man an dem  –Bauteil mit Draht 6 als Ausgang entscheiden, ob keiner oder einer der beiden *an* ist, da in beiden Fällen der Ausgang *aus* bleibt. Damit bei dem  –Bauteil mit Draht 6 der Ausgang *an* ist, müssen beide Eingänge ebenfalls *an* sein. In diesem Fall müssen die beiden Eingänge des  –Bauteils mit Draht 7 als Ausgang entweder beide *an* oder beide *aus* sein, damit Draht 7 *aus* ist. Das ergibt 16 verschiedene mögliche Kombinationen:

Schalter								**Draht**	
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**6**	**7**
immer *an*		**genau einer *an***		**beide *an*, wenn Draht 6 *an*, sonst maximal einer *an***		**genau einer *an*, wenn Draht 7 *an*, sonst beide *an* oder *aus***		**genau einer *an***	
An	An	An	Aus	An	An	An	An	An	Aus
An	An	Aus	An	An	An	An	An	An	Aus
An	An	An	Aus	An	An	Aus	Aus	An	Aus
An	An	Aus	An	An	An	Aus	Aus	An	Aus
An	An	An	Aus	An	Aus	An	Aus	Aus	An
An	An	Aus	An	An	Aus	An	Aus	Aus	An
An	An	An	Aus	An	Aus	Aus	An	Aus	An
An	An	Aus	An	An	Aus	Aus	An	Aus	An
An	An	An	Aus	Aus	An	An	Aus	Aus	An
An	An	Aus	An	Aus	An	An	Aus	Aus	An
An	An	An	Aus	Aus	An	Aus	An	Aus	An
An	An	Aus	An	Aus	An	Aus	An	Aus	An
An	An	An	Aus	Aus	Aus	An	Aus	Aus	An
An	An	Aus	An	Aus	Aus	An	Aus	Aus	An
An	An	An	Aus	Aus	Aus	Aus	An	Aus	An
An	An	Aus	An	Aus	Aus	Aus	An	Aus	An

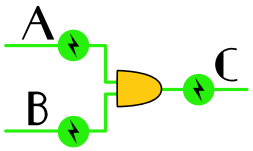
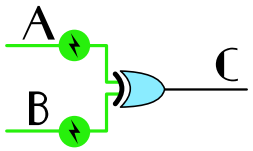
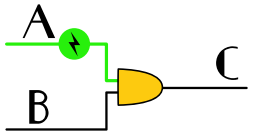
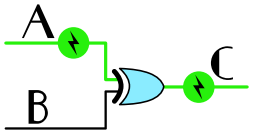
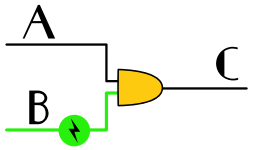
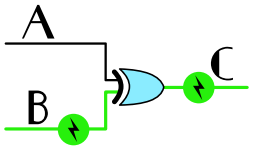
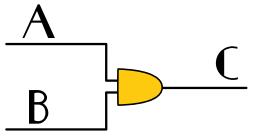
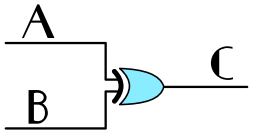
Dies ist Informatik!

Durch die Drähte dieser Aufgabe kann entweder Strom fliessen oder nicht, die Schalter sind also entweder an oder aus. In der Informatik repräsentieren solche Zustände den Wert einer *booleschen Variablen*. Diese werden oftmals auch als *wahr* oder *falsch* respektive als *1* oder *0* benannt.

Heutige Computer funktionieren in der Regel auch nur mit diesen beiden Zuständen. Das liegt unter anderem daran, dass im Kern des Computers Milliarden von *Transistoren* verbaut sind, deren Ein- und Ausgänge ebenfalls nur an oder aus sind.

Aus mehreren Transistoren kann man dann *logische Schaltungen* bauen. Zwei solche Schaltungen kommen in dieser Aufgabe vor: das  –Bauteil ist ein *Und-Gatter*, dessen Ausgang nur dann an ist, wenn beide Eingänge an sind. Das  –Bauteil ist ein *Exklusiv-Oder-Gatter*, dessen Ausgang

dann an ist, wenn genau einer der beiden Eingänge an ist. Man kann diese auch als *Wahrheitstabelle* aufschreiben:

Eingänge		**UND-Gatter**		**Exklusiv-ODER-Gatter**	
Eingang A	**Eingang B**	**Bild**	**Ausgang C**	**Bild**	**Ausgang C**
An	An		An		Aus
An	Aus		Aus		An
Aus	An		Aus		An
Aus	Aus		Aus		Aus

Weitere verbreitete Gatter sind das *Oder-Gatter*, dessen Ausgang dann an ist, wenn mindestens einer der beiden Eingänge an ist, und das *Nicht-Gatter*, dessen Ausgang genau dann an ist, wenn der Eingang nicht an ist. Häufig verbaut man eine Kombinationen aus einem Und-Gatter und einem Nicht-Gatter, weil man dies mit besonders wenigen Transistoren gebaut werden kann. Die Wahrheitstabellen sind:

Eingang A	**Eingang B**	**Ausgang Oder-Gatter**	**Ausgang Nicht-Und-Gatter**
An	An	An	Aus
An	Aus	An	An
Aus	An	An	An
Aus	Aus	Aus	An

Eingang	**Ausgang Nicht-Gatter**
An	Aus
Aus	An

Durch geschickte Kombinationen von *Logik-Gattern* kann ein Computer sehr schnell komplizierte Rechnungen durchführen.

