INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Aufgaben und Lösungen 2019
Schuljahre 11/12/13

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

Herausgeber:
Christian Datzko, Susanne Datzko, Juraj Hromkovič, Regula Lacher

01010011010101100100100101000001001011010101001101010011010010010100001010010010100100001
SV!A
www.svia-ssie-ssii.ch
schweizerischerverein für informatikin derausbildung//société suisse pour l'infor matiquedans l'enseignement//società sviz zera per l'informatica nell'insegnamento

Mitarbeit Informatik-Biber 2019

Christian Datzko, Susanne Datzko, Olivier Ens, Hanspeter Erni, Nora A. Escherle, Martin Guggisberg, Saskia Howald, Lucio Negrini, Gabriel Parriaux, Elsa Pellet, Jean-Philippe Pellet, Beat Trachsler.

Herzlichen Dank an:
Juraj Hromkovič, Michelle Barnett, Michael Barot, Anna Laura John, Dennis Komm, Regula Lacher, Jacqueline Staub, Nicole Trachsler: ETHZ
Gabriel Thullen: Collège des Colombières
Valentina Dagienė: Bebras.org
Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend: Bundesweite Informatikwettbewerbe (BWINF), Deutschland
Chris Roffey: University of Oxford, Vereinigtes Königreich
Carlo Bellettini, Violetta Lonati, Mattia Monga, Anna Morpurgo: ALaDDIn, Università degli Studi di Milano, Italien
Gerald Futschek, Wilfried Baumann, Florentina Voboril: Oesterreichische Computer Gesellschaft, Österreich
Zsuzsa Pluhár: ELTE Informatikai Kar, Ungarn
Eljakim Schrijvers, Justina Dauksaite, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers, Kyra Willekes, Saskia Zweerts: Cuttle.org, Niederlande
Christoph Frei: Chragokyberneticks (Logo Informatik-Biber Schweiz)
Andrea Leu, Maggie Winter, Brigitte Manz-Brunner: Senarclens Leu + Partner

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 Veronica Ostini erstellt.

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Der Informatik-Biber 2019 wurde vom Schweizerischen Verein für Informatik in der Ausbildung SVIA durchgeführt und von der Hasler Stiftung unterstützt.

HASLERSTIFTUNG

Hinweis: Alle Links wurden am 1. November 2019 geprüft. Dieses Aufgabenheft wurde am 2. Januar 2020 mit dem Textsatzsystem LaTeX erstellt.

Vorwort

Der Wettbewerb „Informatik-Biber“, der in verschiedenen Ländern der Welt schon seit mehreren Jahren bestens etabliert ist, will das Interesse von Kindern und Jugendlichen an der Informatik wecken. Der Wettbewerb wird in der Schweiz in Deutsch, Französisch und Italienisch vom Schweizerischen Verein für Informatik in der Ausbildung SVIA durchgeführt und von der Hasler Stiftung im Rahmen des Förderprogramms FIT in IT unterstützt.
Der „Informatik-Biber“ ist der Schweizer Partner der Wettbewerbs-Initiative „Bebras International Contest on Informatics and Computer Fluency“ (`https://www.bebras.org/`), die in Litauen ins Leben gerufen wurde.
Der Wettbewerb wurde 2010 zum ersten Mal in der Schweiz durchgeführt. 2012 wurde zum ersten Mal der „Kleine Biber“ (Stufen 3 und 4) angeboten.
Der „Informatik-Biber“ regt Schülerinnen und Schüler an, sich aktiv mit Themen der Informatik auseinander zu setzen. Er will Berührungsängste mit dem Schulfach Informatik abbauen und das Interesse an Fragenstellungen dieses Fachs wecken. Der Wettbewerb setzt keine Anwenderkenntnisse im Umgang mit dem Computer voraus – ausser dem „Surfen“ auf dem Internet, denn der Wettbewerb findet online am Computer statt. Für die Fragen ist strukturiertes und logisches Denken, aber auch Phantasie notwendig. Die Aufgaben sind bewusst für eine weiterführende Beschäftigung mit Informatik über den Wettbewerb hinaus angelegt.
Der Informatik-Biber 2019 wurde in fünf Altersgruppen durchgeführt:

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

Die Stufen 3 und 4 hatten 9 Aufgaben zu lösen, jeweils drei davon aus den drei Schwierigkeitsstufen leicht, mittel und schwer. Die Stufen 5 und 6 hatten 12 Aufgaben zu lösen, jeweils vier davon aus den drei Schwierigkeitsstufen leicht, mittel und schwer. Jede der anderen Altersgruppen hatte 15 Aufgaben zu lösen, jeweils fünf davon aus den drei Schwierigkeitsstufen leicht, mittel und schwer.
Für jede richtige Antwort wurden Punkte gutgeschrieben, für jede falsche Antwort wurden Punkte abgezogen. Wurde die Frage nicht beantwortet, blieb das Punktekonto unverändert. Je nach Schwierigkeitsgrad wurden unterschiedlich viele Punkte gutgeschrieben beziehungsweise abgezogen:

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

Das international angewandte System zur Punkteverteilung soll dem erfolgreichen Erraten der richtigen Lösung durch die Teilnehmenden entgegenwirken.
Jede Teilnehmerin und jeder Teilnehmer hatte zu Beginn 45 Punkte („Kleiner Biber“: 27 Punkte, Stufen 5 und 6: 36 Punkte) auf dem Punktekonto.
Damit waren maximal 180 Punkte („Kleiner Biber“: 108 Punkte, Stufen 5 und 6: 144 Punkte) zu erreichen, das minimale Ergebnis betrug 0 Punkte.
Bei vielen Aufgaben wurden die Antwortalternativen am Bildschirm in zufälliger Reihenfolge angezeigt. Manche Aufgaben wurden in mehreren Altersgruppen gestellt.

Für weitere Informationen:

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

Inhaltsverzeichnis

1. Rauchsignale

Ein Biber sitzt immer oben auf dem Berg und beobachtet das Wetter. Er übermittelt den Bibern im Tal, wie das Wetter werden wird. Er nutzt dazu Rauchsignale, die aus fünf nacheinander folgenden Rauchwolken bestehen. Eine Rauchwolke ist entweder klein oder gross. Die Biber haben folgende Rauchsignale vereinbart:

Es wird gewittrig.	Es wird regnerisch.	Es wird bewölkt.	Es wird sonnig.

An einem windigen Tag können die Biber im Tal die Rauchwolken nicht gut erkennen. Sie interpretieren Folgendes:

Da dies keines der vereinbarten Rauchsignale ist, nehmen sie an, dass sie eine der Rauchwolken falsch interpretiert haben: eine kleine Rauchwolke müsste also eigentlich gross sein oder eine grosse Rauchwolke müsste eigentlich klein sein.
Wenn also genau eine Rauchwolke falsch interpretiert wurde, was wäre die Bedeutung?

A) Es wird gewittrig.

B) Es wird regnerisch.

C) Es wird bewölkt.

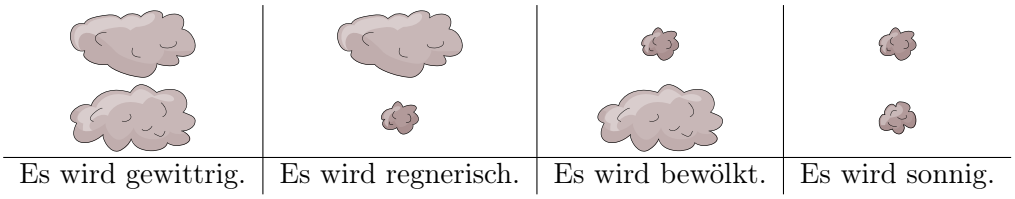
D) Es wird sonnig.

Lösung

Wenn genau eine Rauchwolke falsch interpretiert wurde, könnte es fünf verschiedene Rauchsignale ergeben. Die erste, zweite, vierte oder fünfte Rauchwolke anders zu interpretieren führt jedoch zu keinem der vier vereinbarten Rauchsignale. Die dritte Rauchwolke als kleine Rauchwolke zu interpretieren ergibt aber das Rauchsignal der richtigen Antwort C) „Es wird bewölkt“.
Man kann auch das interpretierte Rauchsignal mit den vier vereinbarten Rauchsignalen vergleichen und schauen, wie viele Rauchwolken unterschiedlich sind. Das sind beim Rauchsignal für „Es wird gewittrig“ zwei Rauchwolken (die oberste und die unterste), beim Rauchsignal für „Es wird regnerisch“ drei Rauchwolken (die obersten beiden und die zweitunterste), beim Rauchsignal für „Es wird bewölkt“ eine Rauchwolke (die mittlere, damit ist dies wie oben geschrieben die richtige Lösung) und beim Rauchsignal für „Es wird sonnig“ vier Rauchwolken (alle bis auf die oberste).

Dies ist Informatik!

Wenn man eine Nachricht übermitteln muss, möchte man, dass die Nachricht richtig beim Empfänger ankommt. Die Nachricht in dieser Aufgabe wird mit Hilfe von grossen und kleinen Rauchwolken übermittelt. Im allgemeinen Fall spricht man von *Symbolen*. Daher ist es sinnvoll, eine Folge von Symbolen so zu wählen, dass die zu übermittelnde Nachricht auch dann verstanden werden kann, wenn sie unterwegs beschädigt wurde. Dies kann man erreichen, indem man mehr Information kommuniziert als absolut notwendig. Man nennt diese zusätzliche Information *redundant*.
Wenn man die beschädigte Nachricht mit höchstens n Fehlern rekonstruieren kann, spricht man von n-selbstkorrigierenden Kodierungen. Nachrichten als Folgen von Symbolen so darzustellen, dass man die Nachrichten rekonstruieren kann, auch wenn ihre Darstellung unterwegs beschädigt wurde, ist eine typische Aufgabe für Informatiker. Sie ermöglichen so zum Beispiel, Musik von CDs oder Videos von DVDs korrekt abzuspielen, auch wenn bei der Übertragung einige Fehler aufgetreten sind.
Für diese Aufgabe hätten übrigens zwei Rauchwolken genügt, um die vier unterschiedlichen Nachrichten zu übermitteln:



Die Biber verwenden aber fünf Rauchwolken. Das erlaubt ihnen in Fällen wo zwei oder in einigen Fällen sogar drei Rauchwolken „unlesbar“ sind, die Nachricht trotzdem richtig zu verstehen. Die Biber haben sich die Nachrichten übrigens so überlegt, dass sich je zwei Nachrichten an mindestens drei Stellen unterscheiden.

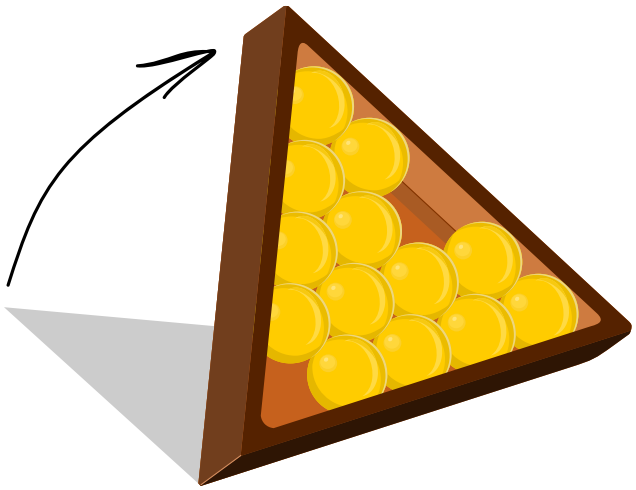
Stichwörter und Webseiten

Fehlerkorrekturverfahren

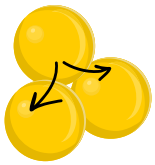
- `https://de.wikipedia.org/wiki/Fehlerkorrekturverfahren`

2. Wackelige Kugeln

In eine dreieckige Box passen fünfzehn gleich grosse Kugeln. Zwei Kugeln werden entfernt wie in der Zeichnung gezeigt. Die Box wird nun gekippt.



Beim Kippen können einige Kugeln „wackelig“ werden. Eine Kugel ist wackelig, wenn ...



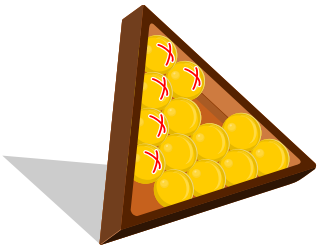
- ... die Kugel links unter ihr oder rechts unter ihr entfernt wurde, ...
- ... oder die Kugel links unter ihr oder rechts unter ihr wackelig ist.

Die Kugeln der untersten Reihe sind nicht wackelig.
Wie viele von den dreizehn Kugeln sind wackelig?

