INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Aufgaben und Lösungen 2022

Schuljahre 9/10

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

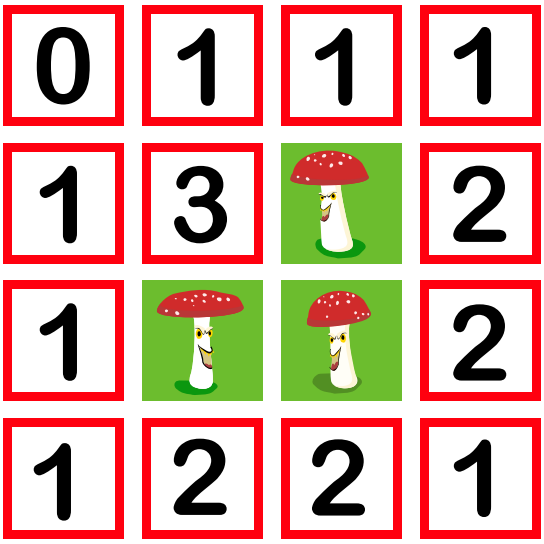
`https://www.informatik-biber.ch/de/kontaktieren/`
`https://www.informatik-biber.ch/`

Inhaltsverzeichnis

1. Achtung Fliegenpilz

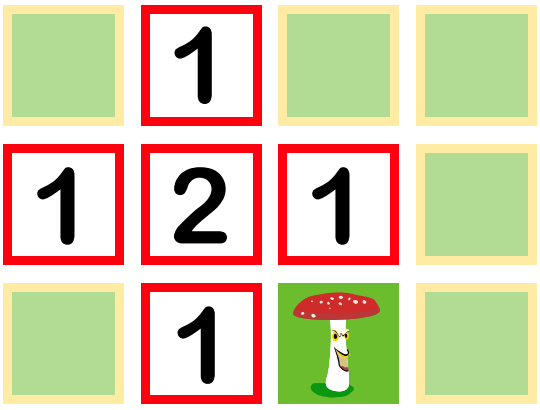
Beim Spiel «Achtung Fliegenpilz» ist zu Beginn genau ein Fliegenpilz zu sehen. Alle anderen Felder des Spielbretts sind zugedeckt. Deckst du ein Feld auf, erscheint entweder ein weiterer Fliegenpilz oder die Anzahl der Fliegenpilze auf den Nachbarfeldern. Wenn du alle Felder aufdeckst, auf denen kein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

Hier ist ein Beispiel für ein vollständig aufgedecktes Spielbrett:



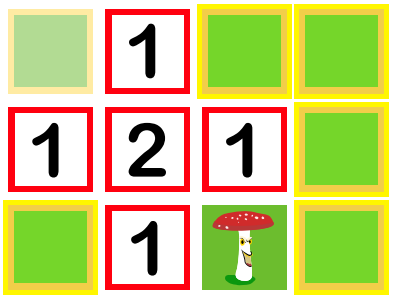
Du hast ein neues Spiel begonnen und bereits einige Felder aufgedeckt.

Auf welchen der übrigen Feldern ist sicher kein Fliegenpilz?

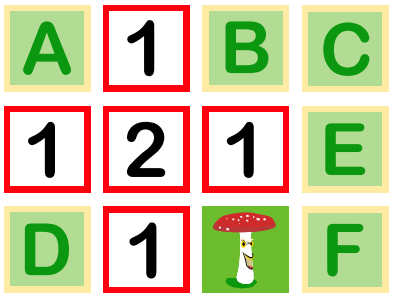


Lösung

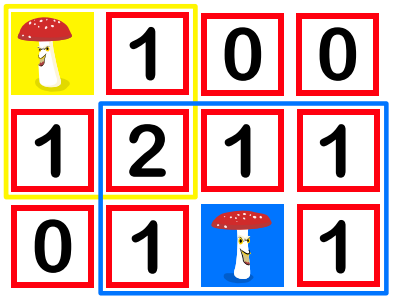
So ist es richtig:



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`

2. Schrauben und Muttern

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

Ben geht strikt nach folgendem Verfahren vor:

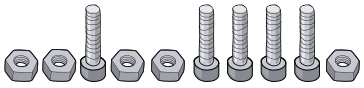
- Ben nimmt das nächste Bauteil vom Fliessband herunter.
- Wenn Ben eine Mutter vom Fliessband genommen hat, legt er sie in den Eimer.
- Wenn Ben eine Schraube vom Fliessband genommen hat, nimmt er eine Mutter aus dem Eimer, schraubt sie auf die Schraube und legt das fertige Teil in den Kasten.

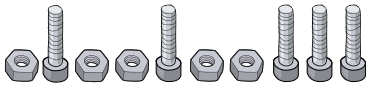
Bei diesem Verfahren können zwei Fehler auftreten:

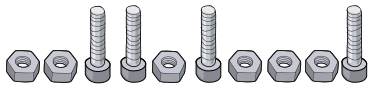
1. Ben nimmt eine Schraube vom Fliessband, aber es ist keine Mutter im Eimer, die er aufschrauben könnte.
2. Ben hat alle Bauteile vom Fliessband verarbeitet, aber es sind immer noch Muttern im Eimer.

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?

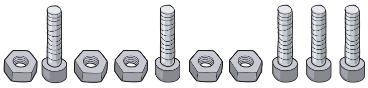
A) 

B) 

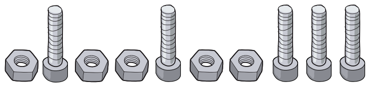
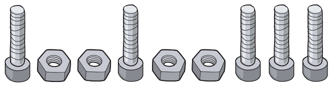
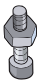
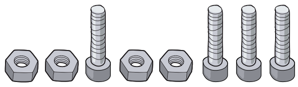
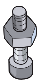
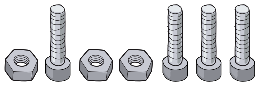
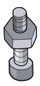
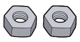
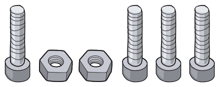
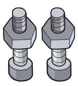
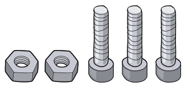
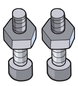
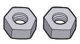
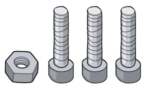
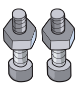
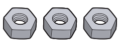
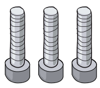
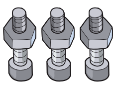
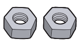
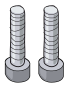
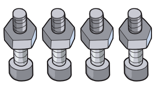
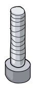
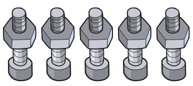
C) 

D) 

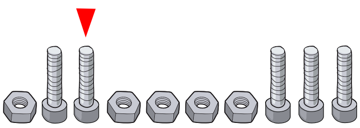
Lösung

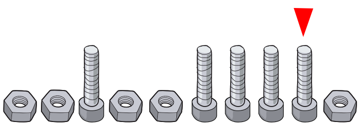
Die richtige Antwort ist C): 

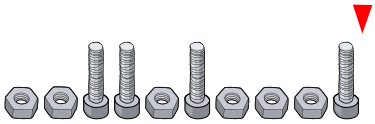
Die Tabelle zeigt den Zustand des Kastens für die fertigen Teile, des Eimers für die Muttern und des Fliessbands.

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

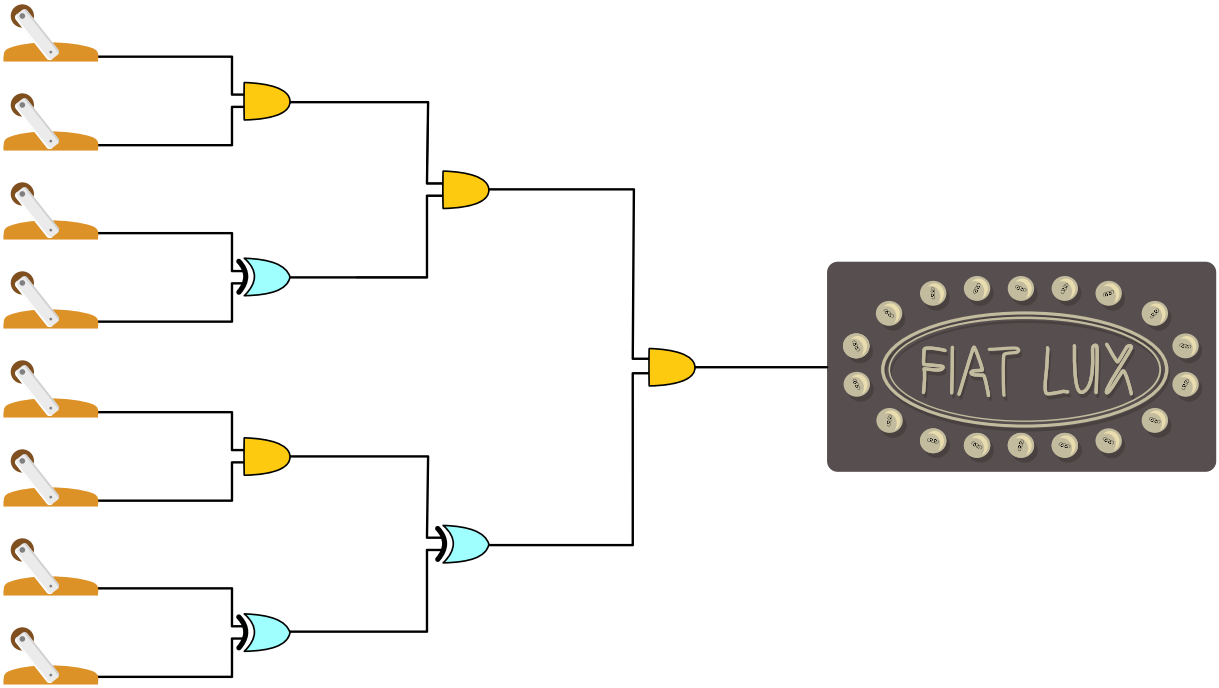
- Theoretische Informatik: `https://de.wikipedia.org/wiki/Theoretische_Informatik`
- Kellerautomat: `https://de.wikipedia.org/wiki/Kellerautomat`
- Kontextfreie Sprache: `https://de.wikipedia.org/wiki/Kontextfreie_Sprache`

3. 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.

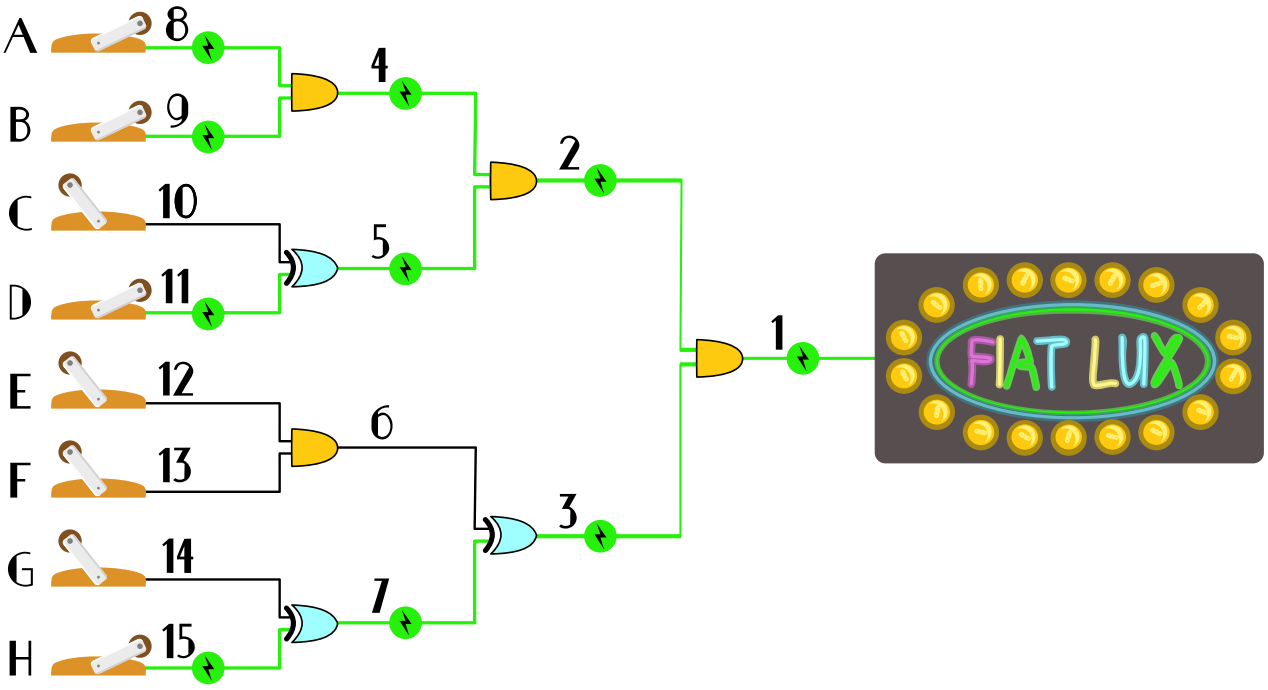
Der Ausgang vom  -Bauteil ist nur dann an, wenn beide eingehenden Drähte an sind. Der Ausgang vom  -Bauteil ist dann an, wenn genau einer der eingehenden Drähte an ist.

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

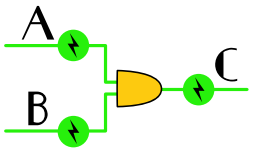
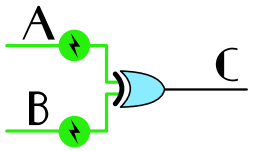
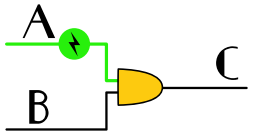
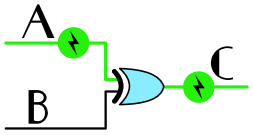
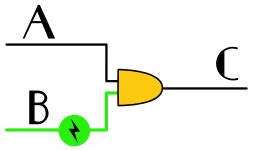
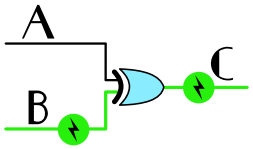
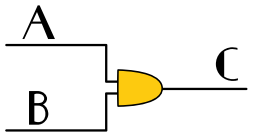
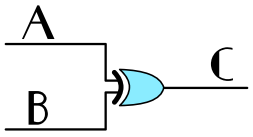
Dies ist Informatik!