Auf einer höheren Ebene werden die Logik-Gatter auch beim Programmieren verwendet: wenn das Ausführen eines Programmteils auf mehreren Bedingungen beruht, können diese Bedingungen mit Hilfe von *logischen Operatoren*, die genau so funktionieren, kombiniert werden. Dies findet man auch

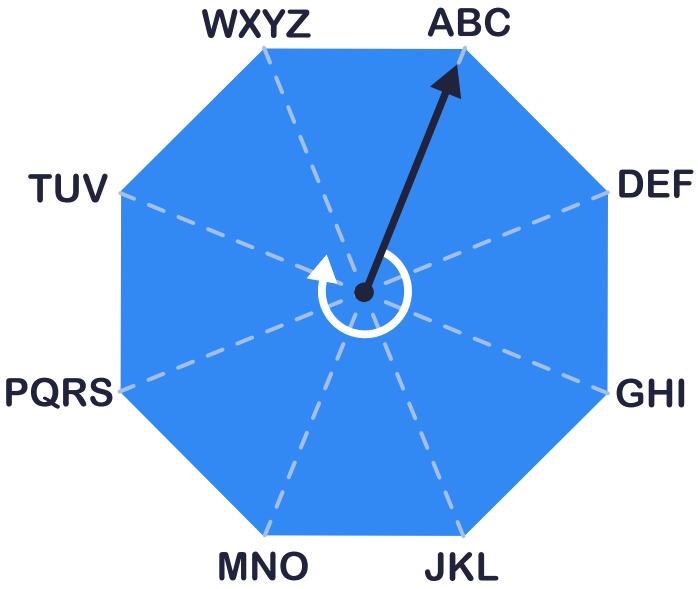
in Computerprogrammen. Manchmal muss ein Programm «Entscheideungen» darüber treffen, was als nächstes zu tun ist, je nachdem, ob eine Sache (oder manchmal auch mehrere Dinge) zuvor passiert sind.

Stichwörter und Webseiten

- boolesche Variable: `https://de.wikipedia.org/wiki/Boolean`
- Transistoren: `https://de.wikipedia.org/wiki/Transistor`
- logische Schaltung: `https://de.wikipedia.org/wiki/Digitaltechnik`
- Und-Gatter: `https://de.wikipedia.org/wiki/Und-Gatter`
- Exclusiv-Oder-Gatter: `https://de.wikipedia.org/wiki/Exklusiv-Oder-Gatter`
- Wahrheitstabelle: `https://de.wikipedia.org/wiki/Wahrheitstabelle`
- Oder-Gatter: `https://de.wikipedia.org/wiki/Oder-Gatter`
- Nicht-Gatter: `https://de.wikipedia.org/wiki/Nicht-Gatter`
- Logik-Gatter: `https://de.wikipedia.org/wiki/Logikgatter`
- logische Operator: `https://de.wikipedia.org/wiki/Logischer_Operator`

12. Code 8

Mit dieser Scheibe werden Klartexte zu Geheimtexten verschlüsselt:



Am Anfang steht der Zeiger der Scheibe auf «ABC».

Jeder Buchstaben wird einzeln verschlüsselt. Dazu werden zwei Ziffern ermittelt:

- Die erste Ziffer gibt an, um wie viele Positionen der Zeiger im Uhrzeigersinn gedreht wird. Dann steht der Zeiger auf dem Block mit dem Buchstaben, der verschlüsselt werden soll.
- Die zweite Ziffer gibt an, der wievielte Buchstabe in dem Block verschlüsselt werden soll.

Das Wort «PAAR» wird beispielsweise als $51 - 31 - 81 - 53$ verschlüsselt.

Was bedeutet der Geheimtext 22-61-62-74?

A) HANS

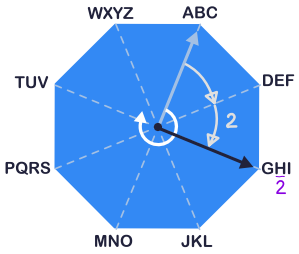
B) HAUS

C) HALLO

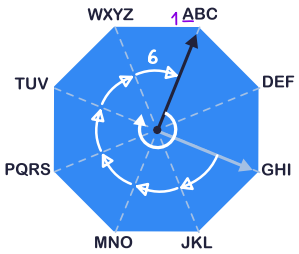
D) HALS

E) HAUT

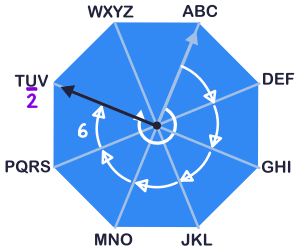
Lösung



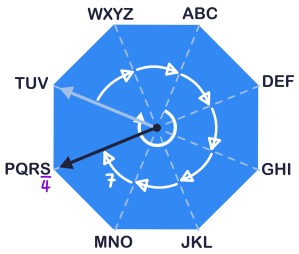
22 bedeutet, dass der Zeiger vom Block «ABC» zum Block «GHI» gedreht wird (erste Ziffer 2), und dass der zweite Buchstabe «H» genommen wird (zweite Ziffer 2).