A) Keine Kugel
B) 1 Kugel
C) 2 Kugeln
D) 3 Kugeln
E) 4 Kugeln
F) 5 Kugeln
G) 6 Kugeln
H) 7 Kugeln
I) 8 Kugeln
J) 9 Kugeln
K) 10 Kugeln
L) 11 Kugeln
M) 12 Kugeln
N) Alle Kugeln

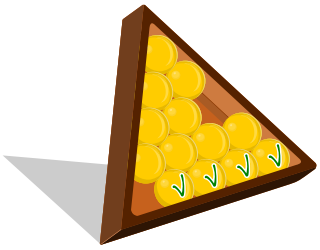
Lösung

Fünf Kugeln sind wackelig. Sie sind in der folgenden Zeichnung gekennzeichnet:

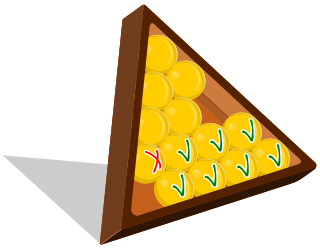


Am einfachsten überlegt man sich das von unten nach oben:

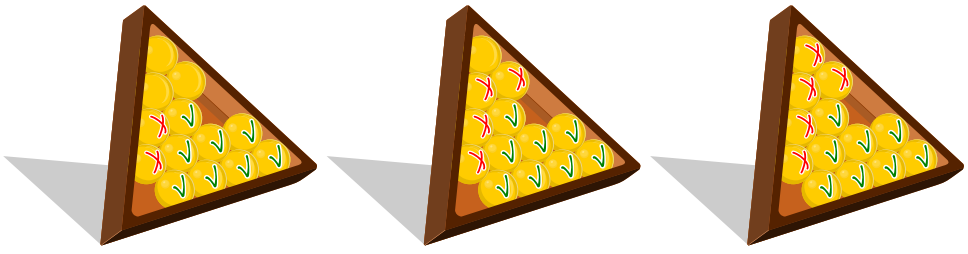
- Alle Kugeln der untersten Reihe sind nicht wackelig.



- Alle Kugeln der zweituntersten Reihe, unter der zwei Kugeln liegen, die nicht wackelig sind, sind ebenfalls nicht wackelig, alle anderen sind wackelig.



- Dies wird bis zur obersten Reihe fortgeführt.



Dies ist Informatik!

Es gibt zwei Bedingungen, welche eine Kugel als wackelig klassifizieren. Die erste Bedingung kann direkt überprüft werden. Damit die zweite Bedingung überprüft werden kann, muss man zuerst wissen, ob sich in der Reihe direkt darunter eine wackelige Kugel befindet. Das ist in der untersten Reihe einfach, denn dort sind alle Kugeln nicht wackelig, da es keine weitere Reihe unter ihnen gibt. Wie in der Lösung erklärt, kann man danach die Reihe darüber überprüfen und dort herausfinden,

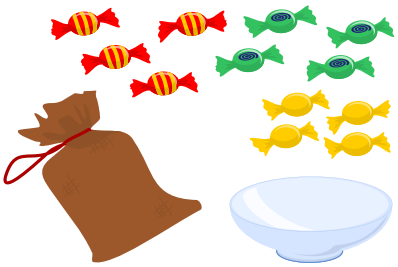
welche Kugeln wackelig sind. Auf diese Art und Weise kann man systematisch alle Reihen von unten nach oben durchgehen und für alle Kugeln herausfinden, ob sie wackelig sind.
Das Prinzip, dass eine Bedingung von dem Ergebnis einer anderen gleichartigen Bedingung abhängig ist, heisst *Rekursion*. Rekursive Bedingungen sind so aufgebaut, dass ihr Ergebnis entweder offensichtlich ist (*Rekursionsende*, in diesem Fall sind alle Kugeln der untersten Reihe nicht wackelig) oder von dem Ergebnis weiterer rekursiver Bedingungen abhängt (*Rekursionsschritt*, in diesem Fall sind das alle Kugeln, die nicht in der untersten Reihe liegen, so dass für sie zunächst die Kugeln darunter geprüft werden müssen).
Das Prinzip der Rekursion wird in der Informatik häufig verwendet. Mit ihm kann man sehr einfach und elegant viele komplexe Probleme lösen. Es ist aber auch möglich, rekursive Lösungsansätze in schrittweise (*iterative*) Lösungsansätze umzuwandeln. Ein klassisches Beispiel wo ein rekursiver Lösungsansatz sehr einfach ist, sind die Türme von Hanoi.

Stichwörter und Webseiten

Rekursion, Türme von Hanoi

- `https://de.wikipedia.org/wiki/Rekursion`
- `https://de.wikipedia.org/wiki/T%C3%BCrme_von_Hanoi`

3. Ein Sack voller Bonbons



Petra hat in einem undurchsichtigen Sack vier rote, vier grüne und vier gelbe Bonbons. Zudem hat sie eine leere Schale.
Petra und Moritz spielen ein Spiel. Moritz darf während drei Runden ein Bonbon aus dem Sack ziehen. Für jedes gezogene Bonbon gelten folgende Regeln:

- Solange das gezogene Bonbon grün ist, legt er es in die Schale und er darf in dieser Runde ein weiteres Bonbon ziehen.
- Wenn das gezogenen Bonbon rot ist, legt es Moritz in die Schale und beendet die Runde.
- Wenn das gezogene Bonbon gelb ist, isst Moritz es direkt, ohne es in die Schale zu legen, und beendet die Runde.

Wie viele Bonbons hat Moritz am Ende des Spiels maximal in der Schale liegen?

A) 0
B) 1
C) 2
D) 3
E) 4
F) 5
G) 6
H) 7
I) 8
J) 9
K) 10
L) 11
M) 12

Lösung

Die richtige Antwort ist H) 7.
Im günstigsten Fall werden alle vier grünen Bonbons gezogen. Das bedeutet, dass zum einen die vier grünen Bonbons in der Schale liegen und zum anderen, dass Moritz im Laufe der drei Runden vier Mal ein weiteres Bonbon ziehen durfte, also insgesamt sieben.
Für die restlichen drei Bonbons zieht Moritz im günstigsten Fall jeweils ein rotes Bonbon, die dann ebenfalls am Ende in der Schale liegen. Das macht dann insgesamt vier grüne und drei rote Bonbons, es liegen also sieben Bonbons in der Schale.
Mehr als sieben Bonbons können es nicht sein. Nach jedem Zug kommt höchstens ein Bonbon in die Schale und da es nur vier grüne Bonbons gibt, bei denen man ein weiteres Bonbon ziehen kann, sind es maximal sieben Bonbons.
Die Reihenfolge, in der die Bonbons im günstigsten Fall gezogen werden, ist relativ egal, solange das letzte gezogene Bonbon ein rotes ist, denn dann kann man durch die grünen Bonbons immer noch ein weiteres ziehen.

Dies ist Informatik!

Zwei der drei Regeln der Aufgabe sind als *Verzweigungen* formuliert: *wenn* eine bestimmte Bedingung zutrifft, *dann* wird eine bestimmte Aktion ausgeführt. Solche Verzweigungen kommen beim Programmieren sehr häufig vor. Häufig werden hierfür die englischsprachigen Schlüsselwörter *if* (engl. für „wenn“) und *then* (engl. für „dann“) verwendet. Eine der Regeln ist so formuliert, dass etwas *solange* wiederholt wird, *bis* eine bestimmte Bedingung nicht mehr stimmt. So etwas nennt man eine *Schleife*, für die häufig das englische Schlüsselwort *while* (engl. für „solange“) verwendet wird. Solche Schleifen können auch als *Zählschleife* formuliert sein, die eine bestimmte Anzahl Wiederholungen vorgibt.
Man könnte also das Spiel von Petra auch so formulieren:

setze Runden auf 3
solange noch mindestens eine Runde vorhanden ist:
    verringere Runden um 1
    ziehe ein Bonbon
    solange das Bonbon grün ist, lege es in die Schale und ziehe ein Bonbon
    wenn das Bonbon rot ist, *dann* lege es in die Schale
    wenn das Bonbon gelb ist, *dann* iss es

Um die Aufgabe zu lösen, muss man das Programm *analysieren*. In einem so einfachen Fall wie diesem Programm könnte man natürlich einfach alle möglichen Reihenfolgen von Bonbons ausprobieren. Dies könnte sogar von einem Computer automatisiert durchgeführt werden. Die in der Lösung gelieferte Erklärung hingegen basiert darauf, die Zusammenhänge zu verstehen und so zu beweisen, dass ein bestimmtes Ergebnis wahr ist, ohne dass das Programm ausgeführt wird. Solche Analysen sind, wie die *Berechenbarkeitstheorie* zeigen konnte, nicht in jedem Fall von einem Computer durchführbar. Donald Knuth, einer der grossen Informatiker des 20. Jahrhunderts hat es mal so auf den Punkt gebracht: *„Vorsicht vor Fehlern im Code; ich habe nur bewiesen, dass er korrekt ist, ich habe ihn nicht ausprobiert“.*

Stichwörter und Webseiten

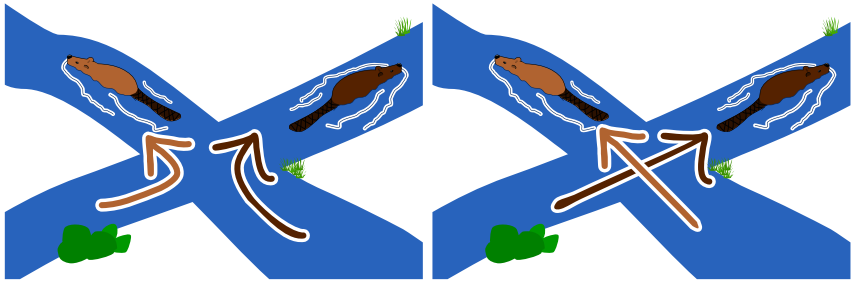
Verzweigung, Schleife, Berechenbarkeitstheorie

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

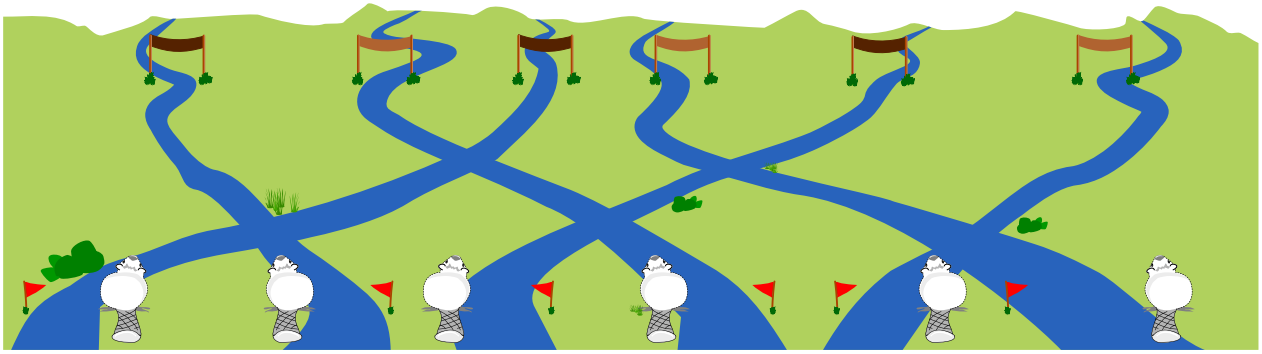
- `https://de.wikipedia.org/wiki/Schleife_(Programmierung)`
- `https://de.wikipedia.org/wiki/Berechenbarkeitstheorie`
- `https://en.wikiquote.org/wiki/Donald_Knuth`

4. Bibernetzwerk

Drei hellbraune und drei dunkelbraune Biber schwimmen durch ein Kanalsystem von unten nach oben. An jeder Kreuzung von zwei Kanälen treffen sich zwei Biber. Wenn diese beiden Biber unterschiedliche Farben haben, schwimmt der hellbraune Biber links und der dunkelbraune Biber rechts weiter. Sonst schwimmt einfach einer links und einer rechts weiter.

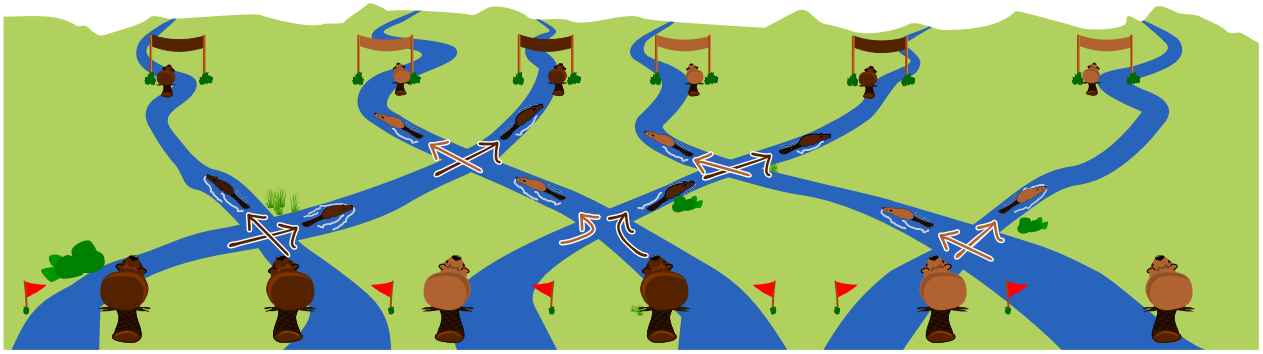


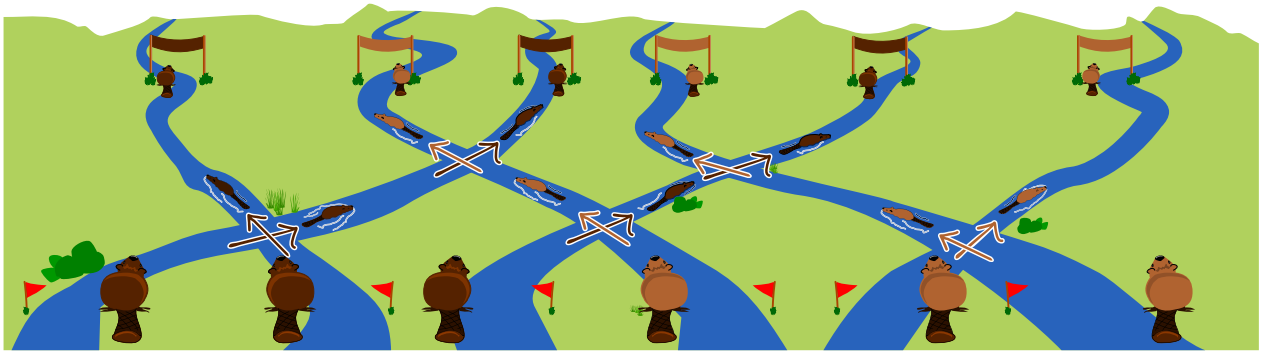
Am Ende sollen die Biber in der folgenden Reihenfolge von links nach rechts ankommen: dunkelbraun, hellbraun, dunkelbraun, hellbraun, dunkelbraun und hellbraun.
Wie müssen die drei hellbraunen und die drei dunkelbraunen Biber starten, dass die Reihenfolge bei der Ankunft korrekt ist?



