INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Aufgaben und Lösungen 2023

Schuljahre 5/6

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

Herausgeber:

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

SVIA

www.svia-ssie-ssii.ch
schweizerischerverein für informatikind
erausbildung//société suisse pour l'infor
matiquedans l'enseignement//società sviz
zera 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 2023

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

Projektleitung: Nora A. Escherle

Herzlichen Dank für die Aufgabenentwicklung für den Schweizer-Wettbewerb an:
Juraj Hromkovič, Angélica Herrera Loyo, Regula Lacher und Manuel Wettstein: ETH Zürich, Ausbildunges- und Beratungszentrum für Informatikunterricht
Tobias Berner: Pädagogische Hochschule Zürich
Christian Datzko: Wirtschaftgymnasium und Wirtschaftsmittelschule, Basel
Fabian Frei: CISPA - Helmholtz-Zentrum für Informationssicherheit
Sebastian Knüsli: Gymnasium Kirschgarten, Basel

Die Aufgabenauswahl wurde erstellt in Zusammenarbeit mit den Organisatoren von Bebras in Deutschland, Österreich, Ungarn und Litauen. Besonders danken wir:
Valentina Dagienė, Vaidotas Kinčius: Bebras.org, Litauen
Wolfgang Pohl, Jakob Schilke: Bundesweite Informatikwettbewerbe (BWINF), Deutschland
Hannes Endreß: Materna Information & Communications SE, Deutschland
Ulrich Kiesmüller: Simon-Marius-Gymnasium Gunzenhausen, Deutschland
Kirsten Schlüter: Bayerisches Staatsminiserium für Unterricht und Kultus, Deutschland
Margareta Schlüter: Universität Tübingen, Deutschland
Jacqueline Staub: Universität Trier, Deutschland
Michael Weigend: WWU Münster, Deutschland
Wilfried Baumann, Liam Baumann, Josefine Hiebler: Österreichische Computer Gesellschaft, Österreich
Gerald Futschek: Technische Universität Wien, Österreich
Zsuzsa Pluhár: ELTE Informatikai Kar, Ungarn

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

Für den Support während den Wettbewerbswochen danken wir:
Hanspeter Erni: Schulleitung Sekundarschule Rickenbach
Gabriel Thullen: Collège des Colombières, Versoix

Für die Organisation und Durchführung des Biberfinals an der ETH danken wir:
Dennis Komm, Hans-Joachim Bückenhauer, Jan Lichensteiger, Moritz Stocker: ETH Zürich, Ausbildunges- und Beratungszentrum für Informatikunterricht
Für die Korrektur der Finalaufgaben:
Fiona Binder, Joel Birrer, Marlene Bötschi, Danny Camenisch, Gianluca Danieletto, Alexander Frey,

Sven Grübel, Laure Guerrini, Charlotte Knierim, Richard Královič, Yanik Künzi, Kenli Lao, Sandro Marchon, Zoé Meier, Dario Näpfer, Kai Zürcher
Für die Übersetzung der Finalaufgaben ins Französische:
Jan Schönbächler: Lycée-Collège de l'Abbaye de St-Maurice

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

Ganz besonderen Dank gilt unseren grossen Förderern Juraj Hromkovič, Dennis Komm, Gabriel Parriaux und der Haslerstiftung. Ohne sie würde es diesen Wettbewerb nicht geben.

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 2023 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, der Kanton Bern, die Post sowie die UBS.

Dieses Aufgabenheft wurde am 10. Januar 2024 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 2023 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 2023 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

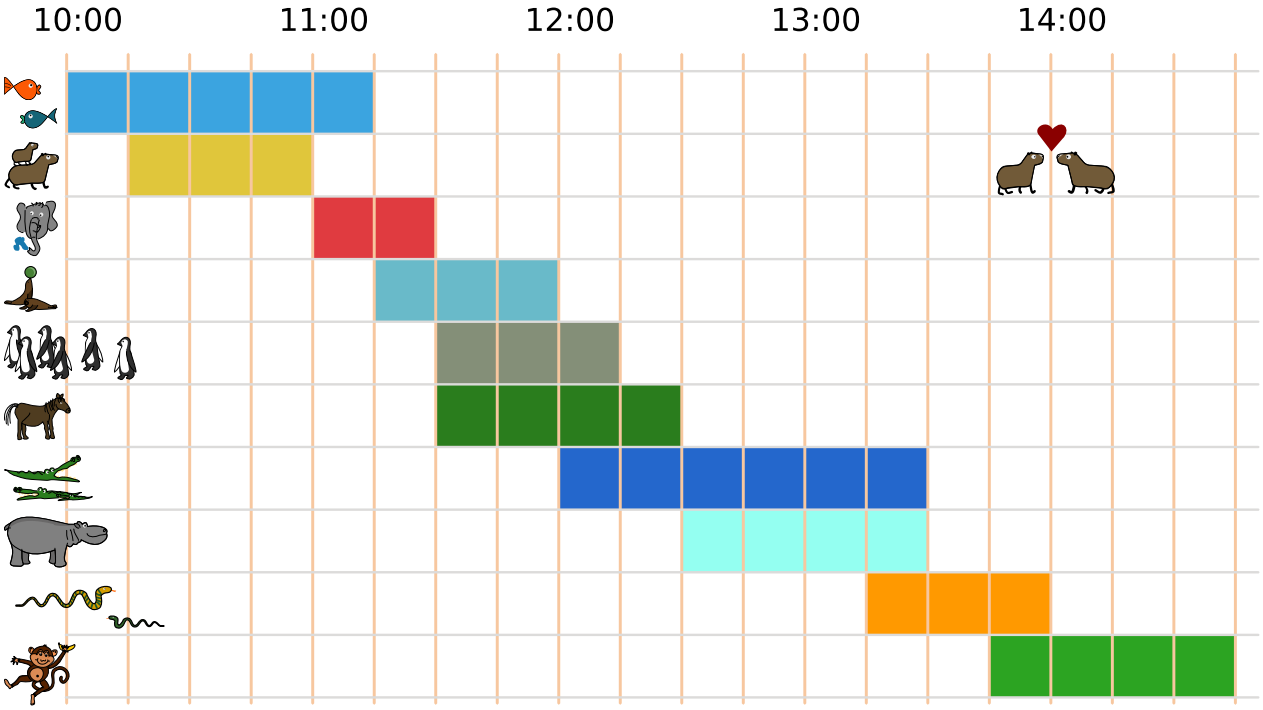
`https://www.informatik-biber.ch/de/kontaktieren/`
`https://www.informatik-biber.ch/`

Inhaltsverzeichnis

1. Spass im Zoo

Heute ist Anja im Zoo. Sie will möglichst viele verschiedene Vorführungen besuchen.