61 bedeutet, dass der Zeiger nun vom Block «GHI» zum Block «ABC» gedreht wird (erste Ziffer 6), und dass der zweite Buchstabe «A» genommen wird (zweite Ziffer 1).



62 bedeutet, dass der Zeiger nun vom Block «ABC» zum Block «TUV» gedreht wird (erste Ziffer 6), und dass der zweite Buchstabe «U» genommen wird (zweite Ziffer 2).



74 bedeutet, dass der Zeiger nun vom Block «TUV» zum Block «PQRS» gedreht wird (erste Ziffer 7), und dass der vierte Buchstabe «S» genommen wird (zweite Ziffer 4).

Damit ist die Antwort B) «HAUS» korrekt.

Man hätte auch schneller auf diese Lösung kommen können: Die Antwort C) HALLO kann gar nicht in Frage kommen, da sie aus fünf Buchstaben besteht, der Geheimtext aber nur vier Buchstaben repräsentiert. Da der letzte Buchstabe mit einer 4 als zweiter Ziffer verschlüsselt ist, kann er nur «S» oder «Z» sein. Nur die Antworten A), B) und D) erfüllen dies. Der Buchstabe davor ist muss aus dem Buchstabenblock sieben Drehungen gegen den Uhrzeigersinn sein, also aus dem Block «TUV». Damit kann es nur noch die Antwort B) «HAUS» sein.

Dies ist Informatik!

Seit tausenden von Jahren versucht der Mensch, Informationen so zu verstecken, dass nur die Empfänger sie entziffern können. Was mit Papierstreifen, die um einen Stab gewickelt wurden, anfing («Skytale»), entwickelte sich über Transpositionschiffrem wie dem «Caesar-Code» und *polyalphabetischen Verschlüsselungsverfahren* (wie dem «Vigenére-Verfahren») zur modernen *Public-Key-Kryptographie* (wie zum Beispiel «GnuPG», das unter anderem das «RSA-Verfahren» nutzt).

Das Verschlüsselungsverfahren aus dieser Aufgabe ist ein polyalphabetisches Verschlüsselungsverfahren, denn derselbe Buchstabe wird nicht notwendigerweise mit demselben Geheimtext verschlüsselt: der Buchstabe «A» im Beispiel wird am Anfang als 31, aber am Ende als 81 verschlüsselt. Prinzipiell sind diese Verschlüsselungsverfahren heute alle mit Hilfe von Computern schnell und einfach zu entziffern.

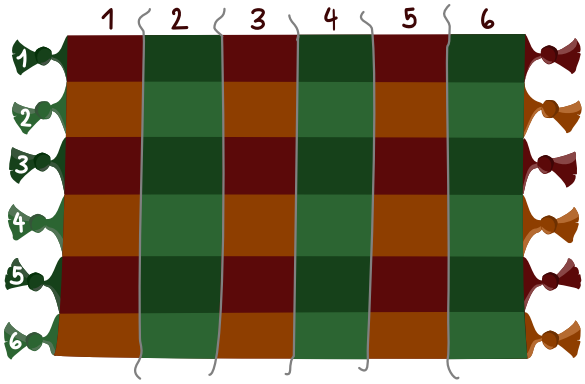
In diesem Fall ist das Entziffern jedoch denkbar einfach: es gibt nur genau einen Schlüssel, um einen Text zu verschlüsseln. Selbst wenn man die Startposition des Zeigers nicht bei ABC sondern bei irgendeinem Block starten lassen könnte, hätte man nur acht verschiedene Schlüssel ... da ist selbst der Caesar-Code, der über 2000 Jahre alt ist, «sicherer». Nun kann man noch argumentieren, dass das Geheime gar nicht der Schlüssel sondern das Verschlüsselungsverfahren ist. Aber das *Kerckhoffs'sche Prinzip*, das Auguste Kerckhoffs (1835 bis 1903) 1883 formuliert hat, und das bis heute gilt, macht deutlich, dass die Sicherheit eines *Kryptosystems* nicht auf dem Geheimhalten eines Verschlüsselungsverfahrens beruhen darf, denn dies könnte zu leicht anderen bekannt werden.