Lösung

Es gibt zwei richtige Antworten:

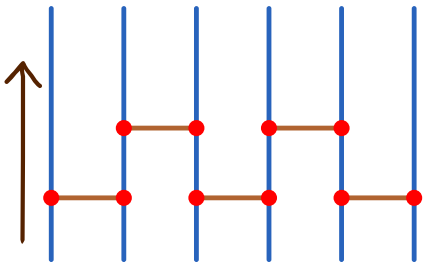




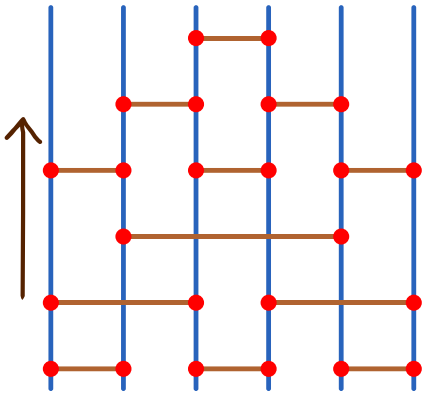
Dies sind auch die beiden einzigen richtigen Antworten. Damit nämlich an der linken Zielposition ein dunkelbrauner Biber ankommen kann, darf zur ersten Kreuzung von links kein hellbrauner Biber schwimmen, da dieser sonst nach links schwimmen müsste. Somit müssen die beiden linken Startpositionen durch zwei dunkelbraune Biber besetzt werden.
Dasselbe gilt für die rechte Zielposition des hellbraunen Bibers: Damit ganz rechts ein hellbrauner Biber ankommen kann, müssen an der ersten Kreuzung von rechts zwei hellbraune Biber aufeinander treffen. Somit müssen die beiden rechten Startpositionen durch zwei hellbraune Biber besetzt werden.
Für die Biber in der Mitte ist es egal, ob der dritte hellbraune Biber links und der dritte dunkelbraune Biber rechts steht oder umgekehrt, denn nach der mittleren Kreuzung schwimmt sowieso der hellbraune Biber nach links und der dunkelbraune Biber nach rechts.

Dies ist Informatik!

Das Kanalsystem der Biber stellt zusammen mit der Regel, wer links und wer rechts schwimmt, ein Teil eines *Sortiernetzes* dar. In einem Sortiernetz wandern Daten entlang einer Linie (die Kanäle in dieser Aufgabe) und bei jeder Verbindung (die Kreuzungen in dieser Aufgabe) wird geprüft, ob getauscht werden soll oder nicht. Einen dunklen Biber kann man sich dann beispielsweise als die Zahl 0 und einen hellen Biber als die Zahl 1 vorstellen. Als Sortiernetz sieht das dann so aus:



Ein vollständiges und minimales Sortiernetz für diese Aufgabe sähe übrigens so aus, man sieht gut wie der Teil eines Sortiernetzes aus dieser Aufgabe dort integriert ist:



Sortiernetze sind dann besonders effizient, wenn man die Vergleiche parallel zueinander ausführen kann. Dafür sind optimale Sortiernetze für grössere Datenmengen schwer zu finden.
Verallgemeinernd kann man sich das Kanalsystem der Biber auch als ein System von Kabeln in einem Computernetz wie dem Internet vorstellen. Hier stellen die Kanäle direkte Kabelverbindungen zwischen zwei Routern, den Kreuzungen, dar. In der Regel sind in solchen Routern feste Routing-Tabellen einprogrammiert, mit deren Hilfe die Datenpakete in Richtung ihres Zieles verschickt werden.

Stichwörter und Webseiten

Sortiernetz, Computernetze, Router, Routing-Tabelle

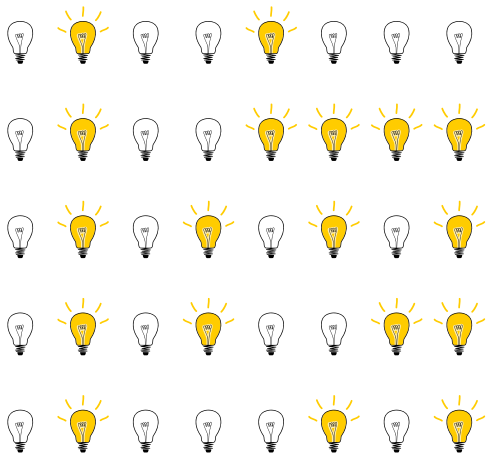
- `https://en.wikipedia.org/wiki/Sorting_network`
- `http://www.inf.fh-flensburg.de/lang/algorithmen/sortieren/networks/optimal/optimal-sorting-networks.htm`
- `https://www.computernetworkingnotes.com/ccna-study-guide/basic-routing-concepts-and-protocols-explained.html`
- `https://de.wikipedia.org/wiki/Routing%23Routing_im_Internet`
- `https://de.wikipedia.org/wiki/Routingtabelle`

5. Lichtsignale

Sina hat acht Lampen mit Schaltern und Kabeln verbunden. Sie kann damit Nachrichten senden. Sie nutzt dafür die folgende Code-Tabelle, in der `0` bedeutet, dass die entsprechende Lampe ausgeschaltet ist (💡) und `1`, dass die entsprechende Lampe eingeschaltet ist (💡):

A: 01000001	J: 01001010	S: 01010011
B: 01000010	K: 01001011	T: 01010100
C: 01000011	L: 01001100	U: 01010101
D: 01000100	M: 01001101	V: 01010110
E: 01000101	N: 01001110	W: 01010111
F: 01000110	O: 01001111	X: 01011000
G: 01000111	P: 01010000	Y: 01011001
H: 01001000	Q: 01010001	Z: 01011010
I: 01001001	R: 01010010	

Sina sendet nun die folgenden Lichtsignale:



Was bedeuten Sinas Lichtsignale?

A) `HOUSE`

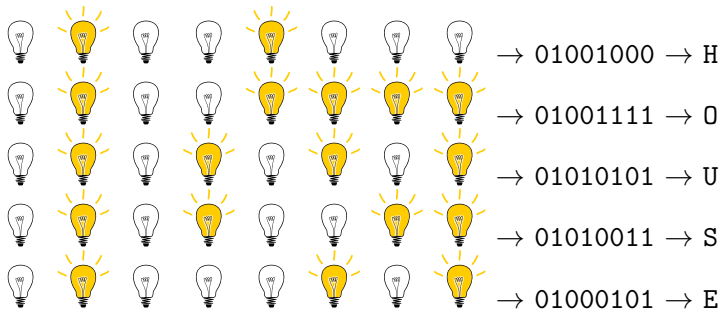
B) `HAPPY`

C) `HORSE`

D) `HONEY`

Lösung

Die Lichtsignale bedeuten:



Damit ist das Lösungswort A) `HOUSE` richtig.
Man kann diese Antwort übrigens ganz schnell finden: der mittlere Buchstabe ist in jedem Wort anders: A) `U`, B) `P`, C) `R` und D) `N`. Da das dritte Lichtsignal `U` bedeutet, kann nur noch die Antwort A) richtig sein.

Dies ist Informatik!

Die Codierung von Sina ist nicht zufällig gewählt. Sie nutzt einen Teil des sogenannten ASCII-Codes, der schon vor über fünfzig Jahren zum Austausch von Nachrichten entwickelt wurde. Er basiert auf dem Prinzip des Binärcodes, der bereits 1679 und 1703 Gottfried Leibnitz (1646–1716) auf der Basis indischer und chinesischer Vorläufersysteme für die Darstellung von Zahlen und das Rechnen mit diesen Zahlen beschrieben hatte. Claude Shannon (1916–2001) wendete diese dann auf die Entwicklung des Computers an.
Heute benutzten Computer Weiterentwicklungen des ASCII-Codes. Da der ASCII-Code lediglich 95 druckbare Zeichen enthielt (grosse und kleine lateinische Buchstaben, die Ziffern 0 bis 9 sowie ein paar Satzzeichen) und die restlichen 33 Zeichen Steuerzeichen (beispielsweise für Drucker) waren, brauchte man bald für Umlaute und andere Schriftsysteme Erweiterungen. Dies geschah zunächst in Form des ANSI-Codes und später im heute fast universell verwendeten Unicode. Dabei sind Sinas Buchstaben weiterhin genau so in der am weitesten verbreiteten Unicode-Variante UTF-8 codiert.
Der erste Block von Zeichen (identisch in ASCII, ANSI und Unicode) ist übrigens (Steuerzeichen sind leer gelassen, ␣ steht für das Leerzeichen):

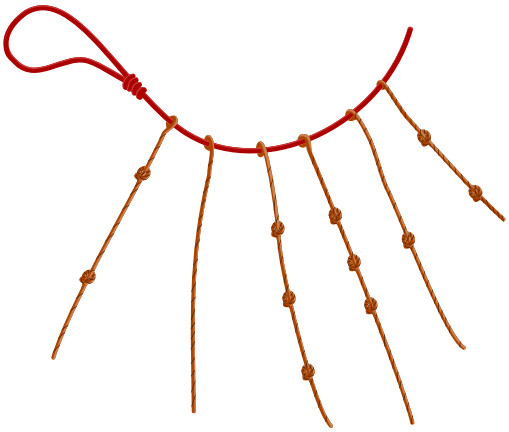
	000 ...	001 ...	010 ...	011 ...	100 ...	101 ...	110 ...	111 ...
...0000			␣	0	@	P	‘	p
...0001			!	1	A	Q	a	q
...0010			"	2	B	R	b	r
...0011			#	3	C	S	c	s
...0100			$	4	D	T	d	t
...0101			%	5	E	U	e	u
...0110			&	6	F	V	f	v
...0111			’	7	G	W	g	w
...1000			(	8	H	X	h	x
...1001			)	9	I	Y	i	y
...1010			*	:	J	Z	j	z
...1011			+	;	K	[	k	{
...1100			,	<	L	\	l	|
...1101			-	=	M	]	m	}
...1110			.	>	N	^	n	~
...1111			/	?	O	_	o	

Stichwörter und Webseiten

ASCII, Unicode, Codierung

- https://de.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange
- https://de.wikipedia.org/wiki/Bin%C3%A4rcode
- https://de.wikipedia.org/wiki/Gottfried_Wilhelm_Leibniz
- https://de.wikipedia.org/wiki/Claude_Shannon
- https://de.wikipedia.org/wiki/ANSI-Zeichencode
- https://de.wikipedia.org/wiki/Unicode
- https://de.wikipedia.org/wiki/UTF-8
- https://www.unicode.org/charts/PDF/U0000.pdf

6. Quipu



Die Inka nutzten früher Knoten zur Nachrichtenübermittlung. An einer Hauptschnur hingen weitere Nebenschnüre, an denen Knoten angebracht wurden. Diese sogenannten Quipus waren gross und aufwendig herzustellen.
Stell Dir vor, es soll eine vereinfachte Version der Quipus entwickelt werden. Die Bedingungen sind:

- An der Hauptschnur hängen immer gleich viele Nebenschnüre.
- Nebenschnüre unterscheiden sich lediglich durch die Anzahl der Knoten.
- Eine Nebenschnur hat 0, 1, 2 oder 3 Knoten.
- Die Reihenfolge der Nebenschnüre ist durch einen Knoten in der Hauptschnur festgelegt.
- Es sollen 30 eindeutig unterscheidbare Quipus für unterschiedliche Nachrichten möglich sein.

Wie viele Nebenschnüre hat die vereinfachte Version der Quipus mindestens unter diesen Bedingungen?

A) 2

B) 3

C) 4

D) 5

E) 8

F) 10

Lösung

Die Antwort B) 3 ist korrekt.
Jedes der Nebenschnüre kann einen von 4 verschiedenen Werten (0, 1, 2 oder 3) speichern. Bei zwei Schnüren hätte man $4 \cdot 4 = 16$ mögliche Kombinationen, bei drei Schnüren $4 \cdot 4 \cdot 4 = 64$ mögliche Kombinationen und so weiter. Damit genügen drei Nebenschnüre, mehr Nebenschnüre würden der Bedingung widersprechen, dass es möglichst wenige Nebenschnüre sein sollen. Da die Reihenfolge der Werte durch den Knoten in der Hauptschnur festgelegt ist, muss man auch nicht darauf achten, dass man die Schnur in der einen oder in der anderen Richtung lesen könnte.

Dies ist Informatik!