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

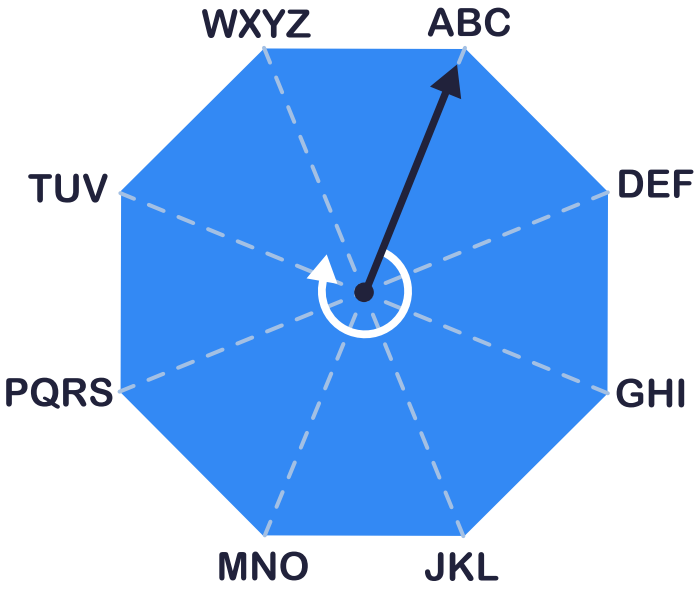
in Computerprogrammen. Manchmal muss ein Programm «Entscheideungen» darüber treffen, was als nächstes zu tun ist, je nachdem, ob eine Sache (oder manchmal auch mehrere Dinge) zuvor passiert sind.

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`

4. 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

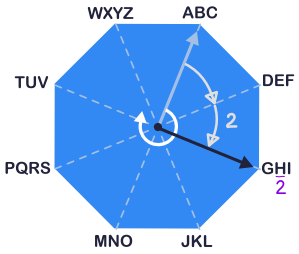
B) HAUS

C) HALLO

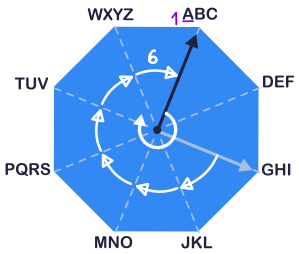
D) HALS

E) HAUT

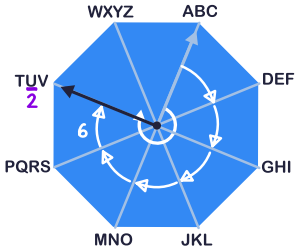
Lösung



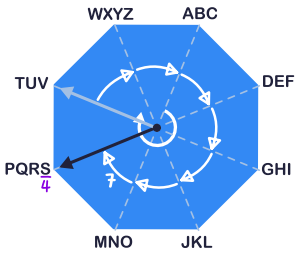
22 bedeutet, dass der Zeiger vom Block «ABC» zum Block «GHI» gedreht wird (erste Ziffer 2), und dass der zweite Buchstabe «H» genommen wird (zweite Ziffer 2).



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

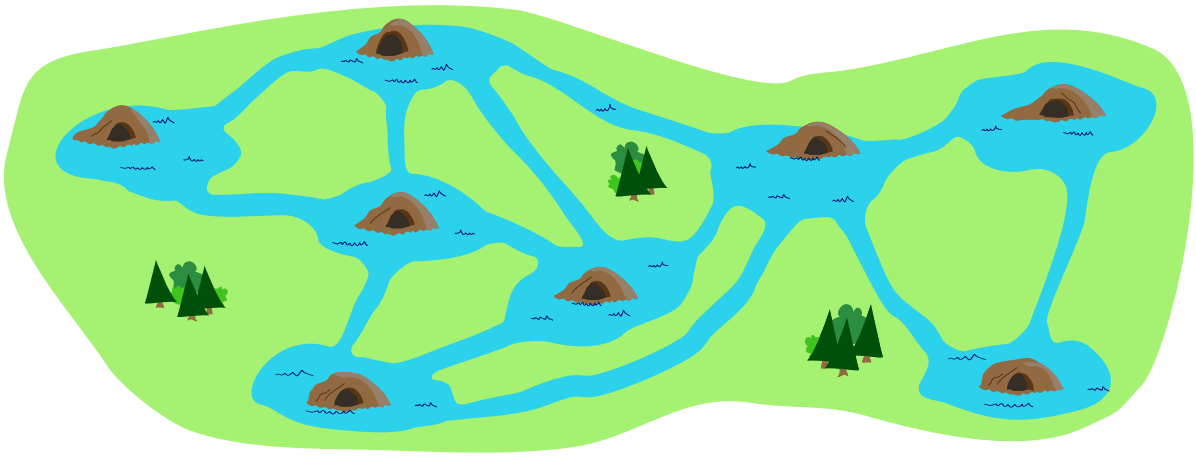
- Caesar-Code: `https://de.wikipedia.org/wiki/Caesar-Verschlüsselung`
- Polyalphabetische Substitution: `https://de.wikipedia.org/wiki/Polyalphabetische_Substitution`
- Verschlüsselungsverfahren: `https://de.wikipedia.org/wiki/Verschlüsselungsverfahren`
- Vigenére-Verfahren: `https://de.wikipedia.org/wiki/Vigenère-Chiffre`
- 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`

5. Lilis Nachbarn

Auf der Karte siehst du die Biberburgen von acht Bibern. Zwei Biber sind Nachbarn, wenn ein Kanal ihre Burgen direkt verbindet.

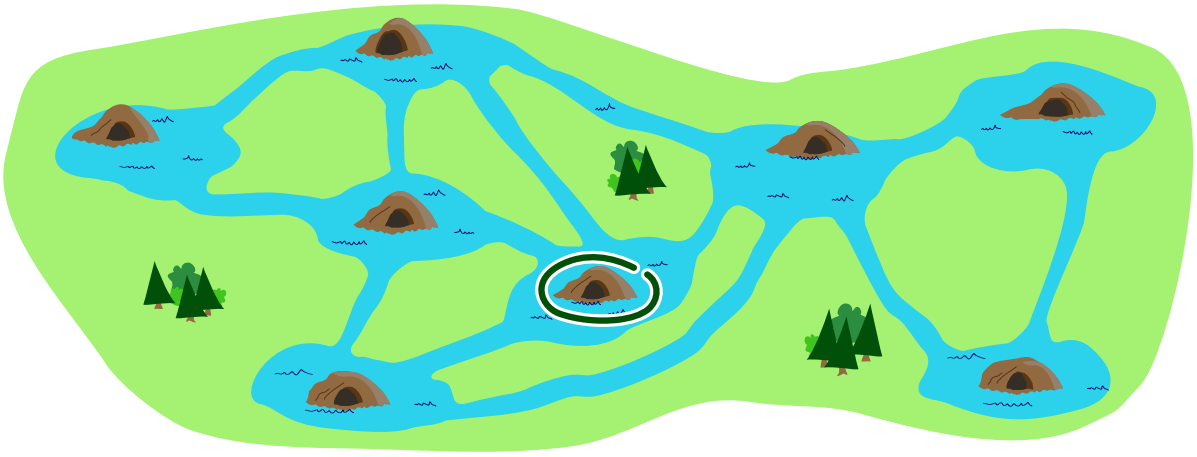
- Lili, Simon, und Peter haben je vier Nachbarn.
- Simon und Peter sind Ninas einzige Nachbarn.

In welcher Burg wohnt Lili?

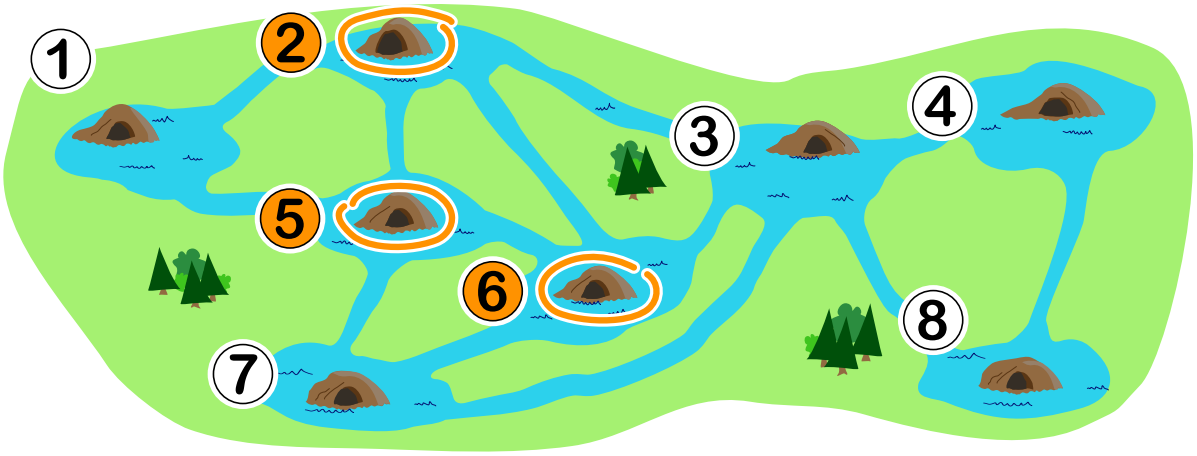


Lösung

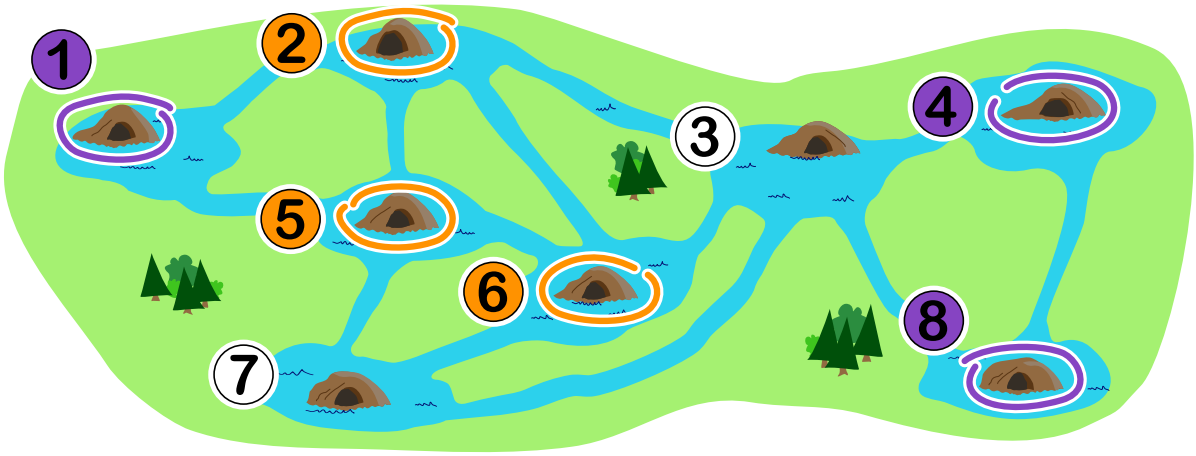
Die richtige Antwort ist:



Um das Problem zu lösen, ist es notwendig, sich auf die Kanäle zwischen den Burgen zu konzentrieren. Wir müssen die Burgen identifizieren, in denen Lili, Peter oder Simon wohnen. Da sie alle 4 Nachbarn haben, müssen von ihren Burgen je genau vier Kanäle abgehen. Es gibt drei solche Burgen: 2, 5 und 6.



Folglich leben Lili, Peter und Simon in je einer dieser drei Burgen. Nun müssen wir herausfinden, in welcher der drei Burgen Lili wohnt. Die anderen beiden Informationen beziehen sich auf Ninas Burg. Aus diesen können wir schliessen, dass von ihrer Burg genau zwei Kanäle abgehen. Also lebt Nina in einer dieser Burgen: 1, 4 oder 8.



Da wir wissen, dass Simon und Peter die beiden Nachbarn von Nina sind, können wir weiterhin folgern, dass

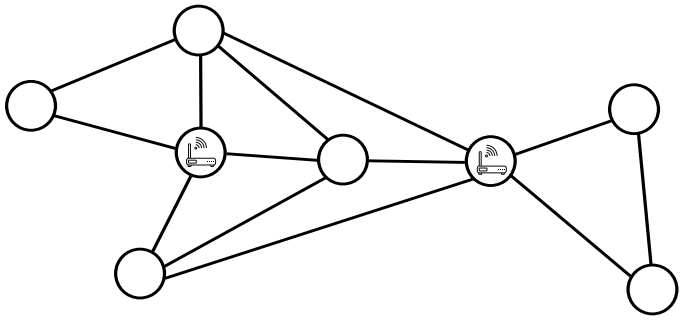
- Nina in der Burg 1 lebt
- Simon und Peter in den Burgen 5 und 7 leben (oder anders herum).

Also gibt es nur eine Burg von der vier Kanäle abgehen, die Lilis Burg sein kann. Es ist die Burg 6!

Dies ist Informatik!

In dieser Aufgabe sind jeweils zwei Biberburgen durch einen Kanal verbunden. Die Gesamtheit der Burgen und der Kanäle bildet ein Netzwerk, welches die *Beziehungen* zwischen allen Burgen aufzeigt. Ein solches Netzwerk von Beziehungen zwischen Objekten nennt man in der Informatik und der Mathematik *Graphen*. Ein Graph kann als eine *Menge* von *Knoten* betrachtet werden, die mit *Kanten* verbunden sind. In dieser Aufgabe stellen die Burgen die Knoten dar, und die Kanäle die Kanten.