Stichwörter und Webseiten

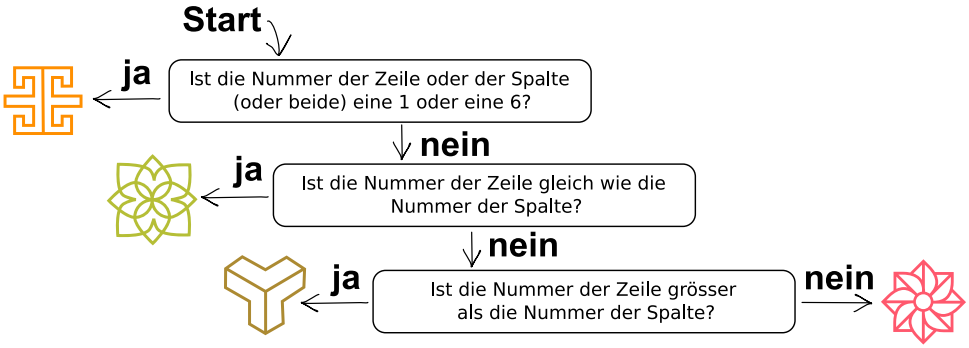
- Caesar-Code: `https://de.wikipedia.org/wiki/Caesar-Verschlüsselung`
- Polyalphabetische Substitution: `https://de.wikipedia.org/wiki/Polyalphabetische_Substitution`
- Verschlüsselungsverfahren: `https://de.wikipedia.org/wiki/Verschlüsselungsverfahren`
- Vigenére-Verfahren: `https://de.wikipedia.org/wiki/Vigenère-Chiffre`
- Public-Key-Kryptographie: `https://de.wikipedia.org/wiki/Asymmetrisches_Kryptosystem`
- GnuPG: `https://de.wikipedia.org/wiki/GNU_Privacy_Guard`
- RSA-Verfahren: `https://de.wikipedia.org/wiki/RSA-Kryptosystem`
- Kerckhoffs'sche Prinzip: `https://de.wikipedia.org/wiki/Kerckhoffs'_Prinzip`
- Auguste Kerckhoffs: `https://de.wikipedia.org/wiki/Auguste_Kerckhoffs`
- Kryptosystems: `https://de.wikipedia.org/wiki/Kryptosystem`
- Kryptographie: `https://de.wikipedia.org/wiki/Kryptographie`

13. Teppichmuster

Hale ist eine türkische Künstlerin. Sie gestaltet ein Teppichmuster mit einem Raster aus sechs Zeilen und sechs Spalten.

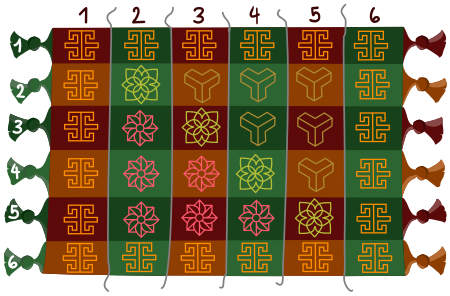


Hale nummeriert die Zeilen und Spalten. Für jedes Feld des Rasters gibt es also die Nummer der Zeile und die der Spalte. Hales Angestellte sollen in jedes Feld ein Symbol setzen. Hale hat ihnen dazu diese Anleitung gegeben:

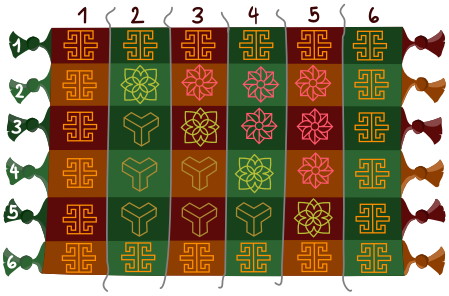


Wie wird der Teppich aussehen?

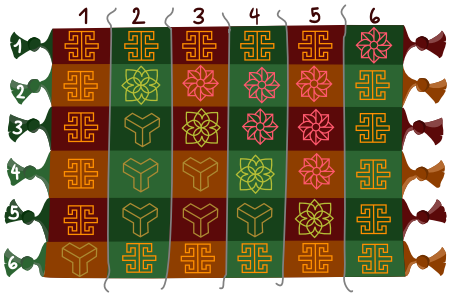
A)



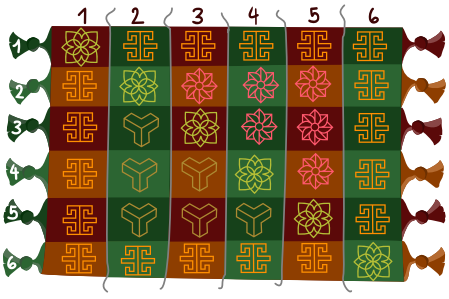
B)



C)



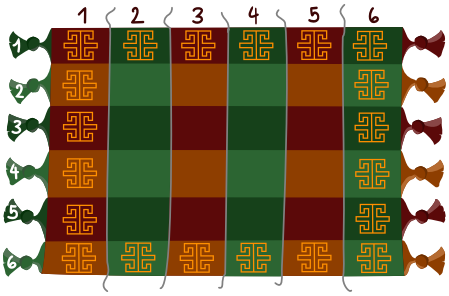
D)



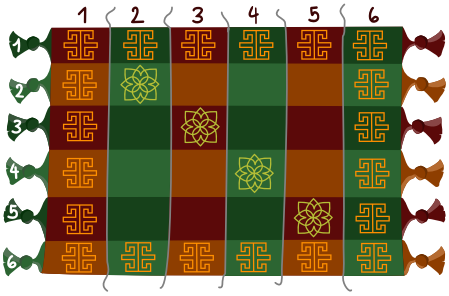
Lösung

Die richtige Antwort ist B).