Hier ist ein Plan mit allen Vorführungen. Zum Beispiel siehst du ganz unten: Die Vorführung der Affen beginnt um 13:45 Uhr und endet um 14:45 Uhr.

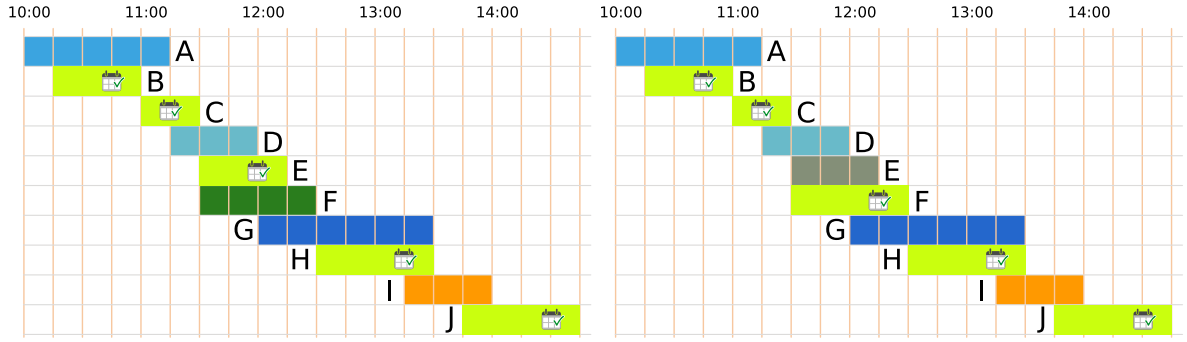


Anja besucht eine Vorführung immer ganz, von Anfang bis Ende. Kannst du Anja helfen?

Wähle so viele Vorführungen wie möglich aus, die Anja nacheinander besuchen kann.

Lösung

Anja kann höchstens 5 Vorführungen nacheinander besuchen. Das sind die beiden richtigen Antworten:



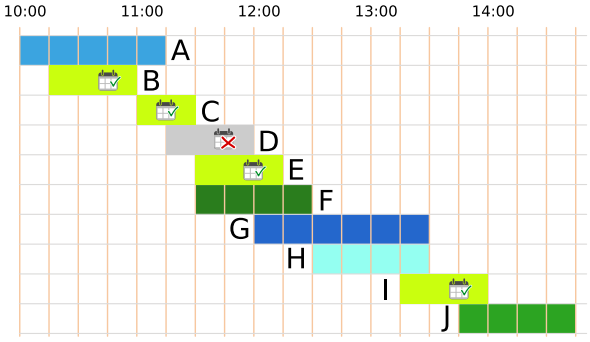
Es gibt unterschiedliche Wege, die richtigen Antworten zu finden.

Ein Besuchsplan für Anja ist eine Auswahl von Vorführungen, die sie nacheinander besuchen kann. Ein Weg zu den richtigen Antworten ist es, alle Besuchspläne aufzulisten. In dieser Liste sind die Pläne mit den meisten Vorführungen die richtigen Antworten. Alle Besuchspläne zu finden, ist leider sehr zeitaufwändig.

Aber könnte es nicht auch einen Besuchsplan mit 6 Vorführungen geben? Wir versuchen einmal, einen zu erstellen. Vorher schauen wir uns die Dauer der Vorführungen genauer an: Der gesamte Tag ist auf dem Plan in 19 Zeiteinheiten zu je einer Viertelstunde unterteilt. Die Vorführungen dauern 2, 3, 4, 5 oder 6 Zeiteinheiten.

Zeiteinheiten	**Vorstellungen**
2	C
3	B, D, E, I
4	F, H, J
5	A
6	G

Um möglichst viele Vorführungen in einen Besuchsplan zu packen, wählen wir so *kurze* Vorführungen wie möglich. Die 6 kürzesten Vorführungen dauern zusammen 18 Zeiteinheiten ($2 + 3 + 3 + 3 + 3 + 4$). Zu diesen kurzen Vorführungen gehören auch die Vorführungen C, D und E. Da die Vorführungen C und E aber genau hintereinander liegen, kann Anja Vorführung D dazwischen nicht besuchen.



Wir müssen also die Vorführung D durch eine andere möglichst kurze ersetzen. Es sind nur noch Vorführungen mit mindestens 4 Zeiteinheiten übrig. Ohne die Vorführung D benötigen wir deshalb insgesamt mindestens 19 Zeiteinheiten für 6 Vorführungen: $2 + 3 + 3 + 3 + 4 + 4$. Aber: Welche zwei Vorführungen mit 4 Zeiteinheiten wir auch wählen, eine davon überschneidet sich immer mit einer Vorführung mit 3 Zeiteinheiten. Wir müssten auch diese durch eine Vorführung mit mindestens 4 Zeiteinheiten ersetzen und würden dann mindestens 20 Zeiteinheiten für 6 Vorführungen benötigen. Es stehen aber nur 19 Zeiteinheiten zur Verfügung! Wir schlussfolgern, dass es keinen Besuchsplan geben kann, der mehr als 5 Vorführungen enthält.

Dies ist Informatik!

Diese Biberaufgabe enthält einen Zeitplan der Vorführungen im Zoo. Solche Zeitpläne herzustellen, ist nicht einfach und wird in der Informatik als *Scheduling-Problem* bezeichnet. Natürlich möchte der Zoo seinen Besuchern ermöglichen, möglichst viele Vorführungen zu sehen, aber es müssen auch andere Bedingungen beachtet werden. Beispielsweise können Vorführungen nur angeboten werden, wenn die Tierpfleger Zeit haben, die verfügbaren Arenas frei sind und die Vorführungen sich mit den Lebensrhythmen der Tiere vereinbaren lassen.