Die Lehre von den Graphen nennt man *Graphentheorie*. Sie kann zur Modellierung von paarweisen Beziehungen zwischen Objekten verwendet werden. Graphen sind mathematische Modelle für netzartige Strukturen in Natur und Technik. Beispiele dafür sind soziale Strukturen, Strassennetze, Computernetze, elektrische Schaltungen, Versorgungsnetze oder chemische Moleküle. Graphen können bei der Beschreibung und Lösung von *Netzwerkproblemen* hilfreich sein, z. B. wenn es darum geht, einen guten Platz für einen Router in einem Gebäude zu finden oder sicherzustellen, dass jedes Zimmer in einem Haus ein starkes Wi-Fi-Signal hat.



Stichwörter und Webseiten

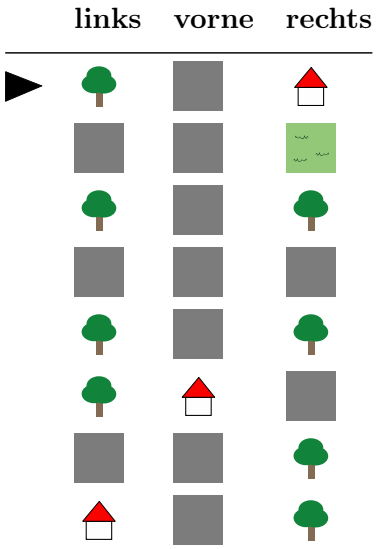
- Beziehung: `https://de.wikipedia.org/wiki/Relation_(Datenbank)`
- Graphen: `https://de.wikipedia.org/wiki/Graph_(Graphentheorie)`
- Menge: `https://de.wikipedia.org/wiki/Menge_(Mathematik)`
- Knoten: `https://de.wikipedia.org/wiki/Knoten_(Graphentheorie)`
- Kante: `https://de.wikipedia.org/wiki/Kante_(Graphentheorie)`
- Graphentheorie: `https://mathepedia.de/Graphentheorie.html`
- Netzwerkprobleme: `https://www.swisseduc.ch/informatik/theoretische_-informatik/hard_problems/docs/schwierigeprobleme_schueler.pdf`

6. Roboter Tina

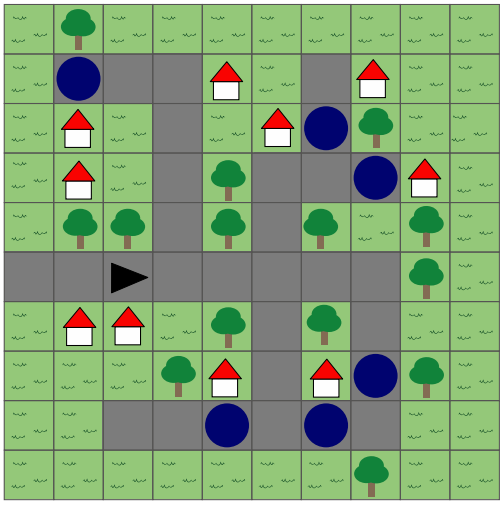
Roboter Tina liefert Post aus. Tina benutzt dazu eine Landkarte, die in Felder eingeteilt ist. Tina bewegt sich der Strasse entlang auf ein benachbartes Feld nach links, rechts oder vorne (also nicht diagonal).

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.

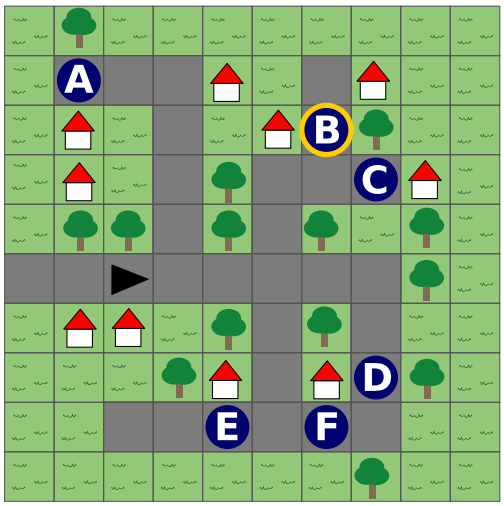


An welchem der dunkelblauen Punkte befindet sich Tina am Ende ihres Weges?



Lösung

Die korrekte Antwort ist Punkt B.

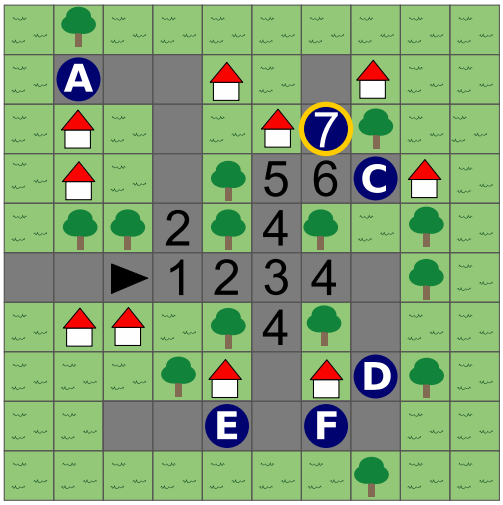


Schritt	links	vorne	rechts
▶	🌳	■	🏠
1	■	■	
2	🌳	■	🌳
3	■	■	■
4	🌳	■	🌳
5	🌳	🏠	■
6	■	■	🌳
7	■	🌳	🌳

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.

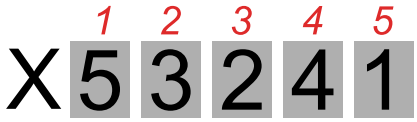
Selbstfahrende Autos sind berühmte Beispiele solcher Roboter. Sie sind mit zahlreichen Sensoren ausgestattet, die nicht nur die Geschwindigkeit oder die aktuelle Position, sondern auch der Abstand vom Strassenrand messen und Objekte auf der Strasse oder am Strassenrand und vieles, vieles mehr erfassen. 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`

7. Datenfolgen

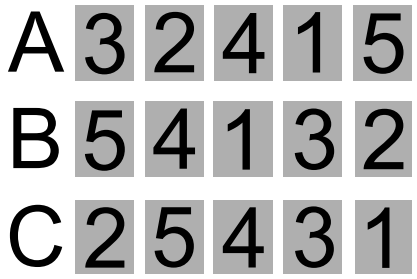
Hier siehst du eine Folge von Zahlen mit Namen X. An den Positionen 1 bis 5 in der Folge X stehen diese Zahlen: 5, 3, 2, 4, 1



Die Zahl an einer bestimmten Position beschreiben wir, indem wir Namen und Position einklammern. Ein Beispiel: Die Zahl an Position 2 von Folge X beschreiben wir so: (X 2). Aktuell ist (X 2) = 3.

Eine so beschriebene Zahl in der Folge kann selbst auch eine Position sein.
Zum Beispiel ist (X (X 2)) = (X 3) = 2.

Hier sind drei andere Folgen: A, B und C.



Welche Zahl beschreiben wir so: (A (B (C 3))) ?

A) 1

B) 2

C) 3

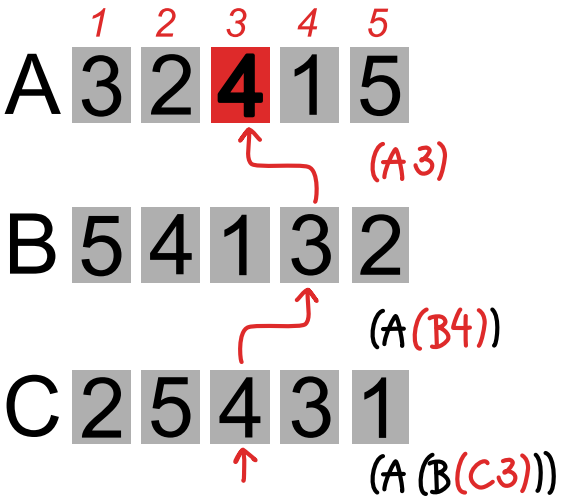
D) 4

E) 5

Lösung

Die richtige Antwort ist D) 4.

Die Beschreibung (A (B (C 3))) sagt: Die beschriebene Zahl steht in Folge A an Position (B (C 3)); die Position steht also in Folge B an Position (C3); und diese Position steht wiederum in Folge C an Position 3. Kompliziert.



Einfacher geht es, wenn wir die Beschreibung «von innen nach außen» auswerten, wie bei einem Rechenausdruck – und wie es im Beispiel der Aufgabenstellung bereits vorgemacht wird:
(A (B (C 3))) = (A (B 4)) = (A 3) = 4

Dies ist Informatik!

Es ist noch gar nicht so lange her, da hat man von *Datenverarbeitung* gesprochen, wenn es um den Einsatz von Computern ging. Zu Recht, denn Computer verarbeiten unterschiedlichste Arten von Daten, wie Zahlen, Texte, Bilder, Töne usw. Die meisten interessanten, in Computern gespeicherten Daten sind komplexer Art und haben Struktur: Die über den Tag gemessenen Temperaturen an einer Wetterstation zum Beispiel kann man als Folge von Paaren speichern, die jeweils aus der Uhrzeit der Messung und der gemessenen Temperatur bestehen. Hier gibt es also eine Paar- und eine Reihenfolge-Struktur.

Daten können unterschiedlichste Strukturen haben, und so haben Informatikerinnen und Informatiker unterschiedlichste so genannte *Datenstrukturen* entwickelt, um Daten geschickt zu speichern und (genau so wichtig) effizient auf die Daten zugreifen zu können. Eine einfache Datenstruktur ist das *Array* (auf Deutsch auch: Feld), das in dieser Biberaufgabe die Hauptrolle spielt. Ein Array kann nämlich eine feste Anzahl an Daten (also auch Zahlen) an aufeinanderfolgenden Positionen speichern. Durch die Positionen haben die Daten im Array eine Reihenfolge-Struktur – ein Array wäre also gut für die oben genannten Uhrzeit/Temperatur-Paare geeignet. Wegen ihrer festen Größe gehören Arrays in der Informatik zu den *statischen* Datenstrukturen. Für Datenfolgen gibt es auch *dynamische* Datenstrukturen wie z.B. Listen, deren Größe sich je nach Bedarf ändern kann.

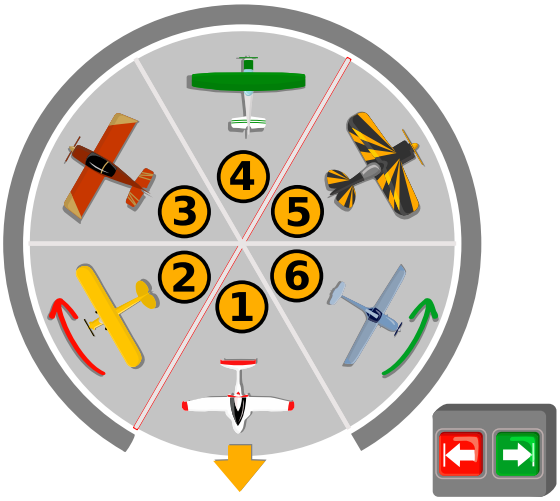
Ob statisch oder dynamisch: Wenn eine Folgen-Datenstruktur Zahlen enthält, können diese Zahlen auch Positionen in der gleichen oder einer anderen Folge angeben. Das wird in der Informatik häufig für die sogenannte indirekte Adressierung benutzt: Die Adresse bzw. Position in einer Folge wird nicht direkt als Zahl, sondern indirekt durch einen (Zahlen-)Wert aus einer Folge angegeben, der auch selbst wieder indirekt adressiert werden kann – usw.

Stichwörter und Webseiten

- Datenverarbeitung: `https://de.wikipedia.org/wiki/Datenverarbeitung`
- Datenstrukturen: `https://de.wikipedia.org/wiki/Datenstruktur`
- Array: `https://de.wikipedia.org/wiki/Feld_(Datentyp)`
- Adressierung: `https://de.wikipedia.org/wiki/Adressierung_(Rechnerarchitektur)`

8. Rundhangar

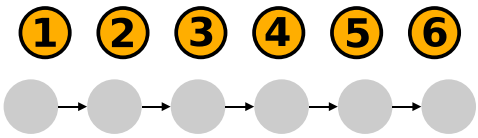
Auf dem Flugplatz von Beavertown parken sechs Flugzeuge in einem Hangar. Sie stehen auf einer Drehscheibe, in sechs Parkpositionen. Aussen gibt es zwei Pfeiltasten ⬅️ ➡️. Mit einem Tastendruck kann man die Drehscheibe um genau eine Parkposition nach links oder rechts drehen.



Morgens, wenn die Piloten ihre Flugzeuge abholen, ist die Parkposition 1 immer beim Hangartor und das Flugzeug darauf kann herausrollen. Im besten Fall müssen die Pfeiltasten dann noch fünfmal gedrückt werden, damit auch alle weiteren Flugzeuge herausrollen können. Wenn beispielsweise die Piloten in der Reihenfolge 1, 6, 5, 4, 3, 2 auf die Parkpositionen zugreifen wollen, genügt es, die Taste ➡️ fünfmal zu drücken.

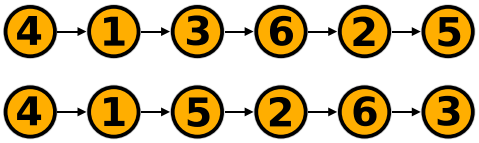
Aber was ist der schlechteste Fall? Bei welcher Reihenfolge müssen die Tasten am häufigsten gedrückt werden?

Gib ein Beispiel für eine solche Reihenfolge.



Lösung

Es gibt zwei richtige Antworten:



Zum Finden der Lösung wird immer das Flugzeug ausgewählt, welches auf der Parkposition ist, die die grösste Entfernung zum Hangartor hat.

«➡ ➡ ➡ 4» bedeutet, dass nach drei Tastendrücken das Flugzeug an Position 4 ausparkt

4 1 3 6 2 5:

4 1 5 2 6 3:

In beiden Fällen werden insgesamt 16 Schritte benötigt.