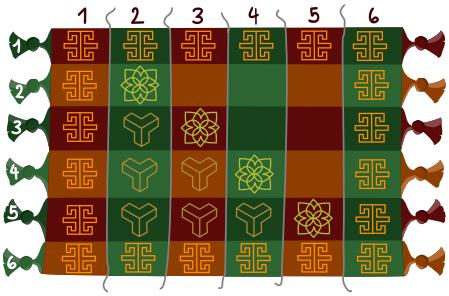
Die erste Frage des Bildes wird für alle Felder am Rand des Gitters mit «Ja» beantwortet. Denn jedes Randfeld befindet sich in der 1. oder 6. Spalte oder in der 1. oder 6. Zeile. Diese Felder erhalten das Symbol und es ergibt sich die folgende Anordnung:



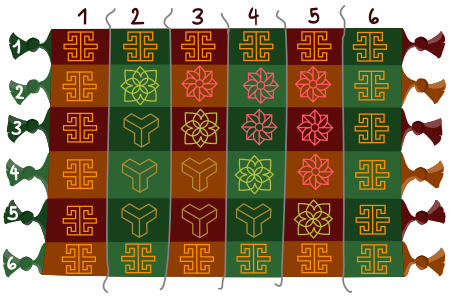
Die zweite Frage wird für alle Felder auf der Diagonalen mit «Ja» beantwortet, denn auf der Diagonalen sind Spalten- und Zeilennummern gleich. Diese Felder erhalten das Symbol  und das Teppichmuster sieht so aus:



Gemäss der dritten Frage erhalten alle Felder, deren Zeilennummer grösser als die Spaltennummer ist, das Symbol  .

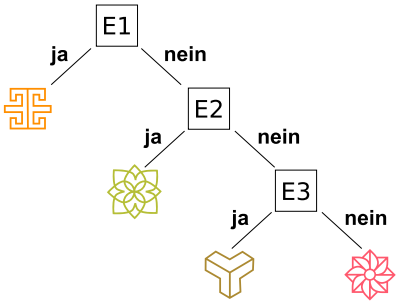


Bei den übrigen Feldern wird die dritte Frage mit «Nein» beantwortet. Das heisst die Zeilennummer ist nicht grösser als die Spaltennummer. Alle diese Felder werden mit dem Symbol  gefüllt. So ergibt sich das Teppichmuster aus Antwort B.



Dies ist Informatik!

Das Bild, das die Künstlerin Hale als Anleitung entwickelt hat, nennt man in der Informatik einen *Entscheidungsbaum*. Wie ein richtiger Baum besteht ein Entscheidungsbaum aus Verzweigungen. An jeder Verzweigung (E1 - E3) steht eine Frage, die mit «Ja» oder «Nein» beantwortet wird. Wenn man den Baum von oben nach unten durchläuft, die Fragen beantwortet und den passenden Linien folgt, führt man eine Entscheidung herbei.



In der Aufgabe ist der Entscheidungsbaum das Herzstück einer Anleitung für das Weben eines Teppichs. Jede Person, die diese Anleitung beim Weben verwendet, stellt genau den gleichen Teppich her. Im Prinzip könnte auch eine Maschine den Teppich produzieren, sofern sie die Anleitung lesen und verstehen kann.

In der Informatik nennt man eine solche eindeutige Anleitung einen *Algorithmus*. Wenn ein Algorithmus in einer *Programmiersprache* geschrieben ist und von einem Computer ausgeführt werden kann, spricht man von einem *Computerprogramm*.

Im Alltag hat man häufig mit Computerprogrammen zu tun, die Entscheidungen treffen: Die Ampelsteuerung entscheidet, wann die Fussgängerampel grün wird. Das Betriebssystem des Handys entscheidet, wann es in den Energiesparmodus wechselt. Die automatische Passkontrolle am Flughafen entscheidet, ob der Reisepass gültig ist.

Hinter all diesen Programmen stecken Entscheidungsbäume.

Stichwörter und Webseiten

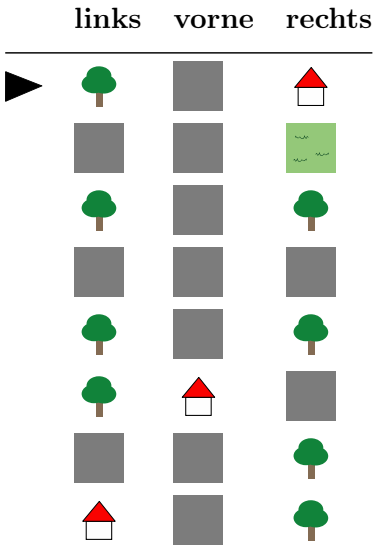
- Entscheidungsbaum: `https://de.wikipedia.org/wiki/Entscheidungsbaum`
- Algorithmus: `https://de.wikipedia.org/wiki/Algorithmus`
- Programmiersprache: `https://de.wikipedia.org/wiki/Programmiersprache`
- Computerprogramm: `https://de.wikipedia.org/wiki/Computerprogramm`