Es gibt viele ähnliche Probleme im Leben, auf die sich dieselben Überlegungen anwenden lassen. Ein Beispiel ist die Erstellung eines Stundenplanes in der Schule, oder die Zuteilung von Kinofilmen zu Kinosälen. Die Erstellung dieser Zeitpläne ist so aufwändig, dass man dies schon für relativ kleine Beispiele (die Stundenpläne deiner Schule) oft nicht mehr von Hand erarbeiten kann. Auch die *Prozessoren* deines Computers müssen viele Aufgaben übernehmen und diese nacheinander abarbeiten. Der Zeitplan, wann welcher Prozessor was tut, wird vom *Betriebssystem* blitzschnell und ohne dass man es merkt erstellt. Scheduling ist eines der grossen Themen der Informatik, mit welchen sich die Forschung auch heute noch beschäftigt.

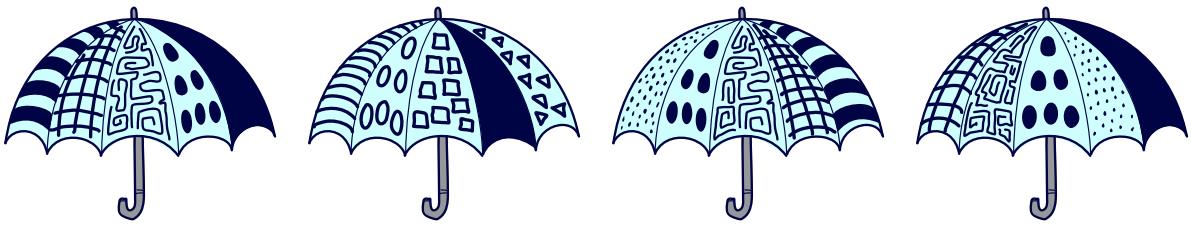
Stichwörter und Webseiten

- Sheduling-Problem: `https://de.wikipedia.org/wiki/Scheduling`
- Betriebssystem: `https://de.wikipedia.org/wiki/Betriebssystem`

2. Annas Regenschirm

Das ist Annas Regenschirm:

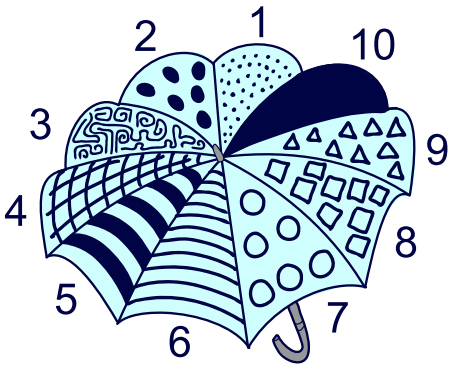
Eines der vier Bilder zeigt Annas Regenschirm. Welches?



A) B) C) D)

Lösung

Jedes Muster auf Annas Regenschirm kommt genau einmal vor.



Um das korrekte Bild zu finden, vergleichen wir nacheinander jedes der Bilder mit Annas Regenschirm:

- wir wählen das Muster, welches am weitesten links ist, und suchen dessen Position auf Annas Regenschirm
- wir prüfen, ob die angrenzenden Muster dieselben sind wie die auf Annas Regenschirm.

	A)	**B)**	**C)**	**D)**
Antwortbild				
Annas Regenschirm				

Jedes der vier Bilder zeigt eine Folge von nur fünf Mustern und nicht alle zehn. Wir können nicht wissen, ob die Musterfolge von einem der vier Bilder mit der vollständigen Reihenfolge aller zehn Muster von Annas Regenschirm übereinstimmt.

Bild C hat als einziges eine Folge von fünf Mustern, die vollständig mit denen auf Annas Regenschirm übereinstimmt. Aus diesem Grund kann nur Bild C Annas Regenschirm zeigen. Alle anderen Bilder weisen Musterfolgen auf, die nicht oder nur teilweise mit denen von Annas Regenschirm übereinstimmen. Diese Bilder können also nicht Annas Regenschirm zeigen.

Dies ist Informatik!

In den Antwortmöglichkeiten ist jeweils nur ein Teil der Musterfolge abgebildet. Obwohl sie nur eine *Teilinformation* enthalten, können wir feststellen, welches der vier Bilder Annas Regenschirm zeigt: Ein Bild zeigt nur dann Annas Regenschirm, wenn die Musterfolge vollständig in der Musterfolge von Annas Regenschirm vorkommt.

Das gleiche Prinzip wie bei der «Regenschirm-Mustersuche» wird bei der Suche in einem Textdokument angewendet. Der Computer sucht mit gegebenen Teilinformationen (Suchwort) nach passenden *Zeichenketten* im Dokument. Eine Zeichenkette ist eine Folge von Zeichen (z.B. Buchstaben, Ziffern, Sonderzeichen). Dabei gilt bei der Suche:

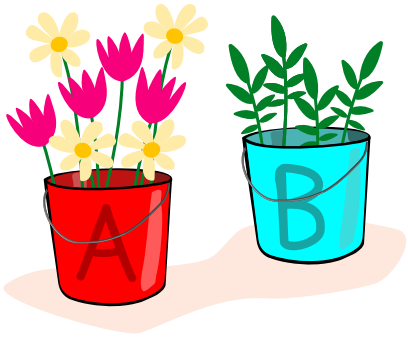
- Je länger das Suchwort, desto weniger mögliche Übereinstimmungen gibt es und desto grösser ist die Chance, die gesuchte Stelle im Dokument zu finden.
- Je kürzer das Suchwort, desto mehr mögliche Übereinstimmungen ergibt die Suche und desto ungenauer ist die Suche.

Um das Durchsuchen zu verbessern, wurden verschiedene Suchverfahren (oder *Suchalgorthimen*) entwickelt. Sie sollen möglichst schnell eine genaue Suche durchführen, und ein passendes Resultat liefern. Diese Suchalgorthimen werden ständig weiterentwickelt und können riesige Datenmengen in sehr kurzer Zeit durchsuchen (z.B. Internetsuchmaschinen verwenden solche Suchalgorthimen).