Quipus wurden tatsächlich von den Inka in Südamerika genutzt. Zur Buchhaltung und Steuererhebung wurden graue Quipus verwendet. Mit Hilfe gefärbter Schnüre konnten, so nimmt man an, bis zu 95 verschiedene Silben kodiert werden, und so konnte Schriftverkehr stattfinden. Im Gegensatz zu der einfachen Variante wie in dieser Aufgabe gab es zudem noch unterschiedliche Arten von Knoten und in einigen Fällen Unterschnüre, die an den Nebenschnüren angeknotet waren.
Das Beispiel der Aufgabe ist eine vereinfachte Variante. Da die Reihenfolge durch den Knoten in der Hauptschnur festgelegt ist, ergeben die einzelnen Werte (0, 1, 2 oder 3) ein *Stellenwertsystem*, in diesem Fall mit der Basis 4. Stellenwertsysteme sind weit verbreitet: in der Regel wird das 10er-Stellenwertsystem verwendet, der Computer nutzt das 2er-Stellenwertsystem (auch *Binärzahlen* genannt). In den Anfängen der Computer gab es auch Versuche, Computer zu bauen, die auf dem *Ternärsystem* mit der Basis 3 (dort als -1, 0 und $+1$ interpretiert) basieren. Mit einem Stellenwertsystem der Basis b kann man bei n Stellen genau b^n verschiedene Werte speichern. Ein Byte (8 Bits, die jeweils 0 oder 1 sein können) kann so $2^8 = 256$ verschiedene Werte speichern (von 0 bis 255), das Quipu dieser Aufgabe $4^3 = 64$ verschiedene Werte.
Für die Inka hätte zum Speichern der Werte von 1 bis 30 übrigens eine einzige Nebenschnur ausgereicht. Sie nutzten ebenfalls ein 10er-Stellenwertsystem wie wir beim Schreiben von Zahlen, einfach mit verschiedenen Knoten auf einer Schnur. So wäre die Einerstelle unter anderem mit einem mehrfach getörnten Überhandknoten und die Zehnerstelle durch die entsprechende Anzahl Stopperknoten kodiert worden wäre. Allerdings hätten sie dazu bis zu 4 Knoten gebraucht und dann auch noch verschiedenartige.

Stichwörter und Webseiten

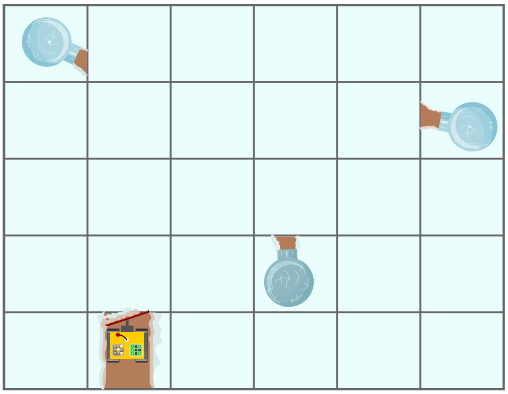
Quipu, Stellenwertsystem

- `https://de.wikipedia.org/wiki/Quipu`
- `https://de.wikipedia.org/wiki/Mehrfacher_%C3%9Cberhandknoten`
- `https://de.wikipedia.org/wiki/Stopperknoten`
- `https://de.wikipedia.org/wiki/Stellenwertsystem`
- `https://de.wikipedia.org/wiki/Tern%C3%A4rer_Computer`

7. Schneesturm

Nach einem heftigen Schneesturm sind überall Schneeverwehungen und die Bewohner der drei Iglus sind isoliert. Die Bewohner können aber mit Hilfe ihres ferngesteuerten Schneepflugs Wege räumen. Das funktioniert so:

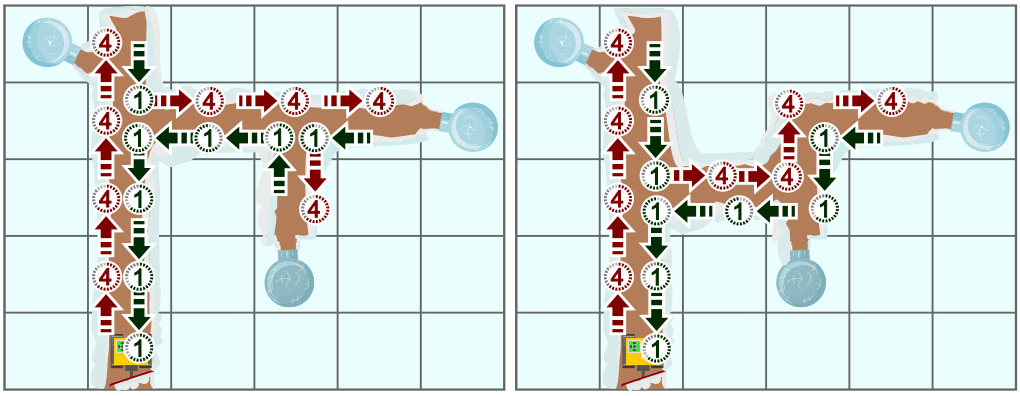
- Der Schneepflug braucht 4 Minuten, um von einem Quadrat auf ein benachbartes verschneites Quadrat zu fahren und es zu räumen.
- Der Schneepflug braucht 1 Minute, um von einem Quadrat auf ein benachbartes schneefreies Quadrat zu fahren.
- Benachbarte Quadrate sind immer nur die Quadrate auf der Karte, die direkt über, unter, links oder rechts von einem Quadrat liegen, der Schneepflug kann also nicht diagonal fahren.
- Sobald das Quadrat vor dem Eingang eines Iglus geräumt ist, können die Bewohner des Iglus den Eingang freischaufeln und sind nicht mehr isoliert.



Wie viele Minuten benötigt der Schneepflug im Idealfall, um alle Iglus von der Isolation zu befreien und zu seinem Ausgangsquadrat zurückzufahren?

Lösung

Die korrekte Antwort ist 40 Minuten. Die folgenden Graphiken zeigen die beiden optimalen Wege des Schneepfluges:



Warum geht es nicht schneller? Um das Iglu oben links zu erreichen müssen vier Quadrate geräumt werden. Das sind 16 Minuten. Um das Iglu rechts zu erreichen müssen weitere drei Quadrate geräumt werden. Das sind weitere 12 Minuten. Um das untere Iglu zu erreichen muss ein weiteres Quadrat geräumt werden, denn entweder muss ein Stichweg zum Querweg geräumt werden oder der Querweg muss geknickt werden. Das sind weitere 4 Minuten. Damit der Schneepflug wieder zurück kommt, muss er vier Quadrate zurück nach unten und drei Quadrate zurück nach links fahren. Das sind weitere 7 Minuten. Für den Umweg durch den Stichweg oder den geknickten Querweg benötigt er zusätzlich 1 Minute. Insgesamt braucht er also mindestens 40 Minuten.
Wenn der Schneepflug nun schneller räumen würde, wäre es eventuell effizienter, wenn er beim Rückweg vom unteren Iglu sich über das verschneite Quadrat links fahren und es dabei räumen würde. Aber das kostet ihn 4 Minuten für das Räumen und 1 Minute für das Weiterfahren auf das bereits schneefreie Quadrat, also 5 Minuten. Der Umweg über die bereits schneefreien Quadrate kostet ihn jedoch lediglich 4 Minuten.

Dies ist Informatik!

In dieser Aufgabe wird nach einem Wegenetz gesucht, das alle Orte (die Iglus und das Startquadrat des Schneepfluges) mit minimalen Kosten (die Zeit, die der Schneepflug braucht) verbindet. Solche Wegenetze enthalten nicht unbedingt die kürzesten Wege zwischen allen Knoten, dafür sind die Kosten, um ein solches Wegenetz zu erstellen, so klein wie möglich. Solche Wegenetze nennt man *Steinerbäume*. Sie werden zum Beispiel für das Erstellen von Computerplatinen oder das Erstellen von wenig genutzten Eisenbahnnetzen für Güter erstellt. Steinerbäume zu finden ist eines der schweren zeitaufwendigen Optimierungsprobleme der Informatik, so dass man häufig Algorithmen verwendet, die eine hinreichend gute Lösung finden, aber nicht unbedingt die beste.
Im Fall dieser Aufgabe werden die Kosten besonders berechnet, weil es nicht nur fixe Kosten für das Erstellen eines Weges berechnet werden (die 4 Minuten, um ein Quadrat freizuräumen), sondern auch die Kosten für das Zurückbewegen der Maschine vorkommen. Daher ist diese Aufgabe eine Verallgemeinerung des Steinerbaumproblems.

Stichwörter und Webseiten

Steinerbaumproblem

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

8. Schön, dass es Bäume gibt

Sergio hat ein Lied geschrieben, das beschreibt, wie aus einem Baum verschiedene Objekte entstehen können. Ein Vers lautet so:

> Schön, dass es Bäume gibt.
> An einem Baum wachsen Blätter,
> An einem Baum wachsen Blüten,
> Aus Blüten wachsen Früchte,
> Aus Blättern und Blüten kann ich Kränze winden.

Sergio ist es dabei wichtig gewesen, dass er nach der ersten Verszeile nur Objekte verwendet, die er vorher schon erwähnt hatte.
Welche der folgenden Verse ist für Sergio falsch?

A) Schön, dass es Bäume gibt.
An einem Baum wachsen Blüten,
An einem Baum wachsen Blätter,
Aus Blättern und Blüten kann ich Kränze winden,
Aus Blüten wachsen Früchte.

B) Schön, dass es Bäume gibt.
An einem Baum wachsen Blüten,
An einem Baum wachsen Blätter,
Aus Blüten wachsen Früchte,
Aus Blättern und Blüten kann ich Kränze winden.

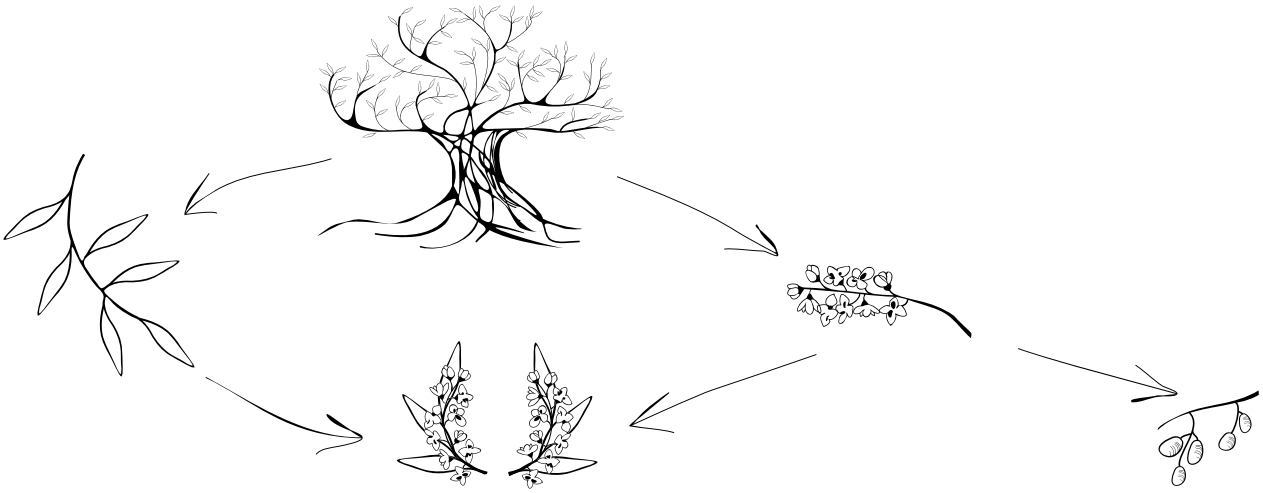
C) Schön, dass es Bäume gibt.
An einem Baum wachsen Blätter,
Aus Blüten wachsen Früchte,
An einem Baum wachsen Blüten,
Aus Blättern und Blüten kann ich Kränze winden.

D) Schön, dass es Bäume gibt.
An einem Baum wachsen Blüten,
Aus Blüten wachsen Früchte,
An einem Baum wachsen Blätter,
Aus Blättern und Blüten kann ich Kränze winden.

E) Schön, dass es Bäume gibt.
An einem Baum wachsen Blätter,
An einem Baum wachsen Blüten,
Aus Blättern und Blüten kann ich Kränze winden,
Aus Blüten wachsen Früchte.

Lösung

Man kann die Abhängigkeiten der Objekte „Baum“, „Blätter“, „Blüten“, „Kränze“ und „Früchte“ mit Hilfe eines Graphen beschreiben, wobei ein Pfeil bedeutet, dass das eine für das andere benötigt wird:



Demnach müssen vor Früchten Blüten genannt sein und vor Kränzen Blätter und Blüten.
Die Antworten haben die folgenden Reihenfolgen der Objekte:

A) Baum, Blüten, Blätter, Kränze, Früchte

B) Baum, Blüten, Blätter, Früchte, Kränze

C) Baum, Blätter, *Früchte*, *Blüten*, Kränze

D) Baum, Blüten, Früchte, Blätter, Kränze

E) Baum, Blätter, Blüten, Kränze, Früchte

Bei der Antwort C) werden Früchte vor Blüten besungen (oben hervorgehoben), was ein Widerspruch ist, da es für Früchte Blüten braucht. In allen anderen Versen sind die Bedingungen eingehalten.

Dies ist Informatik!