14. Roboter Tina

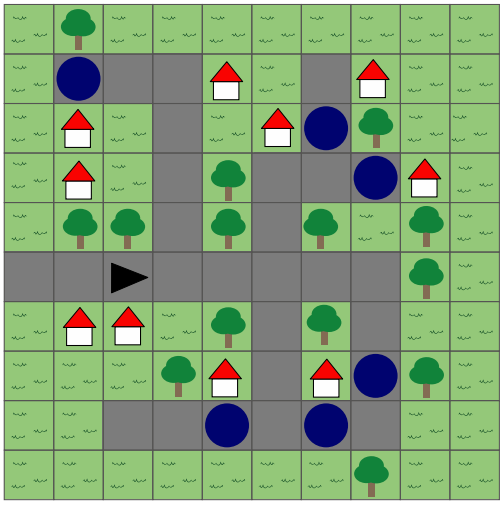
Roboter Tina liefert Post aus. Tina benutzt dazu eine Landkarte, die in Felder eingeteilt ist. Tina bewegt sich der Strasse entlang auf ein benachbartes Feld nach links, rechts oder vorne (also nicht diagonal).

Für die Navigation hat Tina drei Sensoren. Sobald Tina ein Feld betritt (und bevor Tina sich drehen kann), erkennen sie, was sich auf den Feldern links, rechts und vor Tina befindet.

Die Tabelle dokumentiert, was Tinas Sensoren auf jedem Feld ihres Weges erkannt haben. Tina startet auf dem Feld, in Richtung des Pfeiles.

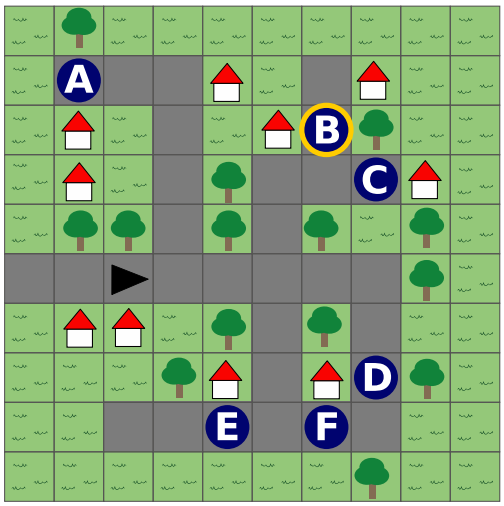


An welchem der dunkelblauen Punkte befindet sich Tina am Ende ihres Weges?



Lösung

Die korrekte Antwort ist Punkt B.

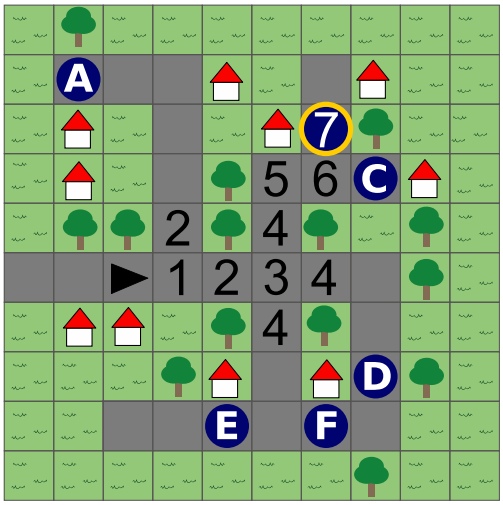


Schritt	links	vorne	rechts
▶	🌳	■	🏠
1	■	■	
2	🌳	■	🌳
3	■	■	■
4	🌳	■	🌳
5	🌳	🏠	■
6	■	■	🌳
7	■	🌳	🌳

Hier ist es effizient, sich auf die sechs Zielpunkte zu fokussieren und zu schauen, ob Sensorangaben von Schritt 7 «🏠 ■ 🌳» passen könnten. Damit kann man C, E und F ausschliessen. Die Sensorangaben von Schritt 6 sind «■ ■ 🌳», somit kann man A und D ausschliessen.

Alternativ kann man versuchen, den in der Tabelle dokumentierten Weg zu gehen. Der Weg zu Punkt B ist der einzige, der dem entspricht.

Wenn man Tinas Weg anhand der Informationen der Sensoren nachvollzieht, kann man nicht immer sofort entscheiden, wohin Tina sich bewegt hat. Im Schritt 4 würde Tina links und rechts Bäume sehen, egal in welche der drei Richtungen sie sich bewegen würde. In dieser Situation muss man auch die Sensorinformationen nach der nächsten Bewegung berücksichtigen um Schritt 4 eindeutig bestimmen zu können.



Dies ist Informatik!