Stichwörter und Webseiten

- Zeichenkette, String: `https://de.wikipedia.org/wiki/Zeichenkette`
- Suchverfahren: `https://de.wikipedia.org/wiki/Suchverfahren`

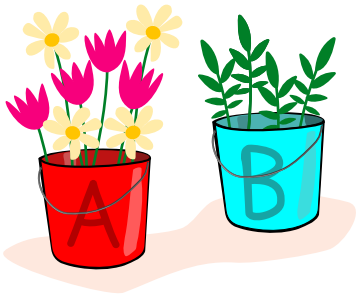
3. Blumenstrauss

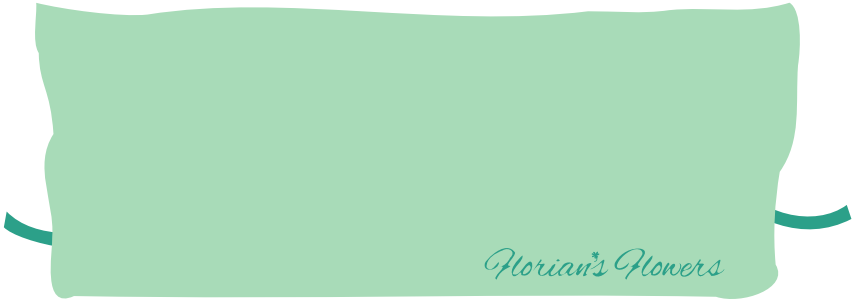


Florian verkauft Blumensträusse. Jeden Blumenstrauss bindet Florian nach dieser Anleitung:

1. Nimm die erste Blume aus Eimer A.
2. Wenn die erste Blume eine Margarite ist, nimm noch eine Margarite .
3. Nun nimm solange einen Zweig aus Eimer B, bis der Blumenstrauss 4 Teile hat. Fertig!

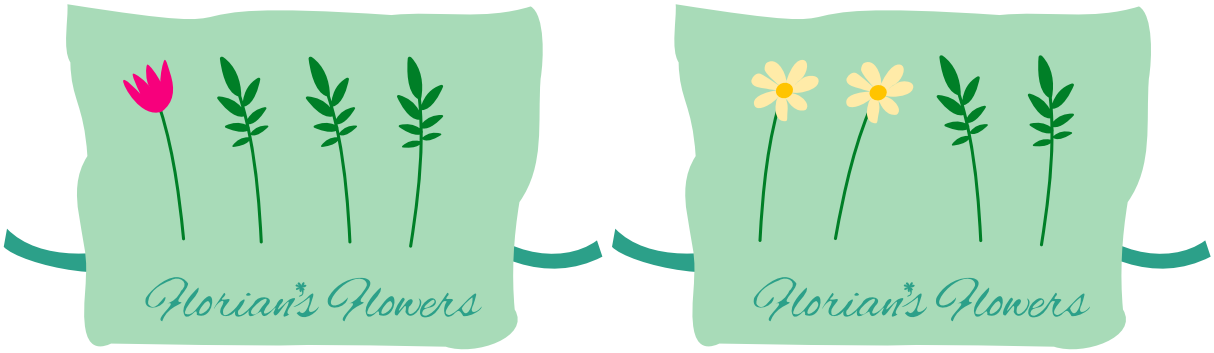
Hilf Florian: Folge der Anleitung und wähle Blumen und Zweige für einen Strauss aus.



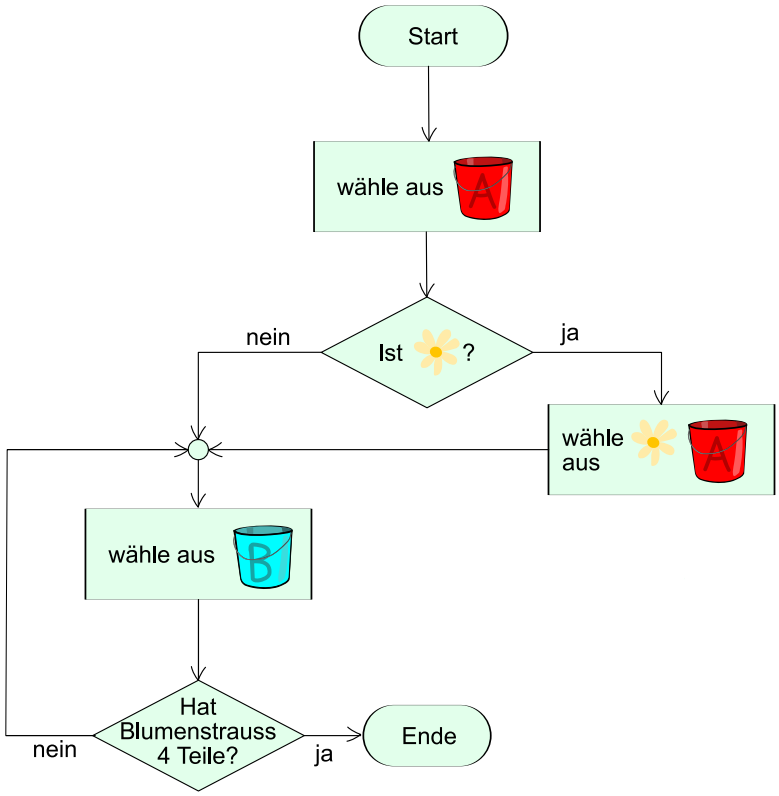


Lösung

Es gibt zwei richtige Lösungen:



Um die Blumensträusse korrekt zu binden, muss Florian die Anleitung befolgen. Wir können die Anleitung auch mit einem Diagramm beschreiben:



Nachdem Florian die erste Blume aus Eimer A gewählt hat, folgt eine Entscheidung, die abhängig von der ersten Blume ist. Entweder nimmt er noch eine Margarite (🌼) oder er folgt dem «nein»-Pfeil und nimmt einen Zweig 🌿.

Dann überprüft er, ob er schon vier Teile hat. Wenn nicht, folgt er dem «nein»-Pfeil und muss einen weiteren Zweig 🌿 nehmen und dann die Anzahl der Teile wieder überprüfen.

Wenn er also zuerst eine Margarite 🌼 nimmt, wird er noch eine Margarite nehmen und dann zweimal einen Zweig 🌿 nehmen. Wenn er aber zuerst eine Tulpe 🌷 nimmt, wird er danach direkt zu «wähle

aus Eimer B» gehen und aus Eimer B solange einen Zweig nehmen, bis er 4 Teile hat – also insgesamt 3 Zweige nehmen.

Dies ist Informatik!