1974 schrieb der italienische Musiker Sergio Endrigo (1933–2005) das Kinderlied „Ci vuole un fiore“ nach einem Text von Gianni Rodari (1920–1980). In diesem Lied besingt er, wenn man einen Tisch will, braucht man zunächst Holz, für Holz einen Baum, für einen Baum einen Samen, für einen Samen eine Frucht und für eine Frucht eine Blume. Er beschreibt auch, dass man für eine Blume über den Umweg eines Astes, eines Baumes, eines Waldes, eines Berges und der Erde ebenfalls eine Blume braucht. Er endet damit, dass man für alles letztlich eine Blume braucht.
Ein Vorrang eines Objekts von einem anderen kann man mit Hilfe eines *gerichteten Graphen* beschreiben. Ein solcher Graph ist in der Antworterklärung abgedruckt. Er ist ein *gerichteter azyklischer Graph*, der eine *Menge zulässiger Reihenfolgen* beschreibt. Wenn man einen Knoten (eines der Objekte) möchte, muss man alle Objekte bereits haben, die auf ihn zeigen. Dasselbe gilt wiederum für diese Objekte, so dass man *rekursiv* zurückgehen muss, bis man bei Objekten landet, auf die keine Pfeile zeigen. Diese kann man als Startobjekte verwenden.
Das Lied von Sergio Endrigo lässt sich übrigens nicht mit Hilfe eines gerichteten azyklischen Graphen beschreiben. Im oben beschriebenen zweiten Teil besingt er, dass man für eine Blüte letztlich eine

Blüte braucht. Das ist ein Widerspruch dazu, dass der Graph azyklisch sein muss, also keine Kreisschlüsse beinhalten dürfen. Durch diesen Logikbruch macht er jedoch seine Aussage umso klarer: „Ci vuole un fiore“ – „Wir brauchen eine Blume“!

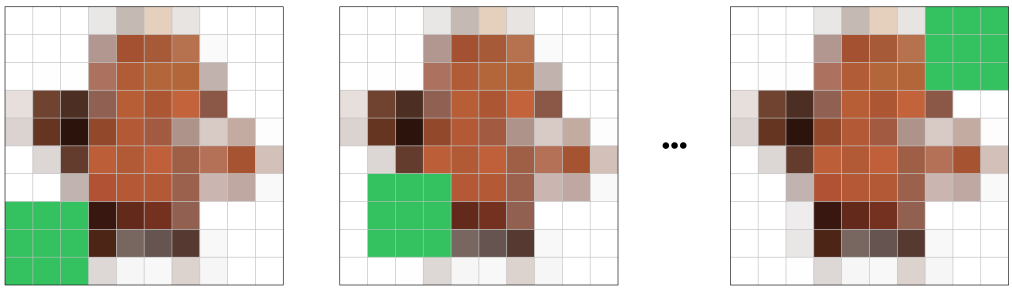
Stichwörter und Webseiten

Gerichtete azyklische Graphen, topologische Sortierung

- `https://www.filastrocche.it/contenuti/ci-vuole-un-fiore/`
- `https://www.youtube.com/watch?v=9ht4tIot8XY`
- `https://en.wikipedia.org/wiki/Precedence_graph`
- `https://de.wikipedia.org/wiki/Topologische_Sortierung`

9. Videokompression

Videos benötigen viel Speicherplatz. Gleichzeitig sind sich jedoch zwei aufeinanderfolgende Standbilder eines Videos häufig sehr ähnlich.
Das folgende Video ist 10×10 Bildpunkte gross. Das grüne Quadrat in der unteren linken Ecke ist 3×3 Bildpunkte gross. Es bewegt sich von Standbild zu Standbild um jeweils einen Bildpunkt nach rechts und nach oben, bis es am Ende in der oberen rechten Ecke landet.

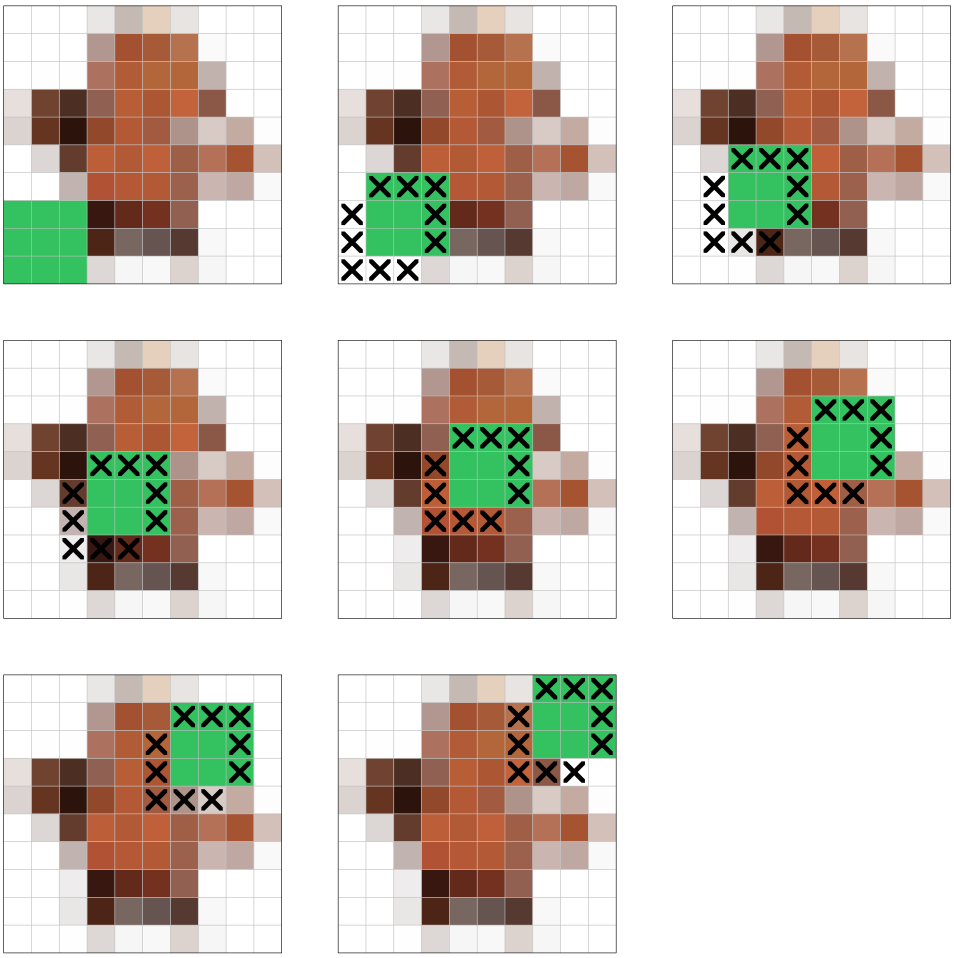


Um Speicherplatz zu sparen werden ab dem zweiten Standbild lediglich die Bildpunkte, die sich geändert haben, gespeichert.
Wie viele Bildpunkte müssen für das gesamte Video gespeichert werden?

A) 100
B) 135
C) 140
D) 170
E) 180
F) 700
G) 800
H) 1000

Lösung

Die einzelnen Standbilder des Videos sehen so aus, wenn man die jeweils geänderten Bildpunkte markiert:



Zuerst stellt man fest, dass das erste Standbild $10 \cdot 10 = 100$ Bildpunkte enthält. Für jedes weitere Standbild müssen lediglich die geänderten Bildpunkte gespeichert werden. Das sind die 5 Bildpunkte unten links vom Quadrat, die durch die Bildpunkte des Hintergrunds ersetzt werden, sowie die fünf Bildpunkte oben rechts im Quadrat, die neu das Quadrat darstellen. Pro Standbild werden so 10 Bildpunkte geändert. Das Quadrat braucht weitere 7 Standbilder um sich von unten links nach oben rechts zu bewegen, also müssen $10 \cdot 7 = 70$ Bildpunkte für die geänderten Bildpunkte zu den 100 ursprünglichen Bildpunkten hinzugefügt werden, so dass die Antwort D) 170 korrekt ist.

Dies ist Informatik!

Wie bereits in der Aufgabe beschrieben, spielt Videokompression eine grosse Rolle heutzutage. Dabei ist das beschriebene Verfahren nur eines von mehreren Ansätzen, wie man Videos komprimieren kann. Ein weiterer Ansatz ist, bestimmte Informationen wegzulassen, die vom Menschen eher nicht wahrgenommen werden. Das Bildformat JPEG nutzt solche Zusammenhänge aus. Bei besonders stark komprimierten Bildern erkennt man das durch Blockbildung, weil für einen solchen Block die Ähnlichkeit der Farbe fälschlicherweise als nicht-wahrnehmbar interpretiert wurde. Weitere Möglichkeiten sind das Verringern des Farbraumes.

Auf diesen Ideen basiert der MPEG-Standard. Er unterscheidet wie in dieser Aufgabe zwischen unterschiedlichen Standbildtypen. Eine Sorte von Standbildern (sogenannte „Intra-Bilder“) stellen ein vollständiges Standbild dar (ähnlich wie unser erstes Standbild). Eine andere Sorte von Standbildern basieren auf vorhergehenden Standbildern („P-Bilder“, wie unsere weiteren Standbilder) oder sogar zusätzlich nachfolgenden Standbildern („D-Bilder“, kommt nicht in dieser Aufgabe vor). Um den Pufferaufwand gering zu halten und bei Übertragungsfehlern wieder „einsteigen“ zu können, werden in regelmässigen Abständen Intra-Bilder eingefügt. Bei besonders stark komprimierten Videos erkennt man P-Bilder und D-Bilder, wenn ein lichtschwacher Hintergrund plötzlich „springt“, obwohl die Szene sich über einige Zeit hinweg nur langsam bewegt hat.
Der Speicherbedarf ist übrigens nicht ganz so toll wie in der Aufgabe nahegelegt: zusätzlich zu den Farbwerten muss auch noch der Ort der veränderten Pixel gespeichert werden. Das gibt vielleicht einen Faktor 2 für den Speicherbedarf eines veränderten Pixels. Aber selbst dann wären 240 Speichereinheiten gegenüber 800 Speichereinheiten immer noch eine beeindruckende Platzersparnis, vor allem weil das in der Aufgabe beschriebene Verfahren im Gegensatz zu MPEG verlustfrei ist!

Stichwörter und Webseiten

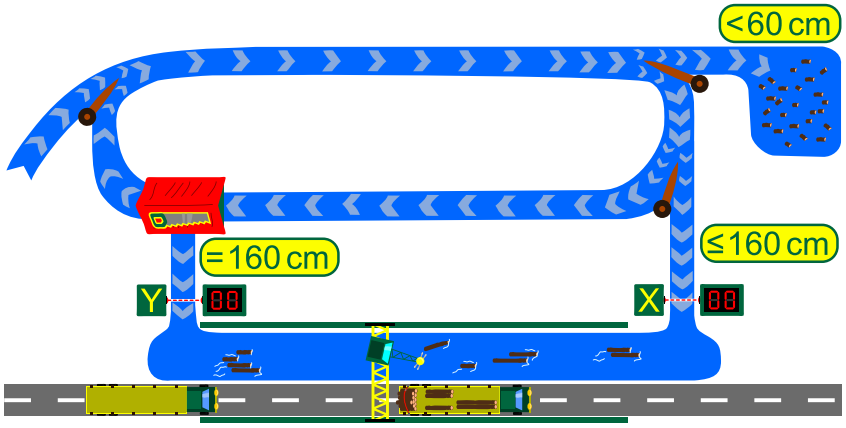
Videokompression

- `https://de.wikipedia.org/wiki/Videokompression`
- `https://de.wikipedia.org/wiki/JPEG`
- `https://de.wikipedia.org/wiki/MPEG-1`

10. Sägerei

In einer Sägerei werden Baumstämme auf Längen zwischen 60 cm und 160 cm gekürzt und dann auf Lastwagen verladen. Innerhalb der Sägerei werden die Baumstämme durch Kanäle transportiert. Zusätzlich gibt es folgende Verarbeitungspositionen:

- Oben links werden Baumstämme angeliefert.
- Oben rechts werden alle Baumstämme aussortiert, die kürzer als 60 cm sind (<60 cm).
- In der Mitte rechts werden alle Baumstämme, die 160 cm oder kürzer sind, auf Lastwagen verladen (≤160 cm). Diese werden beim Sensor X gezählt.
- In der Mitte links wird von allen Baumstämmen ein 160 cm langes Stück abgesägt. Das abgesägte Stück wird auf Lastwagen verladen (=160 cm) und beim Sensor Y gezählt. Das Reststück wird wieder in den Kreislauf gegeben.



Es werden drei Baumstämme mit den Längen 60 cm, 140 cm und 360 cm angeliefert und von der Sägerei verarbeitet.
Wie viele Baumstämme werden vom Sensor X und wie viele Baumstämme werden vom Sensor Y gezählt?

A) Sensor X: keine Stämme, Sensor Y: 4 Stämme

B) Sensor X: 1 Stamm, Sensor Y: 3 Stämme

C) Sensor X: 2 Stämme, Sensor Y: 2 Stämme

D) Sensor X: 3 Stämme, Sensor Y: 1 Stamm

Lösung

Der 60 cm lange Baumstamm wird oben rechts nicht aussortiert, da er nicht kürzer als 60 cm ist. Er wird aber in der Mitte rechts auf Lastwagen verladen, weil er 160 cm oder kürzer ist. Somit ist beim Sensor X bereits ein Stamm gezählt worden.
Der 140 cm lange Baumstamm wird oben rechts ebenfalls nicht aussortiert, da er nicht kürzer als 60 cm ist. Er wird aber in der Mitte rechts auf den Lastwagen verladen, weil er 160 cm oder kürzer ist. Somit ist beim Sensor X ein zweiter Stamm gezählt worden.
Der 360 cm lange Baumstamm wird oben rechts auch nicht aussortiert, da er nicht kürzer als 60 cm ist. In der Mitte rechts wird er aber zur Säge weitergeleitet, da er länger als 160 cm ist. In der Säge wird von ihm ein 160 cm langes Stück abgesägt und auf Lastwagen verladen. Damit ist beim Sensor Y ein Stamm gezählt worden. Das 200 cm lange Reststück wird wieder in den Kreislauf gegeben. Der nun 200 cm lange Baumstamm wird oben rechts nicht aussortiert, da er nicht kürzer als 60 cm ist. In der Mitte rechts wird er wieder zur Säge weitergeleitet, da er länger als 160 cm ist. In der Säge wird ihm ein zweites 160 cm langes Stück abgesägt und auf Lastwagen verladen. Damit ist beim Sensor Y ein zweiter Stamm gezählt worden. Das 40 cm lange Reststück wird wieder in den Kreislauf gegeben. Der nun 40 cm lange Baumstamm wird oben rechts aussortiert.
Damit ist die korrekte Antwort C) Sensor X: 2 Stämme, Sensor Y: 2 Stämme.