Es können auch nicht mehr als 16 Schritte sein, weil nur ganz am Anfang zwei Dreierschritte unmittelbar hintereinander folgen können. Danach können sich höchstens Zweier- und Dreierschritte abwechseln.

Dies ist Informatik!

Der Rundhangar hat den Vorteil, dass Flugzeuge sehr platzsparend geparkt werden können. Das Abholen dauert aber in der Regel länger als bei einem gewöhnlichen Hangar.

Die *Effizienz* eines Verfahrens ist ein sehr zentrales Thema in der Informatik, weil sie ein wichtiges Beurteilungskriterium für *Algorithmen* ist. Sehr oft geht es bei Effizienz um die Zeitdauer der Durchführung (*Laufzeiteffizienz*), aber das ist nicht immer der Fall. In der allgemeinen Definition der Effizienz von Algorithmen geht es um alle benötigten Ressourcen also zum Beispiel auch um die Grösse des benötigten Speichers (*Speichereffizienz*).

Wie in unserem konkreten Hangar-Beispiel führen Einsparungen bei einer Ressource zu einem höheren Bedarf einer anderen Ressource. Es hängt vom konkreten Zusammenhang und von der Verfügbarkeit der Ressourcen ab, welcher Ressource eine grössere Bedeutung beigemessen wird.

Beispielsweise sind *Bubblesort* und *Timsort* beide Algorithmen, um eine Liste von Elementen zu sortieren. Bubblesort sortiert die Liste zeitlich proportional zur Anzahl der Elemente im Quadrat ($\mathcal{O}(n^2)$), erfordert aber nur wenig zusätzlichen Speicher, der in Bezug auf die Länge der Liste konstant ist.

Timsort sortiert wesentlich schneller als Bubblesort ($\mathcal{O}(n \log n)$), hat aber einen mit der Grösse der Liste linear wachsenden Platzbedarf. Wenn für eine bestimmte Anwendung grosse Listen mit hoher Geschwindigkeit sortiert werden müssen, ist Timsort die bessere Wahl; Wenn es jedoch wichtiger ist, den Speicherbedarf der Sortierung zu minimieren, ist Bubblesort die bessere Wahl.

Stichwörter und Webseiten

- Effizienz (Laufzeiteffizienz und Speichereffizienz): `https://de.wikipedia.org/wiki/Effizienz_(Informatik)`
- Algorithmus: `https://de.wikipedia.org/wiki/Algorithmus`
- Bubblesort: `https://de.wikipedia.org/wiki/Bubblesort`
- Timsort: `https://de.wikipedia.org/wiki/Timsort`
- O-Notation: `https://de.wikipedia.org/wiki/Landau-Symbole`

9. Filmabend

Ein paar Freunde möchten einen Film miteinander anschauen. Zur Auswahl stehen sieben Filme. Um eine Entscheidung zu fällen, bewertet jede Person jeden Film gut, mittel oder schlecht.

Das Ergebnis siehst du unten. Leider gibt es keinen Favoriten für den Filmabend.

Ein Film ist ein «Favorit», wenn jede Person diesem Film die eigene beste Bewertung gegeben hat. Film 1 zum Beispiel ist kein Favorit, weil Niklaus seine beste Bewertung einem anderen Film gegeben hat, nämlich Film 4.

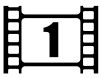
Ada möchte nun so wenig Freunde wie möglich überzeugen, ihre Bewertung zu ändern, damit es doch einen Favoriten gibt.

Hilf Ada und ändere so wenige Bewertungen wie möglich, so dass es einen Favoriten gibt.

	1	2	3	4	5	6	7
Ada	😊	😊	😊	😊	😊	😊	😊
Nancy	😐	😊	😊	😐	😐	😊	😊
Niklaus	☹	☹	☹	😐	☹	☹	☹
Grace	☹	😐	😐	😐	☹	😐	☹
Edsger	😊	😐	☹	☹	😐	😊	😊
Rozsa	😐	☹	😐	☹	😊	😐	😐

Lösung

Zu Beginn gibt es keinen Favoriten. Für jeden Film finden wir Freunde, die andere Filme besser bewerten.

Film	Freunde, die andere Filme besser bewerten
	4: Nancy, Niklaus, Grace und Rozsa
	3: Niklaus, Edsger und Rozsa
	3: Niklaus, Edsger und Rozsa
	3: Nancy, Edsger und Rozsa
	3: Nancy, Grace und Edsger
	2: Niklaus und Rozsa
	3: Niklaus, Grace und Rozsa

Bei Film 6 gibt es nur zwei Freunde, die andere Filme besser bewerten. Bei allen anderen Filmen sind es sogar mehr als zwei. Wenn nur ein Freund eine Bewertung ändert, wird es danach immer noch keinen Favoriten geben. Also muss Ada mindestens zwei Freunde überzeugen ihre Bewertungen zu ändern. Wenn Niklaus und Rozsa je eine Bewertung so verändern, dass Film 6 ihre beste Bewertung erhält, wird Film 6 ein Favorit.

Welche Bewertung können Niklaus und Rozsa jeweils ändern, damit Film 6 ihre beste Bewertung erhält? Die beiden haben jeweils drei Möglichkeiten:

- Niklaus kann seine Bewertung von Film 6 verbessern (zu 😐 oder 🙂) oder seine Bewertung von Film 4 verschlechtern (zu ☹). In allen drei Fällen erhält Film 6 danach seine beste Bewertung.
- Rozsa kann ihre Bewertung von Film 6 verbessern (zu 🙂) oder ihre Bewertung von Film 5 verschlechtern (zu 😐 oder ☹). In allen drei Fällen erhält Film 6 danach ihre beste Bewertung.

Diese jeweils drei Möglichkeiten können beliebig miteinander kombiniert werden. Insgesamt gibt es also $3 \times 3 = 9$ Möglichkeiten, nur zwei Bewertungen so zu ändern, dass es einen Favoriten gibt.

Dies ist Informatik!

Wie gehen wir vor, um diese Aufgabe zu lösen? Eine Idee besteht darin, für jeden Film und jede Person einzeln zu prüfen, ob diese Person andere Filme besser bewertet hat oder nicht. In unserem Fall entsteht dabei die obige Tabelle. Diese Tabelle hilft uns herauszufinden, welche Personen ihre Bewertungen ändern müssen, damit wir tatsächlich mit der kleinstmöglichen Anzahl von Veränderungen zu einem Favoriten gelangen.

Ada kann tatsächlich diesen *Algorithmus* verwenden, um ihr Problem zu lösen.

Ist dieser Algorithmus jedoch *effizient*? Könnte Ada noch schneller sein?

Wir bezeichnen im Folgenden die Anzahl Filme mit M und die Anzahl Freunde mit F. Ada muss alle $M \times F$ Einträge einzeln betrachten und für jeden Eintrag muss sie alle anderen $M - 1$ Bewertungen derselben Person berücksichtigen. Insgesamt muss Ada $M \times (M - 1) \times F$ Bewertungen betrachten.

Um zu entdecken, ob eine der Bewertungen problematisch ist, muss Ada diese Bewertung nur mit der besten Bewertung vergleichen, die diese Person vergeben hat. Wenn diese Person einen anderen Film besser findet, dann kann der von Ada gerade betrachtete Film gar kein Favorit sein.

Anders gesagt, wenn Ada zunächst die besten Gesamtbewertungen für jede einzelne Person herausfindet (indem sie sich alle $M \times F$ Bewertungen ansieht), kann sie für alle $M \times F$ Bewertungen feststellen, ob sie schlechter ausfallen als die beste Bewertung der jeweiligen Person.

Alles in allem führt dieser alternative Algorithmus mit einer gezielten Vorberechnung der besten Bewertungen dazu, dass Anna sich $2 \times M \times F$ Bewertungen ansieht. Bei $\mathrm{M} = 7$ und $\mathrm{F} = 6$ sind das 84 Tabellenzugriffe, während der erste Algorithmus 252 Tabellenzugriffe erfordert. Der zweite Algorithmus löst das Problem von Ada ebenfalls korrekt, ist aber effizienter als der erste Algorithmus.

Eine der wichtigsten Aufgaben in der Informatik besteht darin, Probleme nicht nur korrekt, sondern auch so effizient wie möglich zu lösen. Mit schnelleren Computern werden Lösungen schneller berechnet. Sind dennoch keine effiziente Algorithmen bekannt, um ein Problem zu lösen, können auch schnellere Computer an ihre Grenzen kommen.

Stichwörter und Webseiten

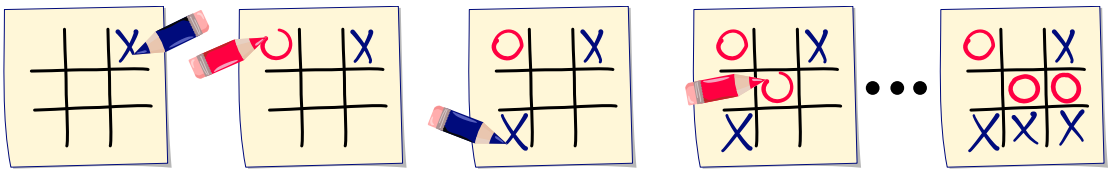
- Algorithmus: `https://de.wikipedia.org/wiki/Algorithmus`
- Effizienz: `https://de.wikipedia.org/wiki/Effizienz_(Informatik)`

10. Tic-Tac-Toe Endstand

Tic-Tac-Toe ist ein Spiel für zwei Personen.

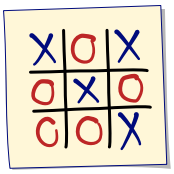
In einem Raster mit 3×3 Feldern füllen die beiden Spieler abwechselnd je ein Zeichen in ein freies Feld: Der eine Spieler X, der andere O. Wer als erster drei Felder in einer Zeile, Spalte oder Diagonale mit seinem Zeichen ausfüllen kann, gewinnt, und das Spiel ist beendet. Wenn alle Felder ausgefüllt sind und niemand gewonnen hat, endet das Spiel unentschieden.

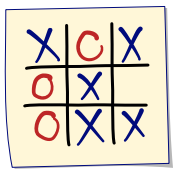
Hier siehst du die Spielstände eines möglichen Spielverlaufs: Die ersten 4 Spielzüge, sowie den letzten Zug. Der Spieler mit X gewinnt.

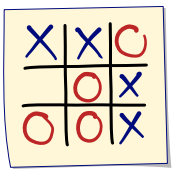


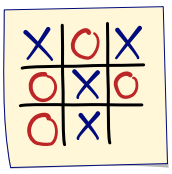
Den Spielstand am Ende eines Spiels nennen wir Endstand. Die Spielregeln legen genau fest, wie die Felder mit X und O ausgefüllt werden können und wann das Spiel endet.

Nur eines der vier Bilder zeigt einen Endstand von Tic-Tac-Toe. Welches?

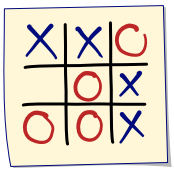
A) 

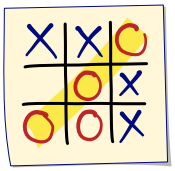
B) 

C) 

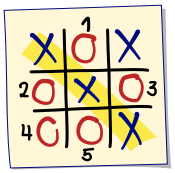
D) 

Lösung

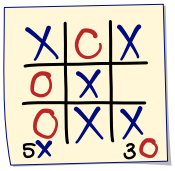
Antwort C ist richtig: 



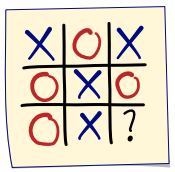
Antwort C ist korrekt, weil ein Spieler gewonnen hatte (drei ○ in einer Diagonalen) und dann keine weiteren Züge mehr erfolgten.



Antwort A ist nicht korrekt. Spieler ✕ hat das Spiel gewonnen, aber Spieler ○ hat weitere Felder ausgefüllt. Da der Gewinner immer das letzte Feld ausfüllt, kann er niemals weniger Zeichen als der Verlierer haben.



Antwort B ist nicht korrekt, weil 5 Felder mit ✕ aber nur 3 Felder mit ○ ausgefüllt sind. Das ist nicht möglich; denn die Anzahl der ○- und die Anzahl der ✕-Zeichen können sich höchstens um 1 unterscheiden.



Antwort D ist nicht korrekt; denn es zeigt keinen Endstand. Es gibt noch keinen Gewinner und die Felder sind nicht vollständig gefüllt.

Dies ist Informatik!

Bei der Lösung der Aufgabe haben wir geprüft, ob die vier Bilder der Antwortmöglichkeiten eine gültige Endstellung dokumentieren. Von den Tic-Tac-Toe-Spieltregeln kann man neue Regeln über gültige Endstellungen ableiten, zum Beispiel diese:

1. Die Differenz zwischen der Anzahl von ✕ und der Anzahl von ○ muss 0, –1 oder 1 sein.
2. Wenn kein Spieler gewonnen hat, müssen alle Felder ausgefüllt sein.
3. Der Verlierer kann höchstens so viele Felder ausfüllen wie der Gewinner.
4. Im Dokument eines beendeten Spiels kann höchstens eine Folge von drei gleichen Zeichen sein.

Diese neuen Regeln sind keine Spielregeln, sondern dienen nur der Überprüfung, ob das ausgefüllte Raster ein Endstand ist. Wenn ein Bild in Konflikt mit einer dieser Regeln steht, kann es kein Endstand sein.

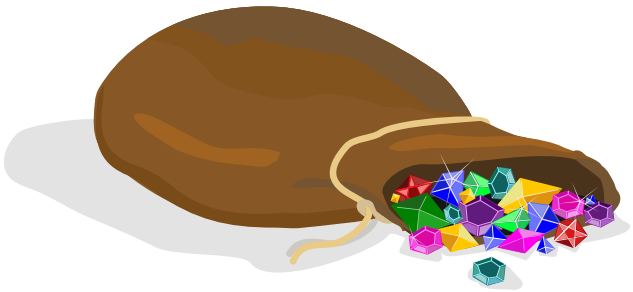
Regeln sind sehr wichtig in der Computertechnik. Ein Interpreter, der ein Programm ausführt, überprüft, ob der eingegebene Text den Syntaxregeln der Programmiersprache entspricht.