Die *Anleitung* fürs Binden des Blumenstrausses sind klar und könnten von einer Maschine ausgeführt werden. In der Informatik nennt man dies einen *Algorithmus*. Die Anleitung benutzt einige Anweisungen, die auch in Computerprogrammen üblich sind:

- Die erste Anweisung ist eine zufällige Auswahl aus einer Menge von Objekten.
- Die zweite Anweisung nennt man eine *bedingte Anweisung* (engl. *if-statement*) oder eine *Verzweigung*: Denn du musst aus zwei oder mehr Möglichkeiten auswählen.
- Die dritte Anweisung sieht relativ einfach aus, muss aber in einem Computerprogramm gut strukturiert werden. Der innere Teil der Anweisung (selbst wieder eine Anweisung: «Nimm einen Zweig aus Eimer B») muss mehrmals ausgeführt werden, bis der Blumenstrauss aus 4 Teilen besteht. Die Ausführung der inneren Anweisung wird also solange wiederholt, bis die Bedingung «Der Blumenstrauss besteht aus 4 Teilen.» erfüllt ist. Eine solche *Wiederholungs-Anweisung* nennt man auch *Schleife*.

Ein Algorithmus kann unterschiedlich dargestellt werden. In dieser Biberaufgabe ist Florians «Blumenstrauss-Algorithmus» als Anleitung in natürlicher Sprache formuliert. In der Lösungserklärung ist er als *Programmablaufplan* dargestellt.

Floristik ist eine Handwerkskunst. Es existieren Traditionen und Regeln, wie ein Blumenstrauss oder ein Kranz gebunden wird. Dies ist ein Beispiel dafür, dass Anleitungen oder Algorithmen in vielen Lebensbereichen vorkommen, nicht nur in der Informatik.

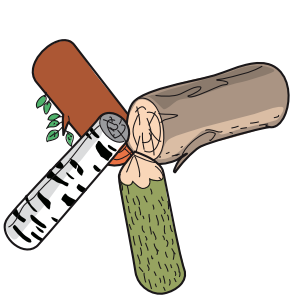
Stichwörter und Webseiten

- bedingte Anweisungen und Verzweigungen:
  `https://de.wikipedia.org/wiki/Bedingte_Anweisung_und_Verzweigung`
- Schleife: `https://de.wikipedia.org/wiki/Schleife_(Programmierung)`
- Pseudocode: `https://de.wikipedia.org/wiki/Pseudocode`
- Programmablaufplan: `https://de.wikipedia.org/wiki/Programmablaufplan`
- Floristik: `https://de.wikipedia.org/wiki/Floristik_(Handwerk)`

4. Foto

Der Biber hat gerade ein Foto gemacht.

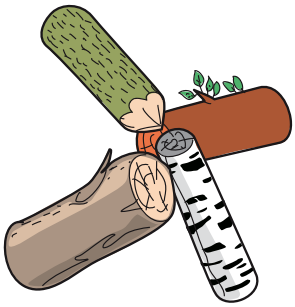
Welches der vier Fotos ist es?



A)

B)

C)



D)

Lösung

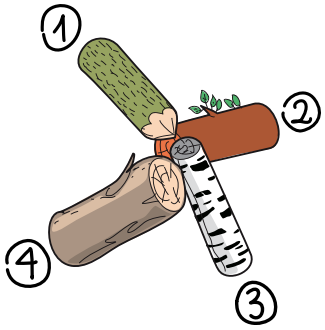
Die richtige Antwort ist D). 

Die Baumstämme, die der Biber fotografiert hat, sind im Kreis angeordnet. Um herauszufinden, welches Foto das richtige ist, betrachten wir die Reihenfolge der Baumstämme in dieser Anordnung. Wir wählen einen Baumstamm aus (z.B. den angespitzten Baumstamm) und geben ihm die Nummer 1. Dann bestimmen wir, welcher Baumstamm links daneben ist und geben ihm die Nummer 2. Das machen wir solange, bis alle Baumstämme eine Nummer haben. In der Situation, die der Biber fotografiert hat, haben die Stämme also diese Reihenfolge: 1 (angespitzter Stamm) – 2 (brauner Stamm mit Blättern) – 3 (Birkenstamm) – 4 (dicker brauner Stamm).

Nun betrachten wir die Reihenfolge der Stämme in den Fotos A bis D. Dabei beginnen wir wie oben mit dem angespitzten Baumstamm 1 und gehen immer nach links:

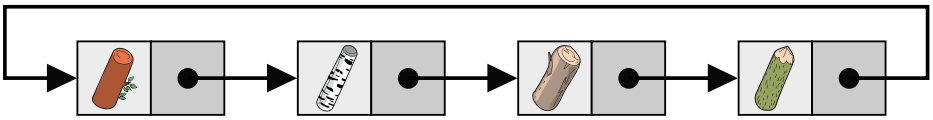
- Foto A: 1 – 3 – 2 – 4
- Foto B: 1 – 4 – 3 – 2
- Foto C: 1 – 3 – 4 – 2
- Foto D: 1 – 2 – 3 – 4

Nur Foto D zeigt die richtige Reihenfolge.



Dies ist Informatik!

In dieser Biberaufgabe wird die Reihenfolge der Baumstämme betrachtet. Was bei wenigen *Elementen* (hier vier Baumstämmen) durch einfaches «Hinsehen» und Vergleichen der Nachbarpaare möglich ist, erfordert bei Problemen mit viel mehr Elementen ein automatisiertes Vorgehen. In einem Computerprogramm, das benachbarte Elemente verarbeiten soll, könnten die Elemente in einer geeigneten Datenstruktur wie einer verketteten Liste gespeichert werden:



In einer *verketteten Liste* wird jedes Datenelement in einem einzelnen Knoten gespeichert. Zusätzlich ist in jedem Knoten ein *Verweis* auf den nächsten Knoten in der Liste gespeichert. Enthält der letzte Knoten einen Verweis auf den ersten Knoten, so handelt es sich um eine ringförmige Datenstruktur. Das ist im Beispiel wichtig, damit man bei jedem beliebigen Baumstamm starten und die Liste durchlaufen kann.