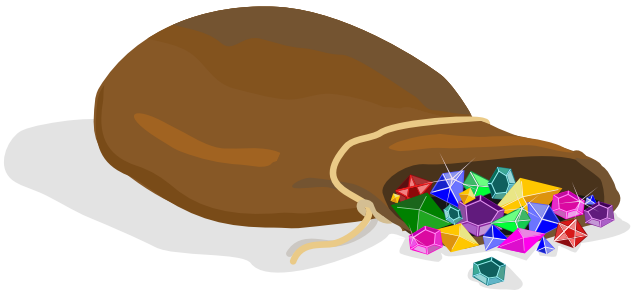
In diesem Task begegnen wir den *Roboter* Tina. Roboter sind speziell ausgestattete Computer, die Informationen aus ihrer Umwelt mit Hilfe von *Sensoren* erfassen, diese Informationen automatisch (d.h. mit einem Programm) verarbeiten und aufgrund des Resultats eine Aktion in ihrer Umwelt, mittels sogenannter *Aktoren*, selbstständig ausführen. Tinas Sensoren erfassen zunächst den Inhalt der Felder links, vorne und rechts. Konkret könnten uns vorstellen, dass die Sensoren Fotos aufnehmen und dass aus der automatisierten Analyse dieser Bilder geometrische Daten extrahiert werden, die der Computer zu einem Haus, einem Baum oder eine Strasse zuordnen kann. Tinas Fahrwerk, d.h. die Aktoren, könnte dann so gesteuert werden, dass Felder mit Bäumen oder einem Haus umgefahren werden.

Selbstfahrende Autos sind berühmte Beispiele solcher Roboter. Sie sind mit zahlreichen Sensoren ausgestattet, die nicht nur die Geschwindigkeit oder die aktuelle Position, sondern auch der Abstand vom Strassenrand messen und Objekte auf der Strasse oder am Strassenrand und vieles, vieles mehr erfassen. Diese Informationen werden mittels zum Teil sehr komplexer Programme verarbeitet, die zum Beispiel Kinder erkennen, die potentiell die Strasse überqueren könnten und diese von einem Strassenschild unterscheiden können. In vielen solcher Szenarien ist das sogenannte *maschinelle Lernen* die Schlüsseltechnologie. Im Fall von selbstfahrenden Autos lernen die Computer aus vielen vorgegebenen Beispiele, wie man Kinder von Strassenschildern unterscheidet. Die Aktoren sind dann zum Beispiel die Bremsen, die selbständig bzw. ohne menschliche Mitwirkung aktiviert werden.

Stichwörter und Webseiten

- Roboter: `https://de.wikipedia.org/wiki/Roboter`
- Sensor: `https://de.wikipedia.org/wiki/Sensor`
- Aktor: `https://de.wikipedia.org/wiki/Aktor`
- maschinelles Lernen: `https://de.wikipedia.org/wiki/Maschinelles_Lernen`

15. Wertvolle Steine



Peter hat einige Edelsteine. Sie sind alle unterschiedlich wertvoll.

Sarah kennt Peters Edelsteine, aber nicht deren Wert. Sie will wissen, welcher Stein der wertvollste ist.

Dazu macht sie Folgendes dreimal:

- Sie wählt vier von Peters Steinen aus und fragt ihn, welcher davon der wertvollste Stein ist.

Jedesmal wählt sie die vier Steine beliebig neu aus, und Peter gibt ihr jedesmal eine ehrliche Antwort.

Danach weiss Sarah, welcher Stein der wertvollste ist.

Wie viele Edelsteine kann Peter höchstens haben?

A) 8 Edelsteine

B) 10 Edelsteine

C) 11 Edelsteine

D) 12 Edelsteine

Lösung

Antwort B) ist richtig: 10 Edelsteine

Wenn Peter 10 Edelsteine hat, kann Sarah bei den ersten beiden Fragen insgesamt acht verschiedene Edelsteine auswählen. Die beiden «Gewinner» der einzelnen Fragen (also die Steine, die jeweils die wertvollsten der vier gewählten Steine sind) können auch «Gesamtsieger» sein, also der insgesamt wertvollste Stein. Die anderen sechs Steine scheiden aus. Bei der letzten Frage wählt sie die beiden Gewinner und die zwei bisher noch nicht gewählten Steine aus. Der Gewinner dieser Frage muss der Gesamtsieger sein.

Für 10 Steine kann Sarah also (unter anderem) so vorgehen, um den wertvollsten Stein zu finden. Wenn Peter 11 Steine hat, kann sie das leider nicht schaffen:

Wenn Sarah, wie oben, bei den ersten beiden Fragen insgesamt acht verschiedene Steine vergleicht, verbleiben die beiden Gewinner und drei weitere Steine, also einer zu viel, um den Gesamtsieger mit der dritten Frage zu ermitteln. Wenn Sarah hingegen den Gewinner der ersten Frage bei der zweiten Frage mit 3 «neuen» Steinen vergleicht, kennt sie danach den wertvollsten der sieben gewählten Steine. Diesen Stein muss sie mit den vier weiteren Steinen vergleichen. Auch das ist ein Stein zu viel für die dritte Frage.

Wenn Sarah bei 11 Steinen für die ersten beiden Fragen nur sechs oder noch weniger verschiedene Steine auswählt, oder wenn Peter mehr als 12 Steine hat, kann Sarah nach drei Fragen erst recht nicht wissen, welcher Stein der wertvollste ist.

Dies ist Informatik!

Bei dieser Aufgabe geht es um einen *Algorithmus*, der durch Bedingungen eingeschränkt wird. In unserem Fall darf Sarah nur drei Fragen stellen und jede Frage darf nur 4 Elemente enthalten.

Trotz dieser Einschränkung funktioniert dieser Algorithmus gut für Sammlungsgrößen kleiner als 11, versagt aber ansonsten.