In der Programmierung werden in sogenannten Zusicherungen Regeln verwendet, um während eines Programmlaufs die Korrektheit eines Programms zu testen.

Stichwörter und Webseiten

- Tic-Tac-Toe: `https://de.wikipedia.org/wiki/Tic-Tac-Toe`
- Interpreter: `https://de.wikipedia.org/wiki/Interpreter`
- Syntax: `https://verify.rwth-aachen.de/programmierung/folien/I2_Grundlagen.pdf`
- Programmiersprache: `https://de.wikipedia.org/wiki/Programmiersprache`

11. 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

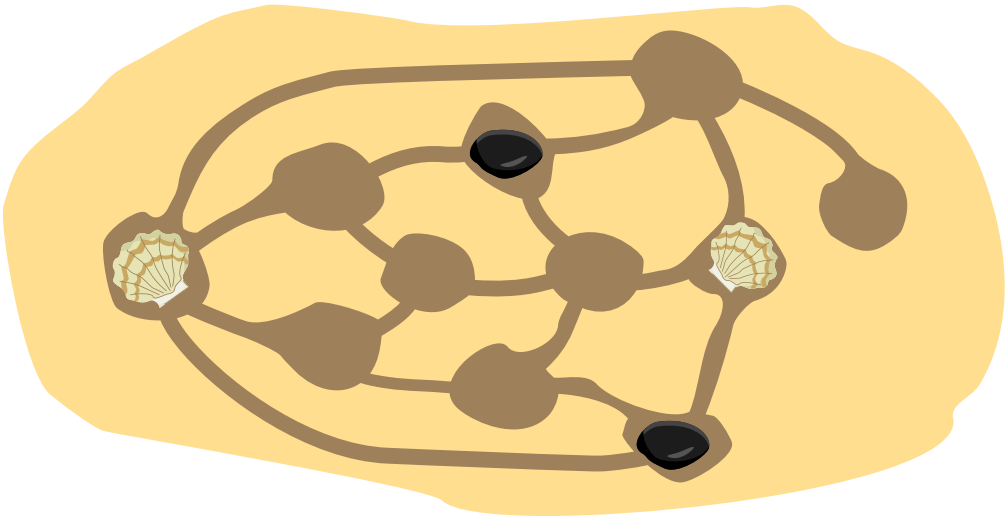
- Algorithmus: `https://de.wikipedia.org/wiki/Algorithmus`
- Zeitkomplexität: `https://de.wikipedia.org/wiki/Zeitkomplexität`

12. Muscheln und Steine

Ann und Bob spielen am Strand. Sie graben einige Mulden und verbinden manche davon mit Furchen, die sie in den Sand ziehen. Anns Spielfiguren sind Muscheln . Bobs Spielfiguren sind Kieselsteine .

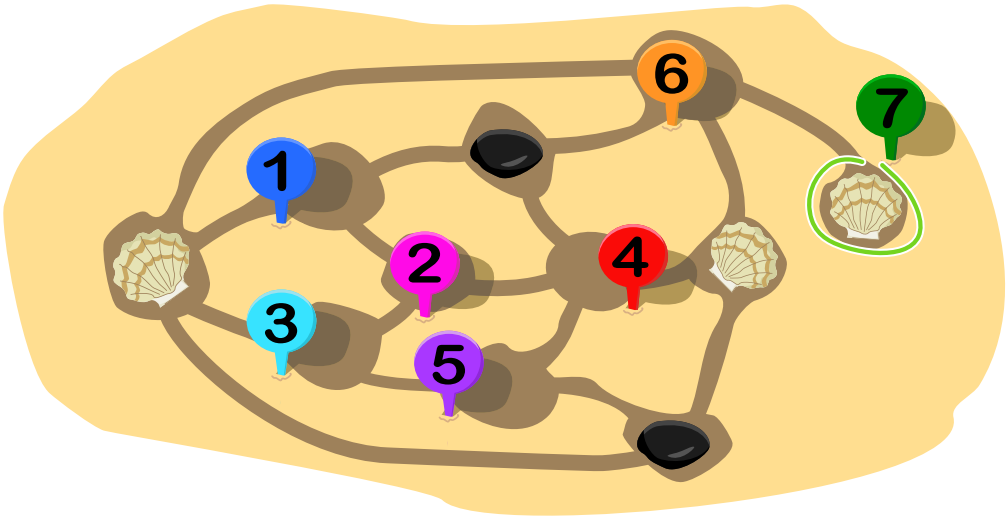
Abwechselnd setzen sie eine ihrer Spielfiguren in eine freie Mulde. Verloren hat, wer als erstes zwei eigene Figuren in zwei direkt verbundene Mulden gesetzt hat. Im Bild siehst du den Spielstand nach einigen Zügen.

Ann ist an der Reihe. In welche der freien Mulden muss sie ihre nächste Muschel setzen, um sich den Sieg zu sichern?

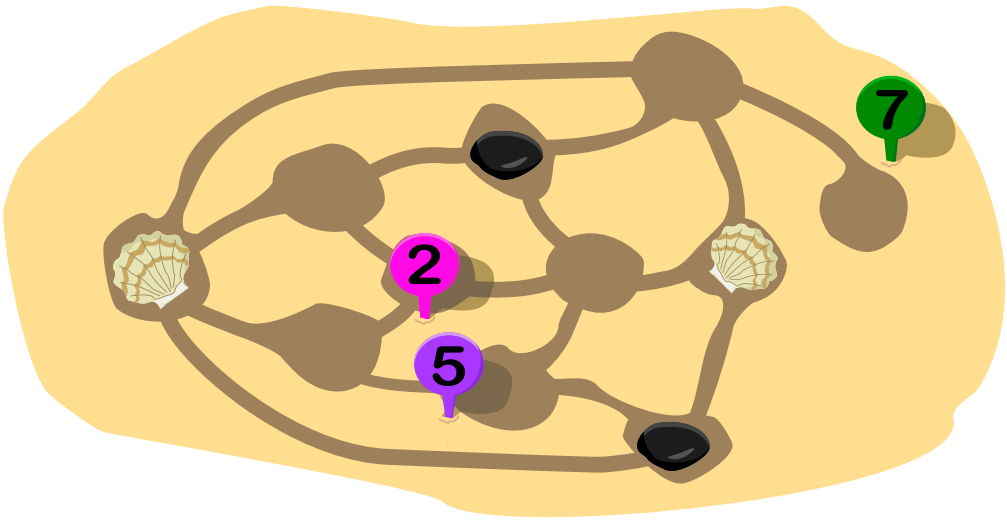


Lösung

Die richtige Antwort ist die Mulde 7.



Ann ist an der Reihe. Für sie kommen die Mulden 1, 3, 4 und 6 nicht in Frage, es bleiben also 2, 5 und 7.



Sie sieht, dass für Bob die Mulden 1, 4, 5 und 6 nicht in Frage kommen. Für ihn bleiben also 2, 3 und 7.

Wenn Ann die 7 spielt, kann Bob entweder 2 oder 3 spielen; in beiden Fällen kann Ann noch die 5 spielen und Bob verliert.

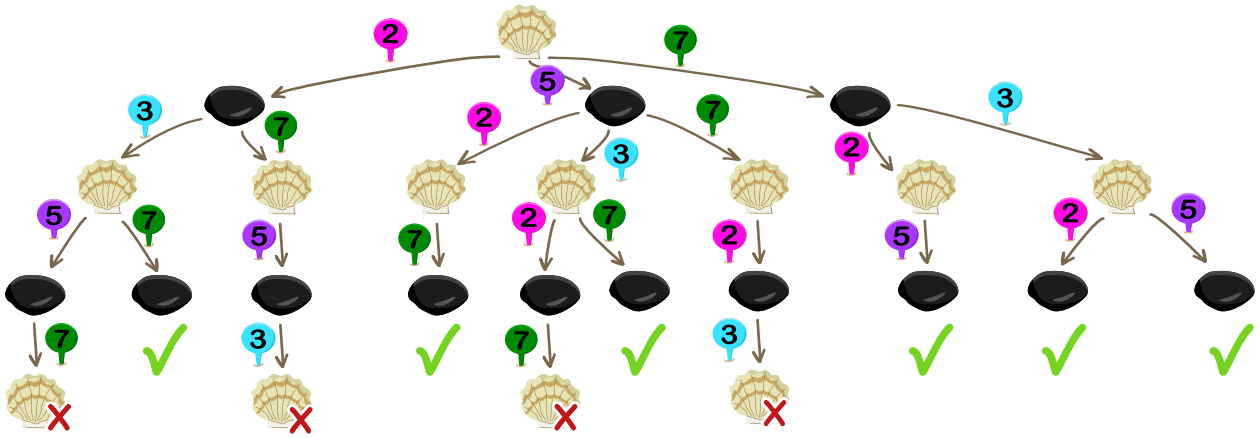
Wenn Ann beim Spielstand im Bild die 2 spielen würde, könnte Bob als nächstes die 7 spielen. Danach müsste Ann die 5 spielen, Bob die 3 und dann hätte Ann verloren.

Würde Ann die 5 spielen, könnte Bob die 7 spielen, Ann müsste die 2 spielen, Bob die 3 und wieder hätte Ann verloren.

Übrigens könnte Bob auch nicht gewinnen, wenn er beim Spielstand im Bild an der Reihe wäre …

Dies ist Informatik!

Um die möglichen Spielzüge von Ann und Bob systematisch darzustellen, bietet sich ein sogenannter Spielbaum an:



In diesem Spielbaum lässt sich ablesen, mit welchem Zug Ann sich den Sieg sichern kann: im rechten Ast, der damit beginnt, dass Ann die 7 spielt, sind nur Situationen erreichbar, in denen sie gewinnt. In der sogenannten *Spieltheorie*, einem Spezialgebiet der Mathematik, werden Aussagen über den Ausgang von Spielen betrachtet, bei denen zwei oder mehr Spieler interagieren. Die Informatik beschäftigt sich mit Algorithmen zur Auswertung solcher Spielbäume. Computer mit ausreichender Rechenleistung können in Spielen wie Schach bereits gegen Menschen antreten und gewinnen. Die Spieltheorie bietet aber auch für die Psychologie, die Wirtschaftswissenschaften und andere Fächer Modelle für komplexe Systeme, in denen «Player» interagieren, etwa für das Kaufverhalten von Kunden bei Preisänderungen oder für die Routenwahl im Strassenverkehr.

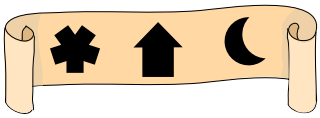
Bei dem Spiel von Ann und Bob handelt es sich um eine Instanz von «COL». Das ist ein Spiel für zwei Spieler, das von Colin Vout eingeführt wurde und im bekannten Buch «On Numbers and Games» des Mathematikers John Horton Conway eine Rolle spielt.

Stichwörter und Webseiten

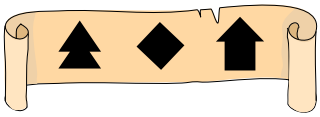
- Spieltheorie: `https://de.wikipedia.org/wiki/Spieltheorie`
- John Horton Conway: `https://de.wikipedia.org/wiki/John_Horton_Conway`

13. Maria auf Schatzsuche

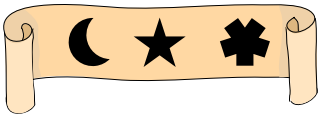
Maria findet eine geheimnisvolle Kiste. Leider ist die Kiste verschlossen. Um sie zu öffnen, muss Maria den «Schlüssel» herausfinden: die richtige Kombination aus drei Symbolen. Zum Glück findet sie neben der Kiste auch diese Hinweise zu einigen falschen Kombinationen:



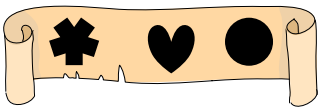
1) Eines der Symbole ist Teil des Schlüssels und an der richtigen Position.



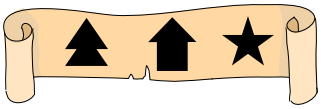
2) Keines der Symbole ist Teil des Schlüssels.



3) Zwei Symbole sind Teil des Schlüssels. Beide sind aber an der falschen Position.



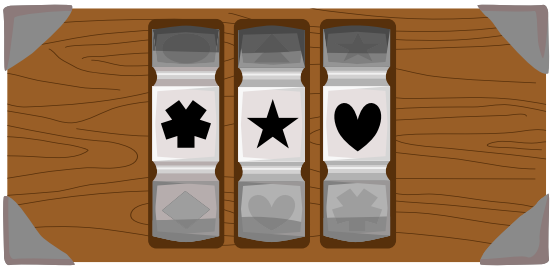
4) Ein Symbol ist Teil des Schlüssels. Dieses ist aber an der falschen Position.



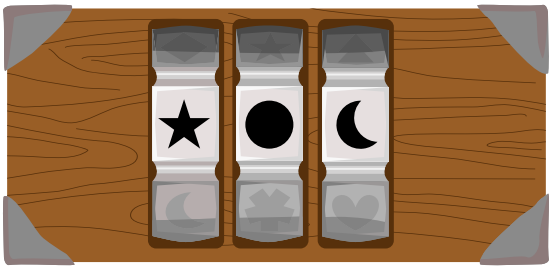
5) Ein Symbol ist Teil des Schlüssels. Dieses ist aber an der falschen Position.

Eine der folgenden Kombinationen ist der Schlüssel für die Kiste. Welche?