Stichwörter und Webseiten

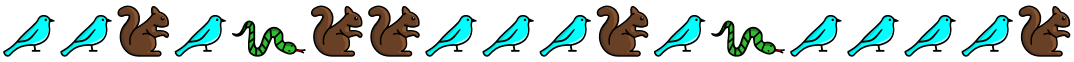
- verkettete Liste: `https://de.wikipedia.org/wiki/Liste_(Datenstruktur)`

5. Besonderer Baum

Ben hat einen besonderen Apfelbaum im Garten:

- Landet ein Vogel  auf dem Baum, wachsen sofort zwei neue Äpfel.
- Klettert ein Eichhörnchen  auf den Baum, fällt ein Apfel runter. Wenn kein Apfel am Baum hängt, passiert nichts.
- Besucht eine Schlange  den Baum, verschwinden alle Äpfel sofort.

Heute Morgen hängen 25 Äpfel am Baum. Dann besuchen einige Tiere nacheinander den Baum, zuletzt ein Eichhörnchen. Ben hat ihre Reihenfolge genau aufgeschrieben:



Wie viele Äpfel hängen danach am Baum?

A) 3 Äpfel

B) 7 Äpfel

C) 17 Äpfel

D) 31 Äpfel

Lösung

Antwort B ist richtig. Nachdem das letzte Eichhörnchen auf den Baum klettert, hängen noch 7 Äpfel am Baum.

Man kann für jeden Tierbesuch ausrechnen, wie viele Äpfel im Moment am Baum hängen:

Tier:	**Start**					
Anweisung:	–	+2	+2	−1	+2	reset
Anzahl Äpfel:	25	27	29	28	30	0

Tier:	**Übertrag**								
Anweisung:	–	–	–	+2	+2	+2	−1	+2	reset
Anzahl Äpfel:	0	0	0	2	4	6	5	7	0

Tier:	**Übertrag**					
Anweisung:	–	+2	+2	+2	+2	−1
Anzahl Äpfel:	0	2	4	6	8	**7**

Da alle Äpfel sofort verschwinden, wenn eine Schlange den Baum besucht, können wir alles ignorieren, was vor der Ankunft der zweiten (und letzten) Schlange passiert. Wie in der Tabelle gezeigt, landen nach dem Besuch der letzten Schlange vier Vögel auf dem Baum. Danach hängen am Baum $4 \times 2 = 8$ Äpfel. Dann klettert ein Eichhörnchen auf den Baum, wodurch ein Apfel herunterfällt und $8 - 1 = 7$ Äpfel übrigbleiben.

Dies ist Informatik!

Der Besuch eines Tieres verändert den Zustand des magischen Apfelbaums – aber nur auf ganz bestimmte Weise: Nur die Anzahl der Äpfel, die am Baum hängen, wird geändert. Auf andere Eigenschaften des Baumes, etwa die Anzahl der Blätter, die Länge einzelner Äste oder die Form der Baumkrone, hat der Besuch eines Tieres keinen Einfluss. Für diese Biberaufgabe ist es also ausreichend, die Anzahl der Äpfel zu betrachten.

Auch ein Computerprogramm hat einen Zustand, der von den einzelnen Anweisungen des Programms verändert wird. Als Zustand werden meist die von einem Programm gespeicherten Daten betrachtet; diese Daten speichert das Programm in den bei der Programmierung eingeführten *Variablen*.

Die Folge der Tierbesuche auf dem Baum in dieser Biberaufgabe ist wie ein Computerprogramm: Jeder Tierbesuch ist eine Anweisung, die den Zustand des Apfelbaums verändert. Dieser Zustand – also die Anzahl der Äpfel, siehe oben – kann in einer einzigen Variable gespeichert werden.

Beim Lösen der Aufgabe ist dir vielleicht aufgefallen, dass du nicht das ganze «Programm» anschauen musstest, sondern nur den Teil nach dem letzten Vorkommen der Schlange. Durch genaue Betrachtung

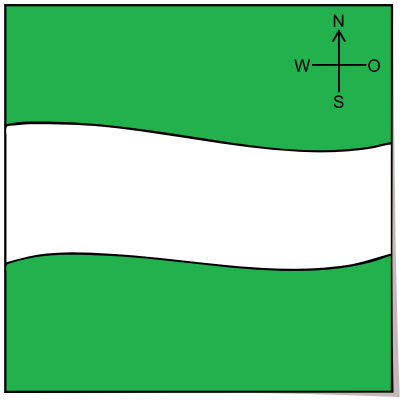
der Auswirkungen der einzelnen Anweisungen auf den Zustand des Programms konntest du eine besondere Eigenschaft des Programms herausfinden. Eine solche Analyse von (Computer-)Programmen gehört zu den häufigen Tätigkeiten von Informatikerinnen und Informatikern.

Stichwörter und Webseiten

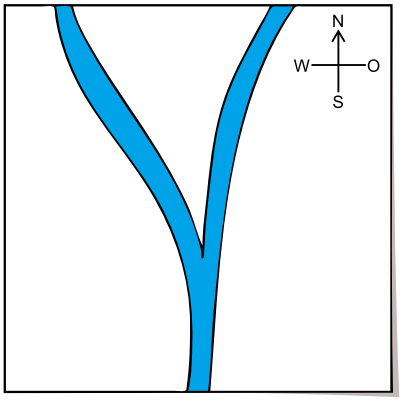
- Variablen: `https://de.wikipedia.org/wiki/Variable_(Programmierung)`
- (Programm-)Zustand: `https://media.kswillisau.ch/in/zustand/zustand.html`

6. Karlas Traumhaus

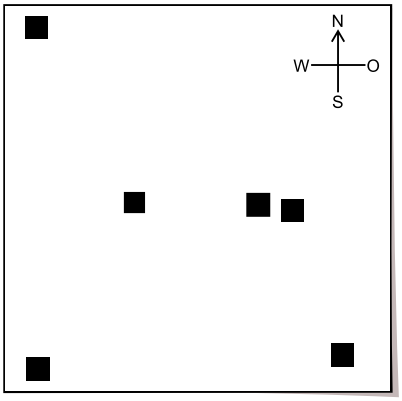
Karla hat drei Karten, die alle genau das gleiche Gebiet zeigen. Eine Karte zeigt die Wälder, eine die Flüsse und eine die Häuser in diesem Gebiet. Karlas Traumhaus liegt im Wald und in der Nähe eines Flusses.