Es kann verschiedene Gründe geben, Algorithmen Beschränkungen aufzuerlegen. Beispielsweise könnte man fordern, dass eine Operation in einer festen Zeitspanne abgeschlossen werden muss, was in Echtzeit-Betriebssystemen erforderlich ist. Ein weiterer Grund könnte sein, dass Vorgänge externe Kosten verursachen oder einen Bauteil beschädigen können.

Es ist kein Problem, dass der Algorithmus ab einer bestimmten Schwelle versagt, solange sichergestellt wird, dass diese Schwelle nie erreicht wird. Beispielsweise darf die eingeschränkte Strategie dieser Aufgabe niemals für Sammlungen mit mehr als 10 verwendet werden.

Stichwörter und Webseiten

- Algorithmus: `https://de.wikipedia.org/wiki/Algorithmus`
- Zeitkomplexität: `https://de.wikipedia.org/wiki/Zeitkomplexität`

A. Aufgabenautoren

Gulgun Afacan
Esraa Almajhad
Waël Almoman
Leo Barichello
Liam Baumann
Wilfried Baumann
Linda Björk Bergsveinsdóttir
Tobias Berner
Sarah Chan
Byeonggyu Cho
Christian Datzko
Susanne Datzko
Justina Dauksaite
Nora A. Escherle
Gerald Futschek
Mark Edward M. Gonzales
Adam Grodeck
Yasemin Gülbahar
Benjamin Hirsch
Alisher Ikramov
Dauksaite Justina
Dong Yoon Kim
Hakin Kim
Jihye Kim
Seulki Kim
Vaidotas Kinčius
Lidija Kralj
Regula Lacher
Taina Lehtimäki
Karolína Miková
Jelena Milojkovic
Ágnes Erdősné Németh
Jean-Philippe Pellet
Margot Phillipps
Zsuzsa Pluhár
Wolfgang Pohl
John-Paul Pretti
Le Quang Quan
Susannah Quidilla
Chris Roffey
Kirsten Schlüter
Giovanni Serafini
Yeh Yi Shan
Bernadette Spieler
Alieke Stijf
Goran Sukovic
Monika Tomcsányiová
Ahto Truu
Troy Vasiga
Michael Weigend
Kyra Willekes

B. Sponsoring: Wettbewerb 2022

HASLERSTIFTUNG

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.

Kanton Zürich
Volkswirtschaftsdirektion
Amt für Wirtschaft und Arbeit

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

UBS

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

verkehrshaus.ch

http://www.verkehrshaus.ch/

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.

senarclens
leu+partner
strategische kommunikation

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

ABZ
AUSBILDUNGS- UND BERATUNGSZENTRUM
FÜR INFORMATIKUNTERRICHT

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

C. Weiterführende Angebote

IT Feuer: `https://it-feuer.ch/`
In der Schweiz engagieren sich zahlreiche Organisationen für die Nachwuchsförderung in Informatik. Die Initiative «IT-Feuer» möchte diese vorhandenen Kräfte bündeln und einen Beitrag leisten, das Thema in der Öffentlichkeit schweizweit bekannter zu machen. Das IT-Feuer präsentiert eine grosse Palette an Angeboten für Lehrpersonen sowie Schüler*innen und Schulklassen.

Das Lehrmittel zum Informatik-Biber

Module
Verkehr – Optimieren
Musik – Komprimieren
Geheime Botschaften – Verschlüsseln
Internet – Routing
Apps
Auszeichnungssprachen

`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.

CoetryLab: `https://www.coetry-lab.org/`
Das Team des CoetryLab möchte Kindern und Jugendlichen den Zugang zum Programmieren und zu Medien ermöglichen. Das CoetryLab soll die Anlaufstelle ausserschulischen Experimentierens und Gestaltens sein und allen die Coding-Welt eröffnen. Eigene Ideen können kreativ umgesetzt und im Team oder alleine Webseiten, Apps, Games und vieles mehr entwickelt werden.

Roteco: `https://www.roteco.ch/de/`
Das ROTECO Projekt bildet eine Community für und mit Lehrpersonen, welche Schülerinnen und Schüler auf die digitale Gesellschaft vorbereiten möchten. Lehrpersonen können auf dieser Plattform Erfahrungen austauschen, erhalten Informationen zu den neusten Kursen und Workshops und finden Aktivitäten, welche sich direkt in den Unterricht integrieren lassen.

I learn it: `http://ilearnit.ch/`
In thematischen Modulen können Kinder und Jugendliche auf dieser Website einen Aspekt der Informatik auf deutsch und französisch selbständig entdecken und damit experimentieren. Derzeit sind sechs Module verfügbar.

Werden Sie SVIA Mitglied – `http://svia-ssie-ssii.ch/svia/mitgliedschaft` und unterstützten Sie damit den Informatik-Biber.
Ordentliches Mitglied des SVIA kann werden, wer an einer schweizerischen Primarschule, Sekundarschule, Mittelschule, Berufsschule, Hochschule oder in der übrigen beruflichen Aus- und Weiterbildung unterrichtet.
Als Kollektivmitglieder können Schulen, Vereine oder andere Organisationen aufgenommen werden.