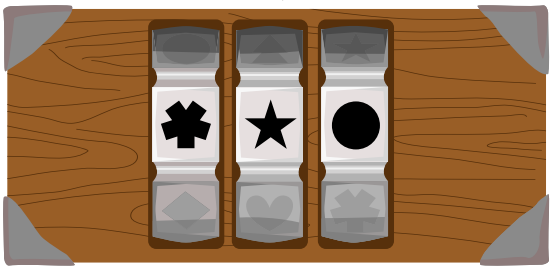
A)



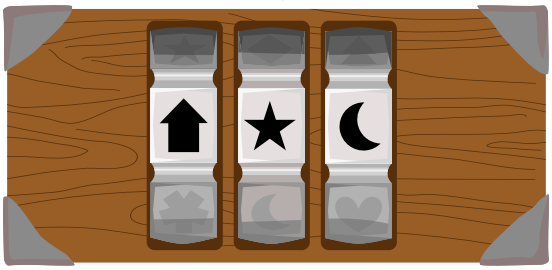
B)



C)



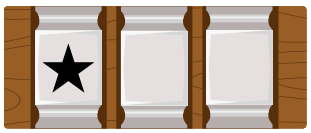
D)



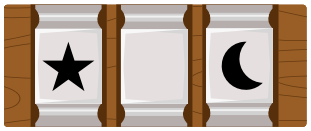
Lösung

Die richtige Antwort ist B).

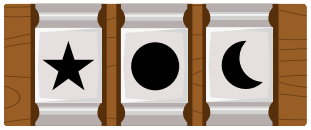
Wir beginnen damit, die Symbole, welche im Schlüssel vorkommen können herauszufinden. Nach Hinweis 2) können wir die Symbole, welche nicht Teil des Schlüssels sein können, entfernen: die Tanne ▲, den Diamanten ◆ und das Haus ⬟. Hinweis 5) gibt an, dass ein Symbol zwar Teil des Schlüssels ist, aber sich an der falschen Position befindet. Da die Tanne ▲ und das Haus ⬟ im Schlüssel nicht vorkommen können, wissen wir, dass der Stern ★ Teil des Schlüssels ist, aber sich an der falschen Position befindet. Hinweis 3) schliesst aus, dass der Stern ★ in der Mitte sein kann. Somit kennen wir die endgültige Position des Sternes:



Da es nur eine Antwortmöglichkeit gibt, welche mit dem Stern beginnt, haben wir den Schlüssel bereits gefunden. Um uns zu überzeugen, suchen wir weiter nach den beiden fehlenden Symbolen. Hinweis 1) zeigt, dass ein Symbol im Schlüssel vorkommt und sich bereits an der richtigen Position befindet. Das Haus ⬟ und die erste Position konnte bereits ausgeschlossen werden. Daher wissen wir, dass sich der Mond an der richtigen Position befindet. Daraus ergibt sich das folgende Bild:



Hinweis 4) zeigt, dass ein Symbol zwar Teil des Schlüssels ist, aber sich an der falschen Position befindet. Das Symbol ✤ können wir ausschliessen. Ausserdem ist nur der mittlere Platz noch frei. Daher kann auch das Herz ♥ nicht Teil des Schlüssels sein. Daraus folgt, dass der Kreis ⬟ die Position in der Mitte einnimmt.



Das richtige Ergebnis kann auch anders ermittelt werden. Jede Möglichkeit führt jedoch zum gleichen Ergebnis.

Dies ist Informatik!

Diese Aufgabe kann logisch gelöst werden zum Beispiel mit Hilfe des «Ausschlussverfahrens». In unserem Fall, haben wir mit Hinweis 2) begonnen und drei Symbole ausgeschlossen, was uns schnell zum Schlüssel führte. Die Prioritätensetzung auf Hinweis 2) könnte man als mentale Strategien, Regel oder Abkürzungen betrachten, die uns halfen, mit begrenztem Wissen und Zeit eine Entscheidung zu treffen. In der Informatik werden solche Regeln als *Heuristiken* bezeichnet, die auch programmiert und automatisiert werden können.

Jeden Tag treffen wir viele kleine Entscheidungen aufgrund von Hinweisen oder müssen verschiedene *(Neben-)Bedingungen* eines Problems verstehen, um es zu lösen. In dieser Aufgabe sind wir den Hinweisen gefolgt und haben das Problem Schritt für Schritt gelöst, um die Kiste zu öffnen.

Wie würde ein Computer dieses Problem lösen? Es gibt insgesamt 336 Möglichkeiten, wie diese acht Symbole an drei Positionen angeordnet werden können. Ein Computer würde sie alle ausprobieren. In der Informatik nennt man dies eine *vollständige Suche*. Die vollständige Suche (auch *Brute-Force* oder *rekursives Backtracking* genannt) ist eine Methode zur Lösung eines Problems, bei dieser der gesamte Suchraum durchlaufen wird. Für uns mag diese Lösung sehr *ineffizient* erscheinen, da wir viel Zeit benötigen würden, um alle Möglichkeiten auszuprobieren (und vergessen würden, was wir bereits ausprobiert haben). Ein Computer kann solche Aufgaben jedoch sehr schnell und damit effizient lösen. Die Symbole im Beispiel könnten auch für ein Passwort stehen. Ein Passwort sollte auch immer so gewählt werden, dass es möglichst viele verschiedene Zeichen enthält, damit auch eine vollständige Suche nicht in angemessener Zeit den Schlüssel ergibt.

Mit Hinweis 2) zu beginnen, daher die Lösungsmöglichkeiten zu minimieren, wird in der Informatik als *Backtracking* bezeichnet. An jedem *Knoten* eines *Baumes* werden die Möglichkeiten, die offensichtlich nicht im Schlüssel vorkommen können, eliminiert. Auf diese Weise werden in jeder Tiefe eines Baumes die Möglichkeiten reduziert.

Stichwörter und Webseiten

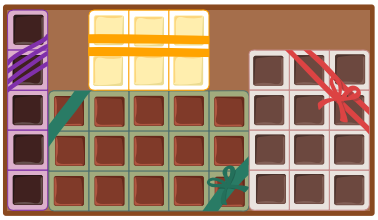
- Heuritstik: `https://de.wikipedia.org/wiki/Heuristik#Informatik`
- Bedingung: `https://studyflix.de/informatik/bedingungen-und-operatoren-684`
- Brute-Force (vollständige Suche): `https://de-academic.com/dic.nsf/dewiki/204529`
- Effizenz: `https://de.wikipedia.org/wiki/Effizienz_(Informatik)`
- Backtracking: `https://de-academic.com/dic.nsf/dewiki/204529`
- Knoten: `https://de.wikipedia.org/wiki/Knoten_(Graphentheorie)`
- Baum: `https://de.wikipedia.org/wiki/Baum_(Graphentheorie)`

14. Pralinés einpacken

Die Schokoladenfabrik «Castocolat» versendet für eine Werbekampagne an jeden ihrer Kunden vier Schachteln mit Pralinés.

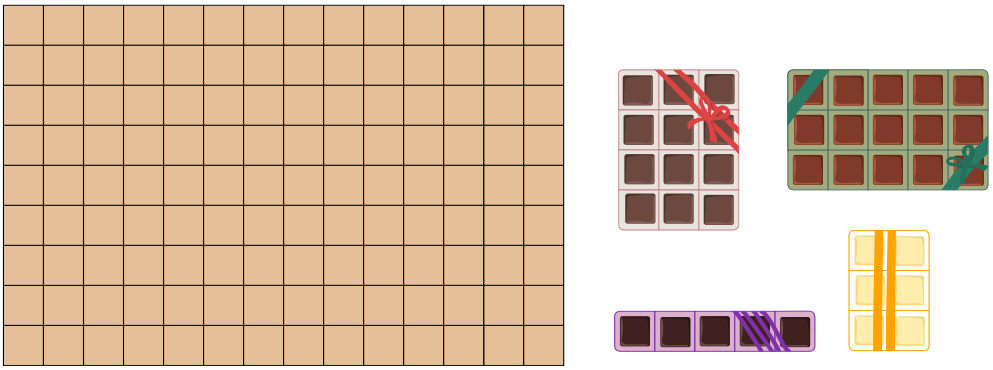
Um Porto und Material zu sparen, soll Linus die vier verschiedenen Schachteln nebeneinander in einen möglichst kleinen Karton legen. Die Schachteln dürfen nicht übereinander gestapelt werden, da die Pralinés sonst zerdrückt werden.

Linus hat die Praliné-Schachteln so in einen Karton für $5 \times 9 = 45$ einzelne Pralinés gelegt.



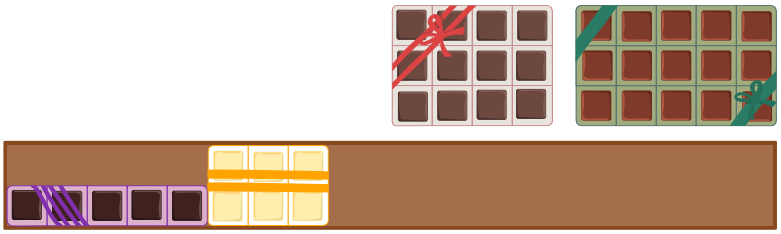
Lina behauptet: «Wenn du die Schachteln anders legst, passen sie in einen kleineren Karton.»

Lege die Schachteln so, dass sie in einen möglichst kleinen Karton passen.



Lösung

Insgesamt sind es $12 + 15 + 6 + 5 = 38$ Pralinés, die Linus in einen Karton legen muss. Ein Karton, in den 38 einzelne Pralinés ohne Leerraum gelegt werden können, müsste entweder die Grösse 1×38 oder 2×19 haben (2 und 19 sind die einzigen Teiler von 38). Die beiden Praliné-Schachteln mit den Grössen 3×4 und 3×5 würden in keinen dieser Kartons passen.



Wählt Linus einen Karton für 39 Pralinés (also mit Leerraum für genau eine weitere Praline) hat dieser entweder die Grösse 1×39 oder 3×13. Die Schachteln 3×5, 3×4, 3×2 passen in den Karton, die Schachtel 1×5 passt aber nicht in den verbleibenden freien Raum der Grösse 2×3.

Ein Karton für 40 Pralinés kann folgende Grössen haben 1×40, 2×20, 4×10, 5×8. In die Kartons mit den Grössen 1×40 oder 2×20 passen nicht alle Schachteln. In die beiden anderen Kartons passen alle vier Schachteln, z.B. so:

Man kann die Schachteln noch zu einigen anderen Anordnungen legen, die in einen Karton für 40 Pralinés passen. Platzsparender als mit Leerraum für 2 Pralinen können diese vier Praliné-Schachteln also nicht gepackt werden.

Dies ist Informatik!

In dieser Biberaufgabe sollen Rechtecke so angeordnet werden, dass das umschliessende Rechteck die minimale Fläche hat. Dieses Problem ist in der Informatik auch als «rectangle packing» bekannt als eines von vielen so genannten Verpackungsproblemen. Für wenige Rechtecke können wir relativ einfach die *optimale* Lösung finden (hier den kleinst möglichen Karton). Bei grösseren Stückzahlen ist es notwendig, den Prozess zu automatisieren; es wird also ein Algorithmus benötigt, der als Computerprogramm realisiert werden kann. Leider ist «rectangle packing», wie viele andere Verpackungsprobleme auch, *NP-vollständig*. Das heisst, dass es für das Problem sehr wahrscheinlich keinen *effizienten Algorithmus* gibt, der optimale Lösungen findet. In der Informatik werden für NP-

vollständige Probleme deshalb effiziente Algorithmen entwickelt, die zwar nicht garantiert optimale, aber nachweisbar gute Lösungen finden können.

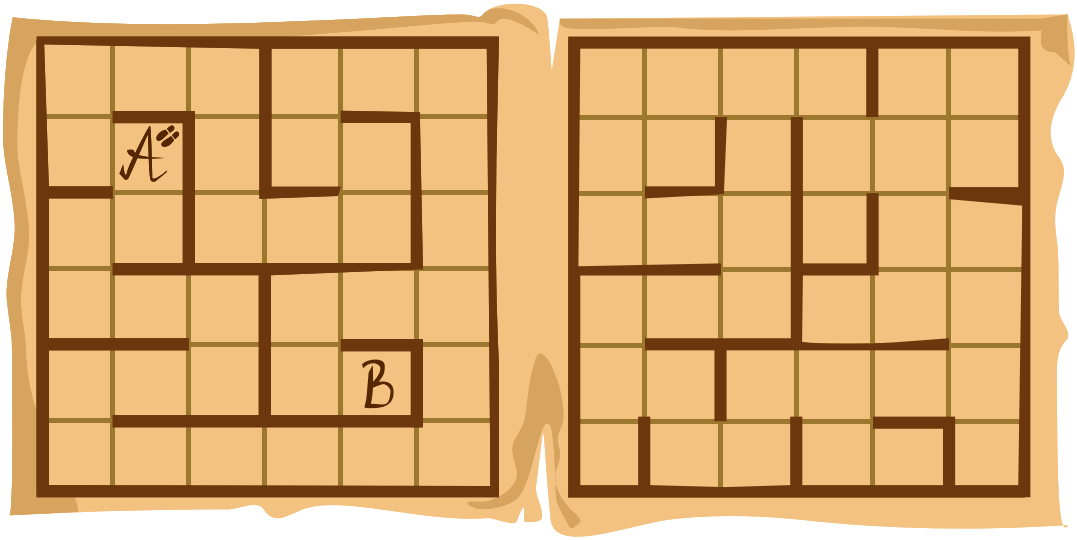
Unter anderem für Logistikunternehmen sind effiziente Lösungsansätze für Probleme solcher Art von grosser Bedeutung, z. B. zum Einlagern in Hochregalen, fürs platzsparende Verpacken von Waren, oder zur Verteilung von Waren auf Container. Ausserdem können scheinbar andersartige Probleme als Verpackungsprobleme beschrieben werden. Ein Arbeitsprozess, den N Arbeitskräfte in M Stunden bewältigen können, kann zum Beispiel als $N \times M$ Rechteck dargestellt werden. Mehrere Prozesse kann man also mit möglichst geringem Aufwand an Personen und Zeit bewältigen, wenn man für die entsprechenden Rechtecke das «rectangle packing» Problem optimal löst.