Waldkarte



Flusskarte

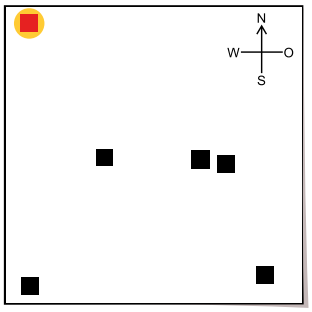


Hauskarte

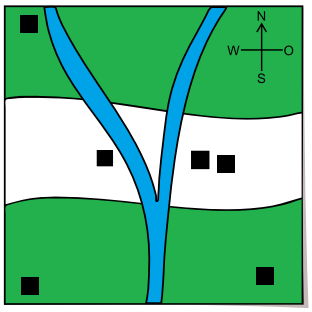
Welches ist Karlas Traumhaus?

Lösung

Das Haus oben links auf der Hauskarte ist Karlas Traumhaus:



Um Karlas Traumhaus zu finden, müssen die Informationen aus allen drei Karten ausgewertet werden. Das Traumhaus muss sich in einem Waldgebiet und in der Nähe eines Flusses befinden. Das trifft nur auf das Haus oben links zu. Dies ist leicht zu erkennen, wenn die Karten übereinander gelegt werden:



Dies ist Informatik!

Wenn die Informationen über die Wälder, die Flüsse und die Häuser auf einer einzigen Karte dargestellt sind, ist es einfach, das gesuchte Haus zu finden.

Ein *Geoinformationssystem* (GIS) führt eine Vielzahl räumlicher Informationen (z.B. Wälder, Strassen, Landesgrenzen, Tankstellen, Rathäuser, Überschwemmungsgebiete usw.) zusammen und stellt diese auf einer Karte dar. Ein GIS dient also der Visualisierung und Analyse sogenannter *Geodaten*. Mit Hilfe eines GIS ist es z.B. für Katastrophenschutzbeauftragte möglich, Informationen für Evakuierungspläne zusammenzustellen.

Die Verwendung mehrerer Ebenen mit unterschiedlichen Bildinformationen ist auch aus Grafikprogrammen bekannt. Eine wichtige Frage ist immer, welche Ebene mit den darin enthaltenen Objekten die oberste ist und deshalb im Vordergrund dargestellt wird. Im Beispiel sollte die Hauskarte die oberste Ebene sein, damit die Häuser nicht von den Waldflächen verdeckt werden.

Stichwörter und Webseiten

- GIS: `https://de.wikipedia.org/wiki/Geoinformationssystem`
- Ebenen: `https://de.wikipedia.org/wiki/Ebenentechnik`

7. Riccas

Evelyn hat fünf Bilder von Riccas. Sie beschreibt in Sätzen, wie Riccas aussehen.

Ihre Freundin Lydia zeigt ihr ein sechstes Bild von einem Ricca:

Nun stellt Evelyn fest: Einer ihrer Sätze über Riccas ist sicher falsch.

Welcher dieser Sätze über Riccas ist nun sicher falsch?

A) Alle Riccas haben Zähne.

B) Einige Riccas haben Flügel.

C) Riccas haben entweder Hörner oder drei Augen, aber nie Hörner *und* drei Augen.

D) Wenn Riccas genau zwei Arme haben, dann haben sie auch genau zwei Beine.

Lösung

Antwort D) ist richtig: *Wenn Riccas genau zwei Arme haben, dann haben sie auch genau zwei Beine.*

Antwort A) enthält eine Aussage, die für alle Riccas gelten muss. Wenn auch nur ein Ricca keine Zähne hätte, wäre sie falsch. Jedoch haben alle sechs Riccas, die Evelyn nun kennt, Zähne. Also kann Evelyns Satz nicht sicher falsch sein.

Antwort B) enthält eine Aussage, die nur für einige Riccas gelten soll. Da eines der sechs Riccas, die Evelyn nun kennt, Flügel hat, ist dieser Satz für die sechs Riccas richtig. Aber selbst wenn keines der sechs Riccas Flügel hätte, könnten andere, weitere Riccas Flügel haben, und der Satz könnte noch richtig sein. Dieser Satz kann nur dann sicher falsch sein, wenn Evelyn alle Riccas kennen würde und keines Flügel hätte.

Antwort C) verknüpft zwei Aussagen mit «entweder»-«oder». Diese verknüpfte Aussage ist genau dann wahr, wenn genau eine der beiden Aussagen wahr ist. Das ist für alle sechs Bilder der Fall: vier Riccas haben Hörner, aber keine drei Augen, die übrigen beiden Riccas haben keine Hörner, aber dafür drei Augen. Damit der Satz falsch wäre, müsste es mindestens ein Ricca mit drei Augen und Hörnern oder ein Ricca ohne Hörner und mit einer anderen Zahl als drei Augen geben. Unter den sechs Riccas, die Evelyn nun kennt, ist kein solches Ricca. Also ist der Satz für die sechs Riccas richtig, und nicht sicher falsch.

Bleibt der Satz von Antwort D). Er ist in Form einer «Wenn»-«Dann»-Aussage formuliert. Nur wenn die Wenn-Bedingung stimmt, muss auch die Dann-Aussage wahr sein. Die Bedingung ist wahr für alle sechs Riccas, die Evelyn nun kennt: sie alle haben genau zwei Arme. Alle Riccas auf Evelyns ersten fünf Bildern haben ebenfalls genau zwei Beine; für sie ist Evelyns Satz also richtig. Jedoch hat das Ricca auf Lydias Bild mehr als zwei Beine, nämlich fünf. Deshalb ist dieser Satz nun sicher falsch.

Dies ist Informatik!

Die Anzahl der Arme, Beine und Augen, und ob Riccas Zähne, Hörner oder Flügel haben, sind *Eigenschaften* von Riccas. Wenn man Riccas beschreibt, formuliert man *Aussagen* über diese Eigenschaften. Das führt zu einem *Modell* davon, was Riccas sind.

Computer haben ebenfalls viele Modelle. Einige sind explizit formuliert wie beispielsweise ein Modell von Schülerinnen und Schülern, das aus Name, Geburtsdatum und Wohnadresse in einer Datenbank besteht. Andere Modelle bilden Computer aus Daten, wenn man ihnen beispielsweise Bilder zum Vergleich beim Training eines neuronalen Netzes gibt.