Dies ist Informatik!

Von den Baumstämmen im Kreislauf ist alleine die Länge wichtig. Man kann die Sägerei also als Programm ansehen, in das ganze Zahlen eingegeben werden und in dem bestimmte Messungen gemacht werden. Damit kann man die Sägerei als *reaktives Programm* ansehen: während die Zahl vom Programm verarbeitet wird, verändern sich mit der Zeit die Messungen. Man findet reaktive Programmierung übrigens prominent in Tabellenkalkulationsprogrammen. Die mit Hilfe von Formeln berechneten Werte in Tabellen reagieren auf Änderungen von Werten in anderen Zellen.
Konkret werden hierzu mehrere reaktive Operationen verwendet: oben links werden zwei Datenströme vereinigt (engl. *merge*), oben rechts gefiltert (engl. *filter*), in der Mitte rechts ebenfalls gefiltert und in der Mitte links verändert (engl. *transform*). Die beiden Sensoren stellen Messungen an (engl. *scan*).
Dynamische Prozesse wie in dieser Aufgabe zu analysieren ist Kerngeschäft von Informatik. Schon lange bevor der Begriff *Computational Thinking* aufkam und durch Jeanette Wing 2006 weltberühmt gemacht wurde, wurden Begriffe wie *Procedural Thinking* oder *Algorithmic Thinking* als besonderes Paradigma der Informatik verwendet.

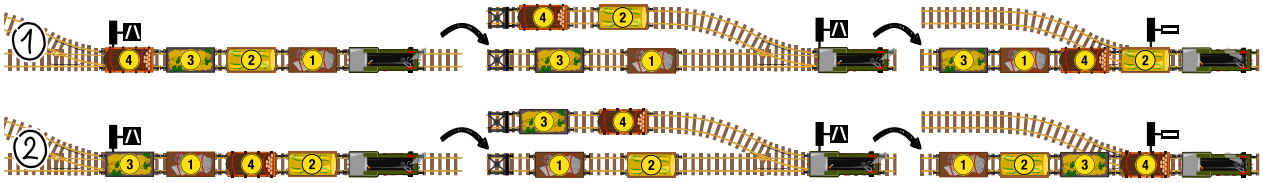
Stichwörter und Webseiten

Reaktive Programmierung

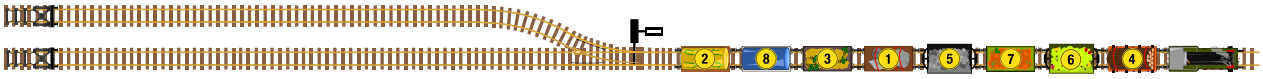
- `https://de.wikipedia.org/wiki/Reaktive_Programmierung`
- `https://en.wikipedia.org/wiki/Computational_thinking`
- `https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf`

11. Rangierbahnhof

Ein Güterzug soll einzelne Güterwagen an Anschlussgleise entlang der Hauptstrecke abliefern. Um Zeit zu sparen und Rangieren auf der Hauptstrecke zu vermeiden, sollen die Güterwagen im Rangierbahnhof den Zahlen nach sortiert werden, so dass ganz links der Güterwagen mit der Nummer 1  steht.
Im Rangierbahnhof gibt es einen Ablaufberg über den die Güterwagen von rechts nach links abgedrückt werden. Am Ablaufberg wird für jeden Güterwagen einzeln entschieden, in welches der beiden Abstellgleise er rollt. Danach zieht die Lokomotive die Güterwagen wieder heraus: zuerst alle aus dem einen und dann alle aus dem anderen Abstellgleis. Dieser Vorgang wird als ein Abdrückvorgang bezeichnet.
Wenn zum Beispiel vier Güterwagen sortiert werden sollen, genügen zwei Abdrückvorgänge (Schritt ① und Schritt ②):



Es ist nicht möglich, die vier Güterwagen in einem Abdrückvorgang zu sortieren.
Wenn die Güterwagen in der Reihenfolge 2 – 8 – 3 – 1 – 5 – 7 – 6 – 4 *stehen, wie viele Abdrückvorgänge braucht es mindestens, damit der Güterzug sortiert ist?*



A) 3

B) 4

C) 5

D) 6

E) 7

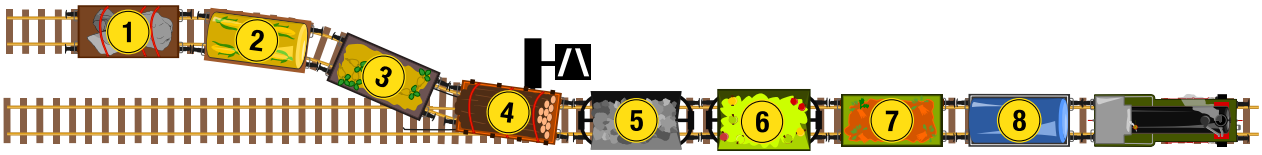
F) 8

Lösung

Die richtige Antwort ist, dass es A) 3 Abdrückvorgänge braucht.
Natürlich kann man Güterwagen mit den verschiedensten Methoden sortieren, aber eine der besten ist, zunächst die Güterwagen 1, 3, 5 und 7 in das obere Gleis und die Güterwagen 2, 4, 6 und 8 in das untere Gleis abzudrücken, und zunächst die Güterwagen aus dem unteren Gleis und dann die aus dem oberen Gleis herauszuziehen:

Damit sind von einem Zweierpaar (1 und 2, 3 und 4, 5 und 6, 7 und 8) von Güterwagen immer der mit der kleineren Nummer links von dem mit der grösseren Nummer.
Als nächstes macht es Sinn, die Güterwagen 1, 2, 5 und 6 in das obere Gleis und die Güterwagen 3, 4, 7 und 8 in das untere Gleis abzudrücken, und zunächst die Güterwagen aus dem unteren Gleis und dann die aus dem oberen Gleis herauszuziehen:

Damit wurde die Sortierung der Zweierpaare von vorher nicht verändert, weil ein Zweierpaar immer in dasselbe Gleis abgedrückt wurde. Zudem sind aber nun die Güterwagen 1 bis 4 und 5 bis 8 relativ zueinander sortiert, die beiden Gruppen sind aber noch gemischt.
Zuletzt braucht man nur noch die Güterwagen 1 bis 4 in das obere und die Güterwagen 5 bis 8 in das untere Gleis abzudrücken und zunächst die Güterwagen aus dem unteren Gleis und dann die aus dem oberen Gleis herauszuziehen:



Die Sortierung der beiden Gruppen wurde nicht verändert, weil alle Güterwagen der Gruppe 1 bis 4 in ein Gleis und alle Güterwagen der Gruppe 5 bis 8 in das andere Gleis abgedrückt wurden. Nun bestehen beide Gruppen aus sortierten Güterwagen und in der einen Gruppe haben alle Güterwagen eine kleinere Nummer als die Güterwagen in der anderen Gruppe.
Schneller kann man die acht Güterwagen nicht sortieren. Ein vollständiger Beweis hierfür wäre an dieser Stelle zu aufwendig, aber die grundlegende Idee ist diese: In einem Abdrückvorgang kann man lediglich die Reihenfolge von einer Teilmenge relativ zu der anderen Teilmenge verändern, aber nicht innerhalb der jeweiligen Teilmenge. Damit können im ersten Abdrückvorgang maximal zwei Güterwagen in ungünstiger Reihenfolge sortiert werden. Jeder weitere Abdrückvorgang verdoppelt die Anzahl der Güterwagen in ungünstiger Reihenfolge, die sortiert werden können. Die acht Güterwagen in der Aufgabe sind so gewählt, dass eine ungünstige Reihenfolge vorliegt, also genügen zwei Abdrückvorgänge nicht.

Dies ist Informatik!

Eisenbahner auf der ganzen Welt müssen so eine Aufgabe tagtäglich lösen, denn Güterwagen zu rangieren ist eine aufwendige und arbeitsintensive Arbeit: jedes Mal müssen Güterwagen gekuppelt und entkuppelt werden, was immer noch eine manuelle Arbeit ist. Dies kostet Zeit und blockiert die Hauptstrecke, insbesondere wenn einige Güterwagen auf der Hauptstrecke gesichert und abgekoppelt werden müssen. Daher haben Eisenbahner schon sehr früh grosse Rangierbahnhöfe mit vielen Abstellgleisen entwickelt. In der Schweiz gibt es Rangierbahnhöfe in Muttenz bei Basel, in Buchs SG, zwischen Spreitenbach und Dietikon bei Zürich, in Denges bei Lausanne und in Chiasso. In dieser Aufgabe hat der Rangierbahnhof lediglich zwei Abstellgleise, eine Herausforderung für grosse Güterzüge aber eine typische Situation für Nebenbahnen und insbesondere für Schmalspurbahnen, die keine direkte Verbindung zu grösseren Bahngesellschaften haben.
Damit Güterzüge effizient sortiert werden können, kann die Informatik viel helfen. In diesem Fall vereinfacht das Prinzip, dieselbe Aufgabe wieder und wieder zu lösen, die Aufgabe stark: eine Methode, die in der Informatik unter der Bezeichnung „Teile und Herrsche“ (engl. *divide & conquer*) bekannt ist. In diesem Fall werden erst jeweils zwei Güterwagen, dann jeweils vier Güterwagen und dann acht Güterwagen sortiert.
Die Abstellgleise für die Güterwagen funktionieren wie der abstrakte Datentyp *Stapel*, der intensiv in der Informatik eingesetzt wird. Die einzigen erlaubten Operationen sind: *das oberste Element entfernen* (engl. *pop*) und *oben ein Element hinzufügen* (engl. *push*). Manchmal kann man bei Stapeln auch noch *das oberste Element* anschauen (engl. *top*) und schauen, *ob der Stapel leer ist* (engl. *empty*).

Stichwörter und Webseiten

Teile und Herrsche (Divide & Conquer), Stapel

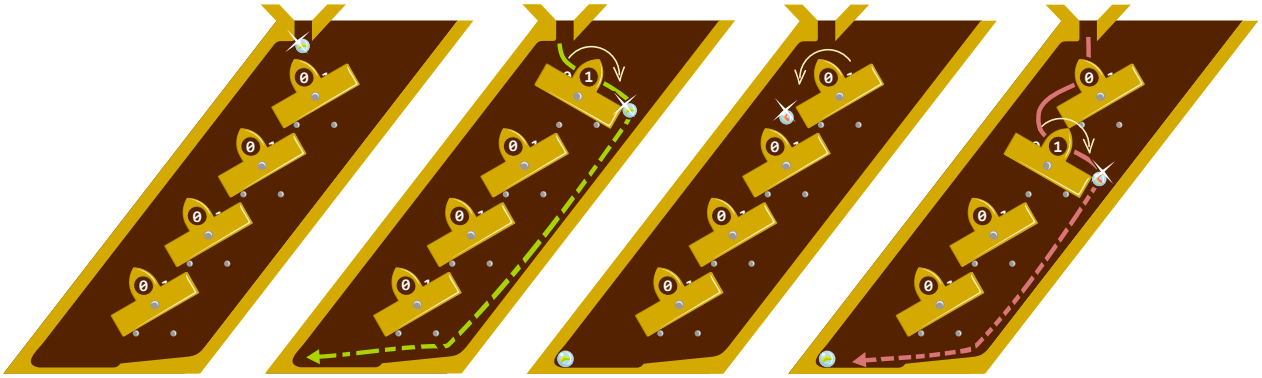
- `https://de.wikipedia.org/wiki/Rangierbahnhof`
- `https://de.wikipedia.org/wiki/Teile-und-herrsche-Verfahren`
- `https://de.wikipedia.org/wiki/Stapelspeicher`

12. Kugelbahn

Eine Kugelbahn enthält vier Wippen, die in zwei Neigungen stehen können:

- Ist die Wippe nach links geneigt, steht sie in der Neigung 0.
- Ist die Wippe nach rechts geneigt, steht sie in der Neigung 1.