Stichwörter und Webseiten

- NP-Vollständigkeit: `https://de.wikipedia.org/wiki/NP-Vollständigkeit`
- Optimierung: `https://de.wikipedia.org/wiki/Optimierung_(Mathematik)`
- Effizienter Algorithmus: `https://de.wikipedia.org/wiki/Effizienz_(Informatik)`

15. Zauberschule

Die Zauberschule hat zwei Stockwerke. Die Stockwerke liegen genau übereinander. Beide sind in Felder eingeteilt, und es gibt Wände zwischen einigen Feldern:



Zauberschüler Ron braucht 1 Sekunde, um auf dem gleichen Stockwerk von einem Feld zum nächsten zu gehen. Leider hat Ron vergessen, wie er durch Wände gehen kann. Er kann aber von einem Stockwerk zum entsprechenden Feld des anderen Stockwerks kommen; dazu braucht er 5 Sekunden.

Ron möchte von Feld A zu Feld B gelangen, und zwar so schnell wie möglich.

Wie viele Sekunden braucht Ron dazu mindestens?

A) 6 Sekunden

B) 16 Sekunden

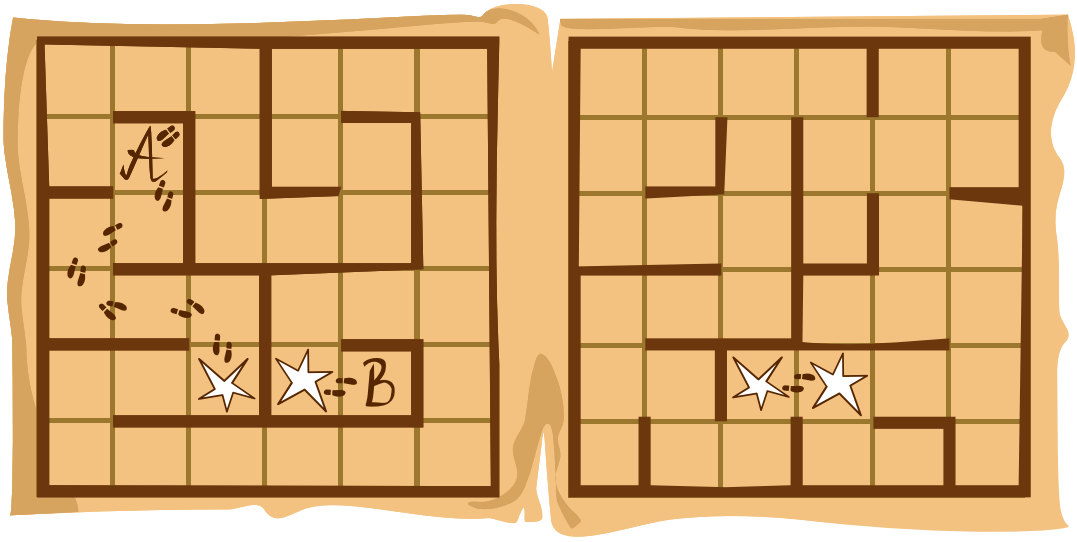
C) 18 Sekunden

D) 20 Sekunden

Lösung

Antwort C) 18 Sekunden ist richtig.

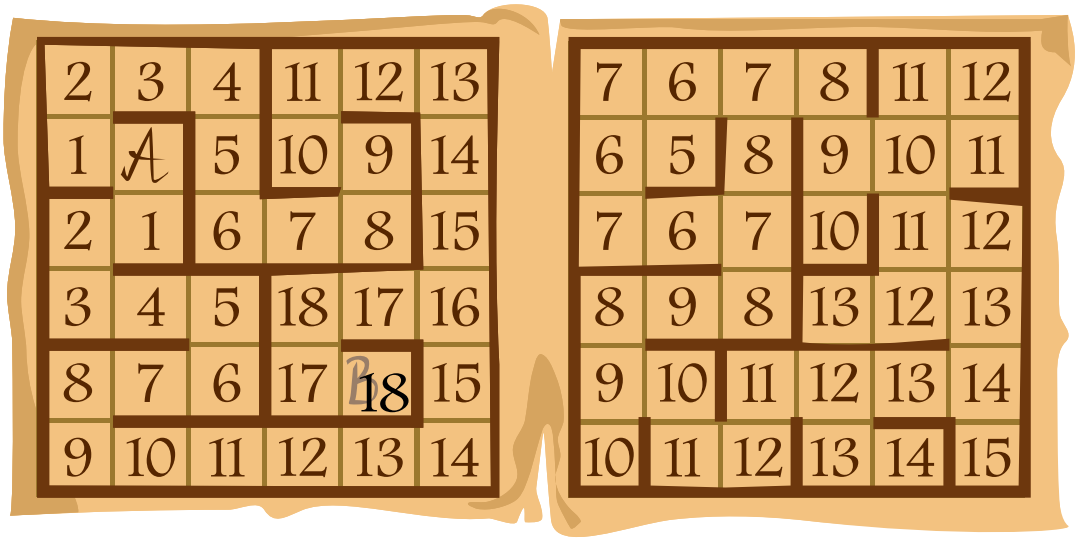
So kann Ron in 18 Sekunden von A nach B gelangen:



Aber ist das der schnellste Weg? Die «kürzesten Zeiten» die Ron benötigt, um von Feld A aus zu irgendeinem anderen Feld zu gelangen, kann man nach und nach so berechnen:

Für Feld A beträgt die kürzeste Zeit offensichtlich 0 Sekunden. Dann geht es schrittweise so weiter: Von allen Feldern, für die bereits eine kürzeste Zeit eingetragen ist, wählt man das mit dem geringsten Wert. (Ganz am Anfang wählt man also Feld A.) Von diesem gewählten Feld aus betrachtet man alle möglichen nächsten Felder und überlegt, wie man vom gewählten Feld am schnellsten dort hin kommt; die berechneten Zeiten trägt man bei den nächsten Feldern ein. Dabei kann es passieren, das eine vorher eingetragene Zeit verbessert wird. Das gewählte Feld darf danach nicht mehr betrachtet werden; es kann also in den nächsten Schritten nicht mehr gewählt werden.

Hier sind die kürzesten Zeiten, die mit dieser Methode berechnet werden, ausgehend von Feld A:



Ron braucht also wirklich mindestens 18 Sekunden, um von Feld A zu Feld B zu gelangen. 6 Sekunden (Antwort A)wäre die Dauer des kürzesten Weges, wenn es keine Wände zwischen den Feldern gäbe. Wenn Ron dann trotzdem einmal zwischen den Stockwerken hin und her wechselte, würden 16 Sekunden (Antwort B) daraus. Gäbe es nur das Stockwerk mit den Feldern A und B, wären 20 Sekunden (Antwort D) die kürzeste Zeit für den Weg von A nach B.

Dies ist Informatik!

Schnellste oder kürzeste Wege müssen recht häufig berechnet werden; ein offensichtliches Beispiel ist die Routenplanung in modernen Karten-Apps. Das Problem wird deutlich vereinfacht, wenn die Wege aus einzelnen Schritten zwischen benachbarten Punkten bestehen und für alle diese Schritte bekannt ist, wie viel sie «kosten»: Zeit, Geld, Energieverbrauch – was auch immer die für das aktuelle Problem wichtige Grösse ist. In diesem Fall lassen sich Punkte, Schritte und die Kosten der Schritte zu einem *Graph* abstrahieren, in dem Schritte zu Wegen zusammengesetzt werden können. Für Graphen sind in der Informatik viele Algorithmen bekannt, mit denen *kürzeste Wege* effizient berechnet werden können. Einer davon wurde vom Informatiker Edsger Dijkstra erfunden; dieser *Dijkstra-Algorithmus* kam oben bei der Erklärung der richtigen Antwort zum Einsatz.

Auch beim Entwurf von Schaltkreisen für Computer spielen kürzeste Wege eine wichtige Rolle. Dabei müssen Schaltpunkte mit möglichst geringen Kosten miteinander verdrahtet werden. Moderne Schaltkreise bestehen aus mehreren Ebenen, und eine Verdrahtung zwischen zwei Ebenen ist teurer als eine (ansonsten vergleichbare) Verdrahtung auf der gleichen Ebene – ähnlich zum Wechsel zwischen den Stockwerken in dieser Biberaufgabe, der teurer ist als ein Schritt auf dem gleichen Stockwerk.

Stichwörter und Webseiten

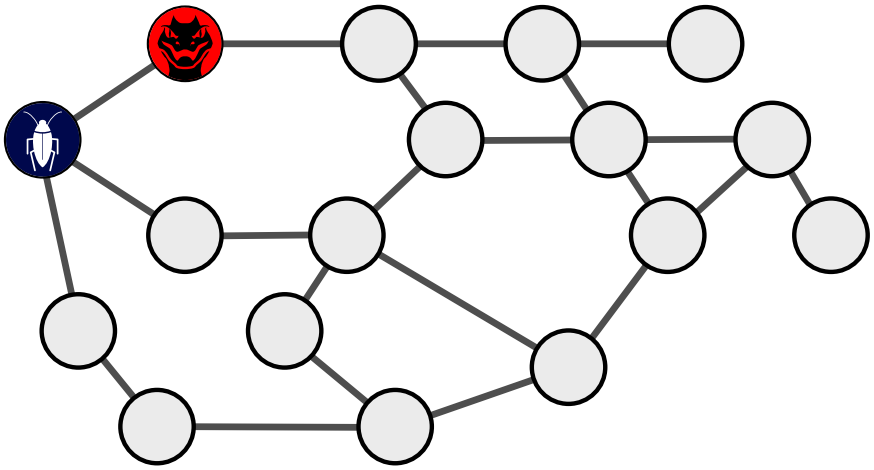
- Graph: `https://de.wikipedia.org/wiki/Graph_(Graphentheorie)`
- Kürzester Pfad: `https://de.wikipedia.org/wiki/Kürzester_Pfad`
- Edsger Dijkstra: `https://de.wikipedia.org/wiki/Edsger_W._Dijkstra`
- Dijkstra-Algorithmus: `https://de.wikipedia.org/wiki/Dijkstra-Algorithmus`

16. Virus

In einem Computernetz haben sich zwei Netzknoten mit Computerviren infiziert: einer mit dem Virus BlueBug, ein anderer mit dem Virus RedRaptor. Immer am Morgen breiten sich beide Viren aus. Jedes Virus infiziert dann zusätzlich alle Knoten, die mit den von ihm bereits infizierten Knoten direkt verbunden sind. Wenn ein Knoten mit beiden Viren infiziert ist, schaltet er nach einigen Stunden wegen Überlastung ab. Die Viren können sich an den folgenden Tagen von dort also nicht weiter ausbreiten.

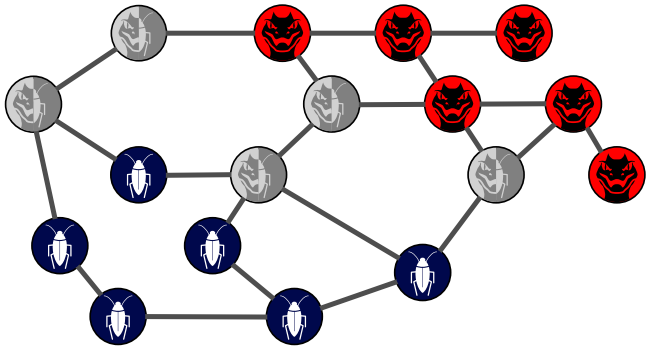
Unten siehst du das Computernetz mit den Knoten und ihren direkten Verbindungen. Die beiden zu Beginn infizierten Knoten sind markiert. Nach einigen Tagen sind alle Knoten mit einem Virus infiziert oder sogar abgeschaltet.

Welche Knoten sind dann mit welchem Virus infiziert oder abgeschaltet?

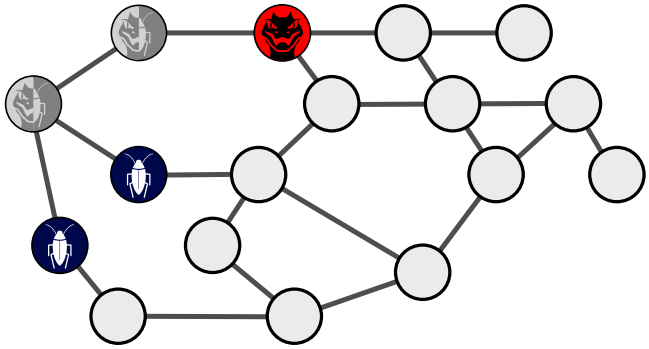


Lösung

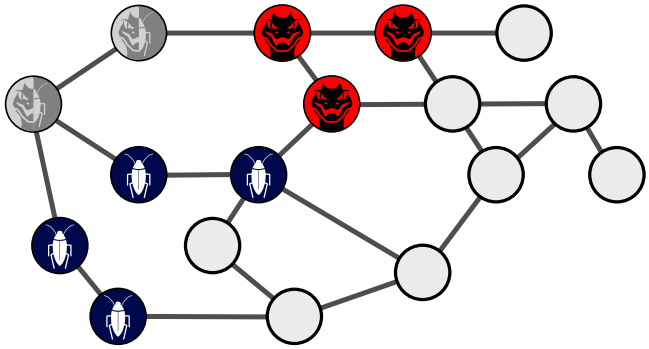
Nach 5 Tagen sind alle Netzwerkknoten infiziert oder abgeschaltet. Dies ist die richtige Lösung:



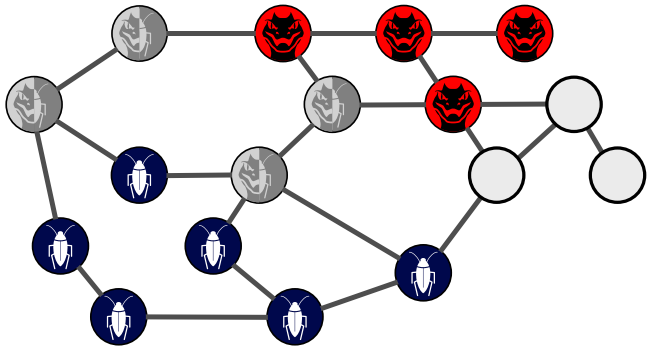
Nach 1 Tag sind fünf Netzknoten infiziert. Die beiden zu Beginn infizierten Knoten sind nun mit beiden Viren infiziert und deswegen abgeschaltet:



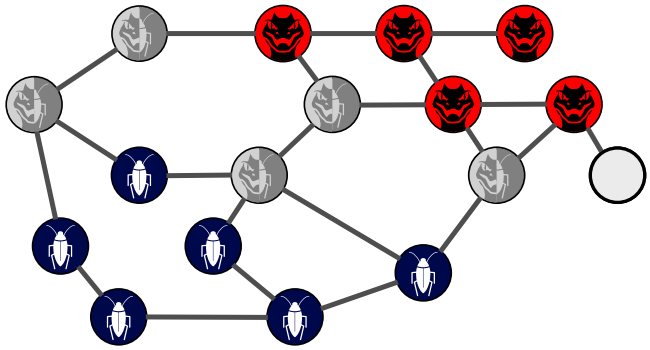
Nach 2 Tagen sind vier weitere Knoten infiziert:



Nach 3 Tagen sind zwei Knoten doppelt infiziert und nun ebenfalls abgeschaltet. Zudem sind drei weitere Knoten mit «BlueBug» und zwei mit «RedRaptor» infiziert:



Nach 4 Tagen ist ein weiterer Netzwerkknoten ausgeschaltet. «BlueBug» kann sich nun nicht mehr weiter ausbreiten.



Am 5. Tag wird der letzte Knoten mit dem «RedRaptor» infiziert.

Dies ist Informatik!

In Computernetzen stellen Viren und andere Schadsoftware eine grosse Bedrohung dar. Sie beeinflussen nicht nur die Leistungsfähigkeit der betroffenen Computer, häufig haben sie noch eine «Nutzlast» (*payload*), die zusätzlichen Schaden anrichtet. In manchen Fällen werden beispielsweise übertragene Daten mitgelesen und so sensible Informationen wie Passwörter oder Benutzerdaten herausgefunden und an einen Auftraggeber übermittelt. In einigen Fällen werden vom Virus Daten auf dem befallenen Computer verschlüsselt. Will der Benutzer wieder auf seine Daten zugreifen, muss er erst einen Geldbetrag auf ein anonymes Konto überweisen. Manchmal werden Gruppen infizierter Computer von Kriminellen ferngesteuert, um Angriffe auf andere Computer durchzuführen (*Botnet*).

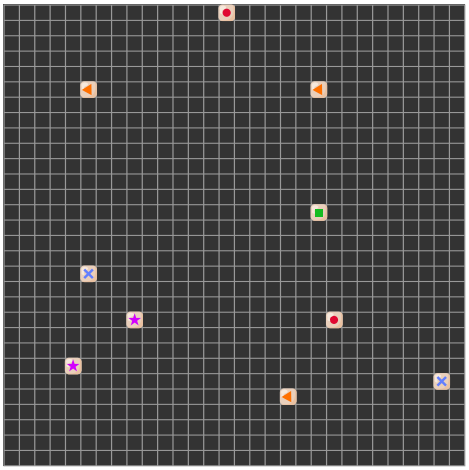
Dass ein Virus einen Computer ganz lahmlegt, ist normalerweise vom Urheber des Virus nicht beabsichtigt, denn dadurch wird die Verbreitung des Virus gestoppt. Manche Viren werden aber gezielt für Sabotage und Cyberkrieg (*Cyberwarfare*) entwickelt. Dadurch können betroffene Computer sogar dauerhaft beschädigt werden.

Die Einspielung aktueller Sicherheitsupdates ist eine wichtige Voraussetzung für die Abwehr von Viren, Antivirusprogramme können den Schutz verbessern, sind aber in manchen Betriebssysteme schon enthalten, sodass eventuell kein zusätzliches Programm erforderlich ist. Regelmässige Datensicherungen und Wachsamkeit im Bezug auf ungewöhnliches Verhalten des Systems sind aber unabdingbar.

Stichwörter und Webseiten

- Computernetz: `https://de.wikipedia.org/wiki/Rechnernetz`
- Virus: `https://de.wikipedia.org/wiki/Computervirus`
- Payload: `https://de.wikipedia.org/wiki/Computervirus#Payload`
- Botnet: `https://de.wikipedia.org/wiki/Botnet`
- Cyberwarfare: `https://de.wikipedia.org/wiki/Cyberkrieg`

17. Boden bemalen



Der Boden eines quadratischen Raumes ist in 30×30 Felder unterteilt. Auf zehn Feldern liegen Chips mit solchen farbigen Symbolen: ●, ✕, ◀, ■ und ★.

Ein Roboter soll den Boden mit diesen Symbolen bemalen, Feld für Feld. Er hat dafür vier verschiedene Regeln. Auf einem Feld, auf dem kein Chip liegt, malt er …

1 … das Symbol des Chips, der ihm am nächsten ist.

2 … das Symbol des Chips, der am weitesten von ihm entfernt ist.

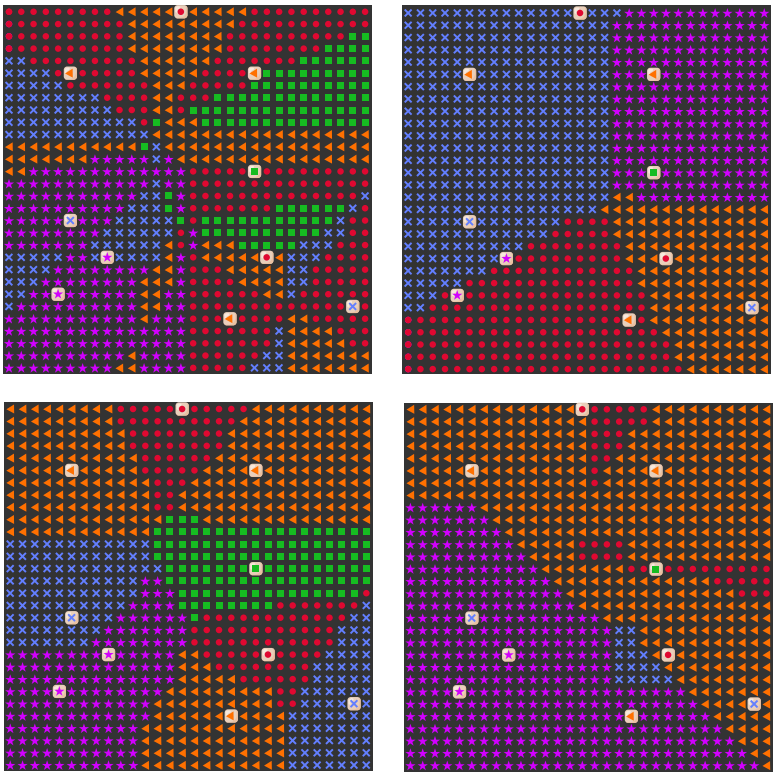
3 … das Symbol des Chips, der ihm am zweitnächsten ist.

4 … das Symbol, das bei den 6 am nächsten liegenden Chips am häufigsten vorkommt.

Der Roboter bemalt alle Felder nach derselben Regel. Wenn die Regel für ein Feld mehrere mögliche Symbole ergibt, sucht der Roboter sich zufällig eines davon aus.

Unten siehst du für jede Regel, wie der Boden am Ende jeweils bemalt ist.

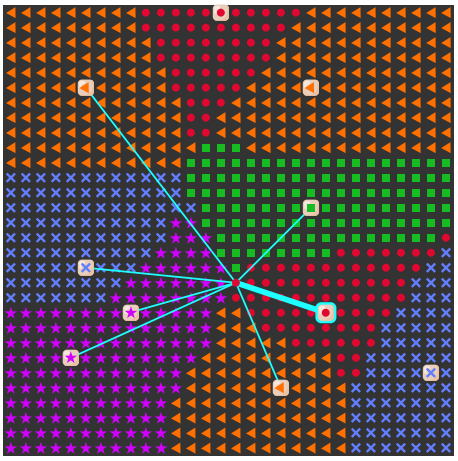
Welcher Boden passt zu welcher Regel? Ordne die Regeln den Böden zu.



Lösung

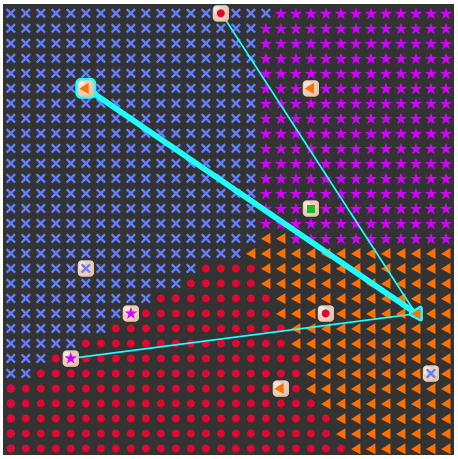
Da alle Felder eines Bodens nach derselben Regel bemalt werden, genügt es, jeweils ein Feld zu überprüfen. Für jeden Boden betrachten wir ein anderes Feld:

Regel 1



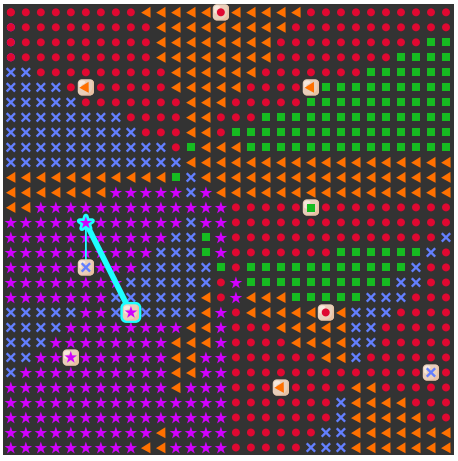
Feld ist mit ● bemalt, weil ein Chip ● am nächsten liegt.

Regel 2



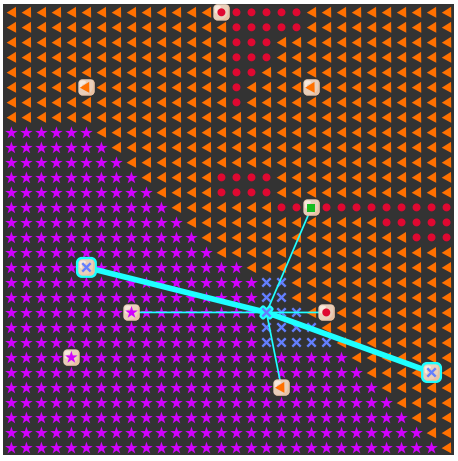
Feld ist mit ◀ bemalt, weil ein Chip ◀ am weitesten von entfernt ist.

Regel 3



Feld ist mit ★ bemalt, weil ein Chip ★ am zweitnächsten liegt.

Regel 4



Feld ist mit ✕ bemalt, weil dieses ✕ bei den 6 am nächsten liegenden Chips am häufigsten vorkommt.

Dies ist Informatik!

Teilungen einer Ebene sowie deren *algorithmische* Konstruktion spielen in verschiedenen Bereichen der Informatik eine wichtige Rolle, zum Beispiel bei Simulationen und in der Computergrafik.

Voronoi-Diagramme, benannt nach dem ukrainischen Mathematiker Georgi Feodosjewitsch Woronoi (*1868 - †1908), unterteilen eine Ebene in Regionen rund um sogenannte *Zentren*. Alle Punkte einer Region liegen keinem anderen Zentrum näher als dem eigenen. Das Ergebnis der Regel 1 ist ein Voronoi-Diagramm. Diese Diagramme kommen häufig in Situationen der realen Welt vor, beispielsweise in der Mobilfunkversorgung. Oder man nutzt sie in der Analyse von Fussballspielen oder anderem sozioökonomischen Verhalten, etwa den Beziehungen zwischen der Bevölkerung und den nächstgelegenen Schulen, Krankenhäusern oder bestimmten Dienstleistern.

Der Meteorologe Alfred H. Thiessen (*1872 - †1956) entwickelte 1911 mit Hilfe der Voronoi-Diagramme ein Verfahren die Durchschnittswerte (z. B. Niederschlagsmengen) von Gebieten realitätsgetreuer zu bestimmen. Den Durchschnitt der Messwerte der Messstationen bestimmt er nicht rein durch die Anzahl der Messtationen, sondern ermittelt anhand des Voronoi-Diagramms zuerst die Fläche, worauf sich die Messwerte beziehen. Dadurch entsteht eine unterschiedliche Gewichtung der lokalen Messwerte.

Stichwörter und Webseiten

- Algorithmus: `https://de.wikipedia.org/wiki/Algorithmus#Informatik_und_Mathematik`
- Voronoi-Diagramme: `https://de.wikipedia.org/wiki/Voronoi-Diagramm`

A. Aufgabenautoren

Esraa Almajhad
Liam Baumann
Wilfried Baumann
Linda Björk Bergsveinsdóttir
Graeme Buckie
Marta J. Burzanska
Sarah Chan
Kris Coolsaet
Darija Dasović
Christian Datzko
Susanne Datzko
Justina Dauksaite
Nora A. Escherle
Gerald Futschek
Adam Grodeck
Benjamin Hirsch
Alisher Ikramov
Thomas Ioannou
Mile Jovanov
Dong Yoon Kim
Hakin Kim
Vaidotas Kinčius
Lidija Kralj
Regula Lacher
Taina Lehtimäki
Marielle Léonard
Monika Maneva
Zoran Milevski
Madhavan Mukund
Ágnes Erdősné Németh
Ilze Nilandere
Veronika Ognjanovska
Mārtiņš Opmanis
Margot Phillipps
Zsuzsa Pluhár
Wolfgang Pohl
John-Paul Pretti
Susannah Quidilla
Lorenzo Repetto
Chris Roffey
Kirsten Schlüter
Giovanni Serafini
Timur Sitdikov
Bernadette Spieler
Emil Stankov
Veronika Stefanovska
Alieke Stijf
Goran Sukovic
Monika Tomcsányiová
Jiří Vaníček
Troy Vasiga
Willem van der Vegt

 Rechilda Villame

 Kyra Willekes

 Michael Weigend

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.