Wenn eine Kugel auf eine Wippe trifft, ändert die Wippe ihre Neigung und die Kugel rollt herunter. Beim Herunterlassen von zwei Kugeln kippen die Wippen so, dass nach der ersten Kugel die oberste Wippe nun in der Neigung 1 steht, und dass nach der zweiten Kugel die oberste Wippe wieder zurück in der Neigung 0 steht und die zweitoberste Wippe in der Neigung 1 steht:



Am Ende sind die Wippen (von links unten nach rechts oben gelesen) in den Neigungen 0, 0, 1 und 0.
Alle Wippen werden wieder auf 0 *gestellt. Wie werden die Wippen (von links unten nach rechts oben gelesen) stehen, wenn zehn Kugeln durch die Kugelbahn rollen?*

A) 0, 0, 0 und 0
B) 1, 0, 0 und 0
C) 0, 1, 0 und 0
D) 0, 0, 1 und 0
E) 0, 0, 0 und 1
F) 1, 1, 0 und 0
G) 1, 0, 1 und 0
H) 1, 0, 0 und 1
I) 0, 1, 1 und 0
J) 0, 1, 0 und 1
K) 0, 0, 1 und 1
L) 1, 1, 1 und 0
M) 1, 1, 0 und 1
N) 1, 0, 1 und 1
O) 0, 1, 1 und 1
P) 1, 1, 1 und 1

Lösung

Die oberste Wippe ändert ihre Neigung bei jeder Kugel. Wenn also wie in diesem Fall eine gerade Anzahl von Kugeln (10) durch die Kugelbahn rollt, ist die Neigung der obersten Wippe wieder auf 0. Damit ist die letzte Ziffer eine 0.
Die zweitoberste Wippe ändert ihre Neigung nur, wenn die oberste Wippe vorher in der Neigung 1 war und die Kugel nach links weggerollt war. Das ist nur jedes zweite Mal der Fall. Sie ändert ihre Neigung also fünfmal und steht am Ende auf 1. Damit ist die vorletzte Ziffer eine 1.
Die drittoberste Wippe ändert ihre Neigung nur, wenn die zweitoberste Wippe vorher in der Neigung 1 war und überhaupt eine Kugel auf sie trifft, wenn also auch die oberste Wippe vorher in der Neigung 1 war. Damit ändert die drittoberste Wippe ihre Neigung nur jedes vierte Mal. Das ist bei der vierten und bei der achten Kugel der Fall, also zweimal insgesamt. Damit steht sie am Ende auf 0 und die drittletzte Ziffer ist eine 0.
Die unterste Wippe ändert ihre Neigung nur, wenn alle drei Wippen über ihr vorher in der Neigung 1 waren und sie damit jeweils von einer Kugel getroffen werden. Das ist nur bei der achten Kugel der Fall, die unterste Wippe ändert ihre Neigung also nur einmal. Damit ist die erste Ziffer eine 1.
Insgesamt ist die Neigung der Wippen 1, 0, 1 und 0.

Dies ist Informatik!

Die Wippen der Kugelbahn stellen ein elektronisches Schaltelement dar, das zwischen zwei Zuständen hin- und herschalten kann. Solche Schaltelemente sind Grundbausteine elektronischer Geräte, in diesem Fall ist es eine Variante eines *Flipflops*.
Die Wippen funktionieren zusammen wie ein (binäres) *Zählwerk*. Hierbei sind die Flipflops je als *Halbaddierer* verbunden. Ein solcher Halbaddierer nimmt als Eingabe zum einen den gespeicherten Zustand des Flipflops sowie einen Impuls. Als Ausgabe hat es einen neuen zu speichernden Zustand und einen Übertrag.

Gespeicherter Zustand	Impuls	Zu speichernder Zustand	Übertrag
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Die Schaltung muss sicherstellen, dass die Änderung der Speicherzustände in der richtigen Reihenfolge stattfindet und sich nach einer Änderung nicht sofort wieder selbsttätig ändern.
Die Zustände der Wippen repräsentieren also letztlich eine Binärzahl, die mit jeder Kugel um 1 erhöht wird. Zehn Kugeln ergeben also mit dem Ausgangszustand die ersten 11 Binärzahlen: 0000 → 0001 → 0010 → 0011 → 0100 → 0101 → 0110 → 0111 → 1000 → 1001 → 1010.

Stichwörter und Webseiten

Binäres Zahlensystem, Zählwerk, Halbaddierer, Flipflop

- `https://de.wikipedia.org/wiki/Z%C3%A4hlwerk`
- `https://www.youtube.com/watch?v=zELAfmp3fXY`
- `https://de.wikipedia.org/wiki/Synchronz%C3%A4hler`
- `https://elektroniktutor.de/digitaltechnik/synchron.html`

- `https://de.wikipedia.org/wiki/Flipflop`
- `https://de.wikipedia.org/wiki/Halbaddierer`
- `https://de.wikipedia.org/wiki/Dualsystem`

13. Vier Fische

In der Informatik wird die Funktionsweise von Operatoren wie `+` oder `*` teilweise davon abhängig gemacht, was für Datentypen involviert sind. Die folgende Tabelle zeigt verschiedene typische Kombinationen für Ausdrücke:

Allgemein	Beispiel
Zahl + Zahl → Zahl (Addieren)	`2+3` → `5`
Zahl + Text → Fehler	`2+"3"` → Fehler
Text + Zahl → Fehler	`"2"+3` → Fehler
Text + Text → Text (Aneinanderketten)	`"2"+"3"` → `"23"`
Zahl * Zahl → Zahl (Multiplizieren)	`2*3` → `6`
Zahl * Text → Text (Zahl mal den Text aneinanderketten)	`2*"3"` → `"33"`
Text * Zahl → Text (den Text Zahl mal aneinanderketten)	`"2"*3` → `"222"`
Text * Text → Fehler	`"2"*"3"` → Fehler

Wenn als Ergebnis „Fehler“ steht, bedeutet das, dass für diese Kombination keine Funktionsweise definiert ist. Wenn in einem Ausdruck ein Fehler vorkommt, ist das Gesamtergebnis ebenfalls ein Fehler.
Bei der Kombination von Operatoren gilt die „Punkt- vor Strichregel“: der Operator `*` wird vor dem Operator `+` ausgeführt. Mit Klammern kann dieses anders gesteuert werden. Die Klammern werden von innen nach aussen ausgeführt.
Welche der folgenden Ausdrücke erzeugt die folgende Textzeile?

```
"...><(((°>.....><(((°>.....><(((°>.....><(((°>..."
```

A) `(3*"."+"><"+3*"("+"°>"+3*".")*"2"*2`

B) `(3*"."+"><"+3*"("+"°>")*2*2+3*"."`

C) `(3*"."+"><"+"3"*"("+"°>"+3*".")*2*2`

D) `(3*"."+"><"+3*"("+"°>"+3*".")*2*2`

Lösung

Die richtige Antwort ist D) `(3*"."+"><"+3*"("+"°>"+3*".")*2*2`.
Die fünf „Summanden“ in der Klammer ergeben jeweils:

- `3*"."` → `"..."`
- `"><"` → `"><"`
- `3*"("` → `"((("`
- `"°>"` → `"°>"`
- `3*"."` → `"..."`

Aneinander gekettet ergibt das `"...><(((°>..."`.
Die beiden darauffolgenden Multiplikationen `*2*2` bewirken jeweils, dass der Text zweimal aneinander gehängt wird `"...><(((°>.....><(((°>..."` und letztlich `"...><(((°>.....><(((°>.....><(((°>.....><(((°>..."` ergibt.
Die Antwort A) produziert einen Fehler, denn das Ergebnis der Klammer ist ein Text (`"...><(((°>..."`), der mit einem weiteren Text (`"2"`) multipliziert werden soll.
Die Antwort B) produziert den Text `"...><(((°>...><(((°>...><(((°>...><(((°>..."`. Dies ist zwar sehr nahe am gewünschten Resultat, jedoch stimmen die Anzahl der Punkte zwischen den Fischen nicht mit dem gewünschten Resultat überein.
Die Antwort C) produziert einen Fehler, denn der Ausdruck `"3"*"("` führt bereits zu dem Fehler.

Dies ist Informatik!

Werden Operatoren (oder auch Unterprogramme) in Abhängigkeit von den Operanden (oder auch den Parametern) unterschiedlichen Funktionsweisen zugeordnet, so nennt man das *Überladen*. Vor allem bei Operatoren, die auch ausserhalb der Programmiersprache häufig verwendet werden, kommt dies oft vor. Die oben beschriebene Art des Überladen der Operatoren `+` und `*` kommt in vielen Programmiersprachen vor. So praktisch das Überladen von Operatoren ist, so birgt es die Gefahr, dass die Lesbarkeit des Programmcodes darunter leidet.
Das folgende kleine Programm erzeugt die gleiche Zeichenkette aus der Aufgabe (die Fische), aber welche der Variablen von welchem Typ ist und welcher Operator demnach wie funktioniert, muss mühsam überprüft werden:

```
a ← 3
b ← "."
c ← "><"
d ← "("
e ← "°>"
f ← 2
Ausgabe (a*b+c+a*d+e+a*b)*f*f
```

Das Überladen von Begriffen mit verschiedenen Bedeutungen ist übrigens kein reines Phänomen aus der Informatik. In der Sprache nennt man dies *Polysemie*. Das Wort „Läufer“ beispielsweise hat unterschiedliche Bedeutungen im Kontext von Sport, Schach, Teppichen, ...
Der erzeugte Fisch ist übrigens ein klassisches Beispiel für ASCII-Art. Er ist eine Variante des „roten Herings“, der im Internet manchmal Trollen zugeworfen wird.

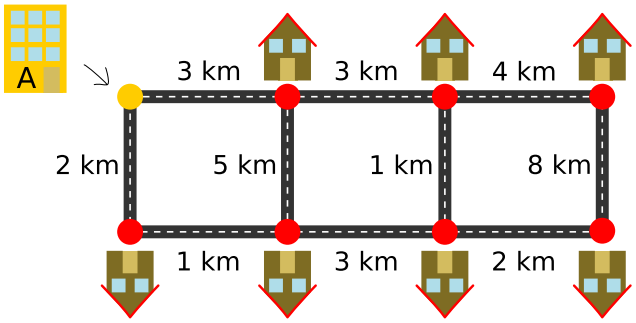
Stichwörter und Webseiten

Überladen, ASCII-Art

- `https://de.wikipedia.org/wiki/%C3%9Cberladen`
- `https://de.wikipedia.org/wiki/Polysemie`
- `https://de.wikipedia.org/wiki/Teekesselchen`
- `https://de.wikipedia.org/wiki/ASCII-Art`

14. Ferienjob

Für einen Ferienjob lieferst Du mit dem Velo Pakete aus. Du startest am Ort A und lieferst an allen sieben weiteren Orten jeweils ein Paket aus. Am letzten Ort ist Deine Tour beendet und Dein Arbeitgeber holt Dich mitsamt Deinem Velo ab.
Um fit zu bleiben, möchtest Du eine möglichst grosse Gesamtlänge der Wege mit den Paketen zurückgelegt haben. Für jeden Weg ist die Einzellänge in der Karte unten notiert. Dein Arbeitgeber lässt Dir freie Wahl, welche Wege Du nimmst, Du möchtest aber keinen Ort doppelt anfahren.



Wie lang ist der Weg mit der grössten Gesamtlänge ohne einen Ort doppelt anzufahren?

A) 22 km

B) 23 km

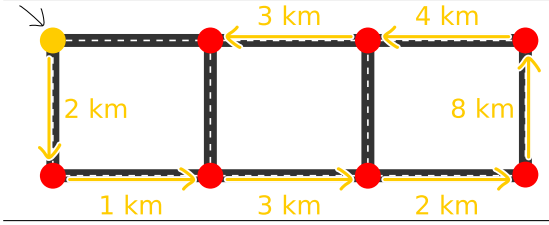
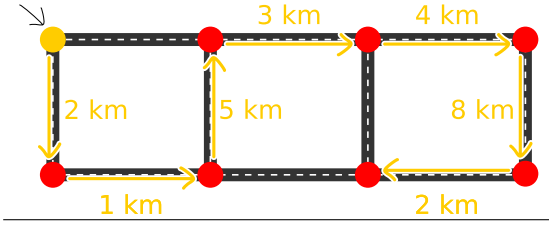
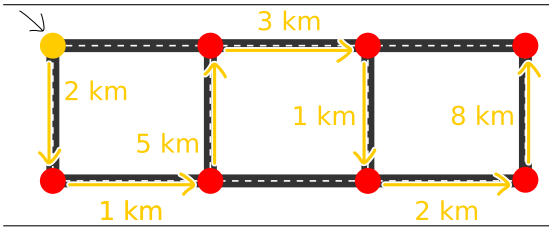
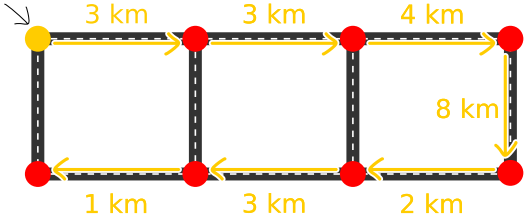
C) 24 km

D) 25 km

E) 26 km

Lösung

Insgesamt gibt es nur vier mögliche Wege, ohne dass ein Weg doppelt gefahren wird. Die Längen sind:

Weg	Länge
	$2\,\text{km} + 1\,\text{km} + 3\,\text{km} + 2\,\text{km} + 8\,\text{km} + 4\,\text{km} + 3\,\text{km} = 23\,\text{km}$
	$2\,\text{km} + 1\,\text{km} + 5\,\text{km} + 3\,\text{km} + 4\,\text{km} + 8\,\text{km} + 2\,\text{km} = 25\,\text{km}$
	$2\,\text{km} + 1\,\text{km} + 5\,\text{km} + 3\,\text{km} + 1\,\text{km} + 2\,\text{km} + 8\,\text{km} = 22\,\text{km}$
	$3\,\text{km} + 3\,\text{km} + 4\,\text{km} + 8\,\text{km} + 2\,\text{km} + 3\,\text{km} + 1\,\text{km} = 24\,\text{km}$

Andere Wege kann es nicht geben: in allen anderen Fällen müsste man einen Ort zweimal anfahren, um über ihn zu einem anderen Ort zu kommen.
Damit ist klar, dass D) 25 km die richtige Antwort ist.

Dies ist Informatik!

In dieser Aufgabe geht es darum einen Weg zu finden, der sämtliche Orte auf einer Karte genau einmal besucht. Einen solchen Weg nennt man einen *Hamiltonpfad*. Da man eine Karte als *Graph*, die Orte als *Knoten* und die Wege als *Kanten* interpretieren kann, ist die Frage nach einem Hamiltonpfad ein graphentheoretisches Problem. In einem beliebigen Graphen einen Hamiltonpfad zu finden oder überhaupt herauszufinden, ob es einen gibt, ist eines der berühmten NP-vollständigen Probleme, die von Computern nicht effizient gelöst werden können.
In diesem Fall wird nicht nur irgend einen Hamiltonpfad gesucht, es wird von allen möglichen Hamiltonpfaden derjenige gesucht, der den maximalen Wert hat, berechnet aus der Summe der Kantenwerte. Dies macht das Problem noch aufwendiger.
In diesem Fall schlagen auch Ansätze fehl, die in anderen Fällen eine akzeptable, wenn auch suboptimale Lösung finden. Ein klassischer Ansatz wäre das *greedy*-Verfahren, bei dem man möglichst grosse Werte sammelt, man also zuerst den Weg mit 3 km Länge wählt. Mit diesem Verfahren findet man aber nur den Weg, der 24 km lang ist. Wenn man gar so „gierig“ weiter geht und als zweites den

Weg mit 5 km Länge geht, verbaut man sich die Möglichkeit, überhaupt alle Orte einmal angefahren zu haben.

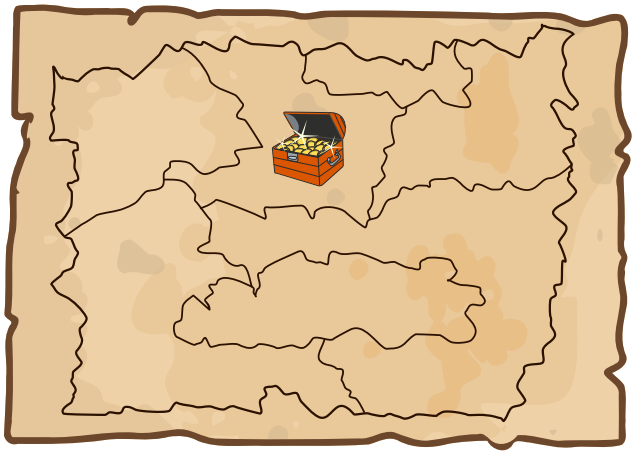
Stichwörter und Webseiten

Hamiltonpfad, NP-vollständig

- `https://de.wikipedia.org/wiki/Hamiltonkreisproblem`
- `https://de.wikipedia.org/wiki/NP-Vollst%C3%A4ndigkeit`
- `https://de.wikipedia.org/wiki/Greedy-Algorithmus`

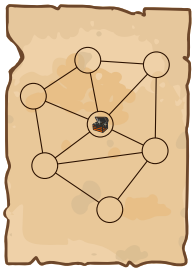
15. Schatzkarte

Der König der Biber regiert über sieben Provinzen, deren Grenzen auf der Karte unten dargestellt sind. In einer der Provinzen hat er seinen Schatz versteckt:

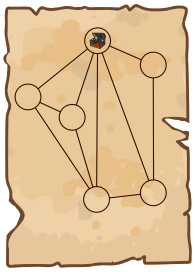


Der König hat eine Schatzkarte anfertigen lassen, in der die Provinzen als Kreise dargestellt sind. Die Provinz mit dem Schatz hat er gekennzeichnet. Zwei Kreise sind immer dann verbunden, wenn die entsprechenden Provinzen eine gemeinsame Grenze haben. Um Räuber davon abzuhalten, den Schatz zu finden, hat der König zusätzlich vier falsche Schatzkarten anfertigen lassen.
Welches ist die richtige Schatzkarte?

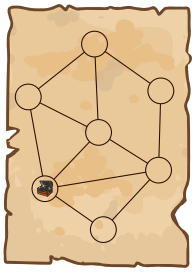
A)



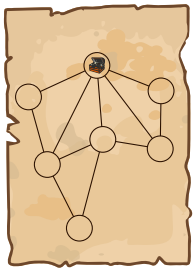
B)



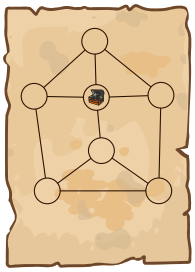
C)



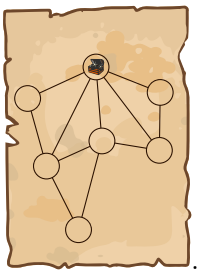
D)



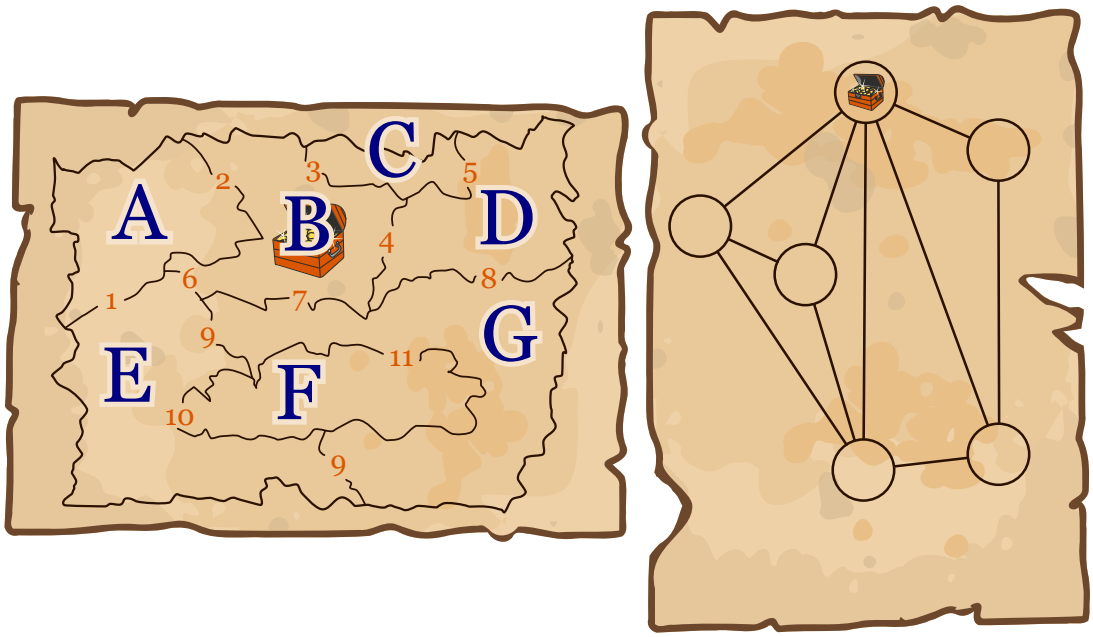
E)



Lösung

Die korrekte Antwort ist D)  .

In der folgenden Ansicht sind die Gebiete mit den Buchstaben A, B, C, D, E, F und G bezeichnet. Die Grenzen zwischen den Provinzen sind mit den Zahlen 1 bis 11 bezeichnet. Da es zwischen den Provinzen E und G zwei Grenzen gibt, die Bedingung jedoch lediglich lautet, ob eine gemeinsame Grenze existiert, sind beide mit 9 bezeichnet. Mit Hilfe dieser Bezeichnungen kann man sich schnell selber davon überzeugen, dass dies eine korrekte Schatzkarte ist.

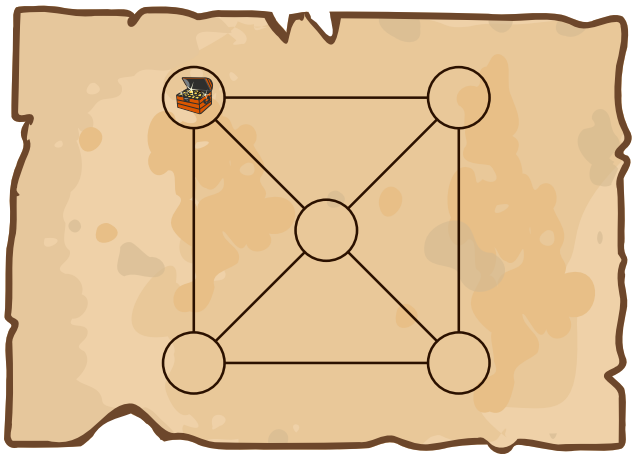


Die Antwort A) ist falsch: Die drei Provinzen A, C und F grenzen nur an zwei weitere Provinzen an. Daher müsste es auf der Schatzkarte drei Kreise geben, von denen jeweils zwei Linien ausgehen. Es gibt aber lediglich einen Kreis von dem zwei Linien ausgehen.
Die Antwort B) kann nicht richtig sein, da sie nur sechs Kreise hat. Es gibt aber sieben Provinzen.
Die Antwort C) kann ebenfalls nicht richtig sein: Die Provinz mit dem Schatz ist Nachbar der fünf Provinzen A, C, D, E und G, also müssten fünf Linien von dem gekennzeichneten Kreis ausgehen, es sind aber nur vier.
Die Antwort E) sieht der Karte der Provinzen am ähnlichsten. Jedoch sind die Grenzen nicht alle korrekt eingezeichnet, die Provinz B mit dem Schatz beispielsweise hat eine Grenze zu wenig.

Dies ist Informatik!

In dieser Aufgabe geht es darum, eine Karte mit Hilfe eines *Graphen* zu repräsentieren. Ein solcher Graph stellt eine *Abstraktion* der Karte dar (so wie die Karte ebenfalls schon eine Abstraktion der Realität darstellt). Wie bei jeder Abstraktion gehen unwichtige Informationen verloren. In dieser Aufgabe sind das die geographischen Positionen der Provinzen zueinander: obwohl die Provinzen B

und D in der Karte auf derselben Höhe liegen, liegen sie im Graphen auf unterschiedlichen Höhen. Die wesentlichen Informationen bleiben bei der Abstraktion jedoch erhalten, das ist im Fall dieser Aufgabe die Antwort auf die Frage, welche Provinzen benachbart sind. Dennoch birgt auch eine korrekte Abstraktion die Gefahr, dass eine scheinbar unwichtige Information verloren geht: der folgende Graph eines anderen Staates ist dreh-symmetrisch, so dass weiterhin nicht klar ist, wo der Schatz verborgen sein könnte:



Ein Graph besteht aus *Knoten* (in dieser Aufgabe die Kreise) und *Kanten* (in dieser Aufgabe die Linien). Der Graph dieser Aufgabe beschränkt Kanten zudem darauf, lediglich zwei verschiedene Knoten zu verbinden. Auch gibt es keine Möglichkeit, zwei Knoten mit mehreren Kanten zu verbinden, was zwischen den Provinzen E und G durchaus denkbar gewesen wäre.
Graphen werden in der Informatik häufig zum Abstrahieren von Informationen verwendet. In vielen Fällen ist dieser Abstraktionsschritt schon fast die Lösung eines Problems: da Graphen in der Informatik sehr gut erforscht sind, genügt es häufig, ein Problem auf eines der typischen Graphen-Probleme zurückzuführen und die Lösungen anzuwenden, die man dort bereits gefunden hat.

Stichwörter und Webseiten

Graph, Graphentheorie

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

A. Aufgabenautoren

Tony René Andersen
Michelle Barnett
Michael Barot
Wilfried Baumann
Jan Berki
Linda Bergsveinsdóttir
Laura Braun
Špela Cerar
Mony Chanroath
Sébastien Combéfis
Kris Coolsaet
Christian Datzko
Maria Suyana Datzko
Susanne Datzko
Guillaume de Moffarts
Gerald Futschek
Arnheiður Guðmundsdóttir
Martin Guggisberg
Juraj Hromkovič
Alisher Ikramov
Takeharu Ishizuka
M. Faiz Ahmad Ismail
Anna Laura John
Mile Jovanov
Ungyeol Jung
Ilya Kaysin
Jihye Kim
Vaidotas Kinčius
Mária Kiss
Bohdan Kudrenko
Regula Lacher
Anh Vinh Lê
Greg Lee
Judith Lin
Lynn Liu
Violetta Lonati
Vũ Văn Luân
Mattia Monga
Samart Moodleah
Madhavan Mukund
Tom Naughton
Tomohiro Nishida
Assylkan Omashev
Margot Phillipps
Zsuzsa Pluhár
Sergei Pozdniakov
Nol Premasathian
J.P. Pretti
Milan Rajković
Chris Roffey
Eljakim Schrijvers
Humberto Sermeno
Daigo Shirai
Jacqueline Staub
Nikolaos Stratis
Bundit Thanasopon
Peter Tomcsányi
Nicole Trachsler
Jiří Vaníček
Troy Vasiga
Michael Weigend

B. Sponsoring: Wettbewerb 2019

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

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

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

http://www.verkehrshaus.ch/

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

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

UBS

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

bbv Software Services

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

PRESENTEX
Das Geschenk – die gute Werbung

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

OXOCARD

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

DIARTIS

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

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

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

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

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

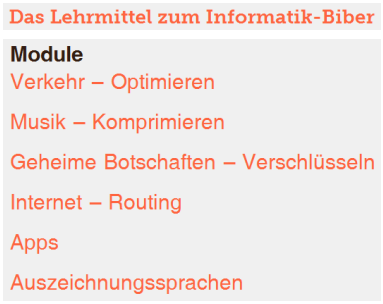
http://www.phlu.ch/
Pädagogische Hochschule Luzern

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

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

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

C. Weiterführende Angebote



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

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

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

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