Evelyns Sätze – also ihr Modell der Riccas in dieser Biberaufgabe – sind als *logische Ausdrücke* formuliert. Einige haben *Quantoren* («(für) alle» oder «es gibt»/«einige»), andere nutzen *logische Operatoren* («entweder»-«oder» bzw. «wenn»-«dann»). Diese logischen Ausdrücke sind *formalisiert*: Das heisst, dass es eine Festlegung gibt, wie man sie verwendet und was sie bedeuten.

Anhand dieser Festlegungen

- können (einfache) Ausdrücke mit Hilfe von Quantoren und Operatoren zu komplexeren Ausdrücken verknüpft werden.
- kann die Bedeutung der komplexeren Ausdrücke aus den einfacheren Ausdrücken berechnet werden.

Logische Ausdrücke sind eine in der Informatik verbreitete Methode, Modelle zu beschreiben.

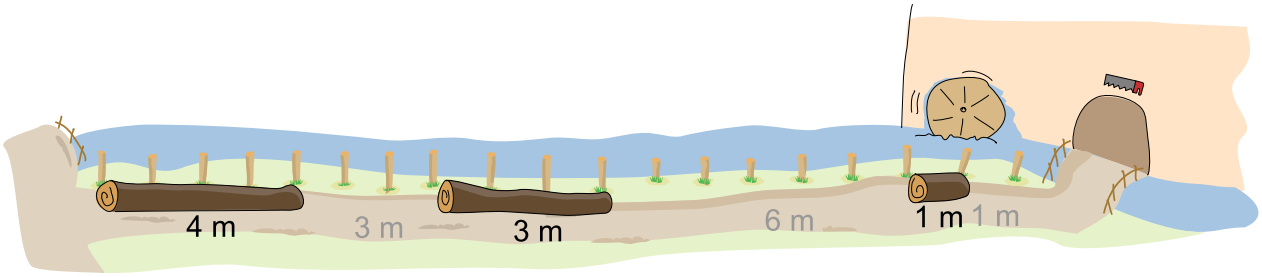
Stichwörter und Webseiten

Modellbildung: `https://de.wikipedia.org/wiki/Modell#Modellbildung`,
`https://de.wikipedia.org/wiki/Objektorientierte_Analyse_und_Design` Maschinelles Lernen:
`https://de.wikipedia.org/wiki/Maschinelles_Lernen`

8. Timeas Sägerei

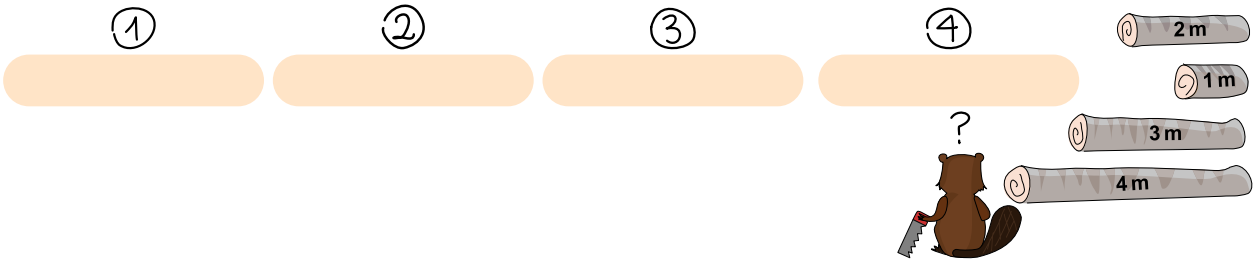
Biber Timea schneidet Holzstämme in verschiedenen Längen zu und verkauft sie dann. Sobald sie einen Stamm zugeschnitten hat, legt sie ihn auf dem 18 Meter langen Weg ab. Dabei beachtet Timea folgende Regel: Sie legt den Stamm in die erste Lücke von links, in die der Stamm passt.

Sie verkauft einige Stämme. Danach gibt es drei Lücken auf dem Weg:



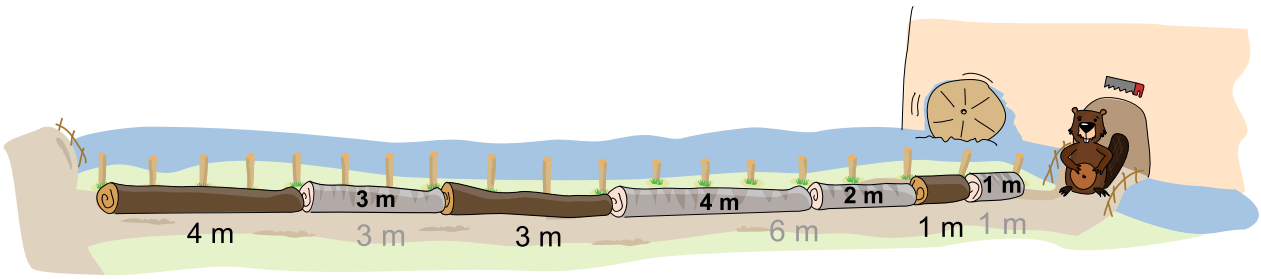
Nun will Timea vier Stämme zuschneiden, mit Längen von 1, 2, 3 und 4 Metern.

In welcher Reihenfolge muss Timea die Stämme zuschneiden, damit sie alle vier in die Lücken legen kann?



Lösung

So ist es richtig:



Schneidet Timea die Stämme in der Reihenfolge (3 m, 4 m, 2 m, 1 m) zu, passen sie alle auf den Weg: Für den 3 m-Stamm ist die 3 m-Lücke ganz links die erste freie Lücke von links, in die der Stamm passt; dort legt Timea den Stamm ab. Der 4 m-Stamm kommt dann in die 6 m-Lücke links. Dann ist die verbleibende 2 m-Lücke die erste freie Lücke von links; darein passt der nächste Stamm, und den letzten Stamm legt Timea in die 1 m-Lücke ab.

Weitere richtige Reihenfolgen sind (3 m, 2 m, 4 m, 1 m) und (4 m, 3 m, 2 m, 1 m).

Alle anderen Reihenfolgen führen dazu, dass Timea nicht in der Lage ist alle Baumstämme abzulegen: Der 1 m lange Stamm muss immer als letztes an der Reihe sein, weil nur dieser Stamm den letzten freien Platz ausfüllen kann. Der 2 m-Stamm darf nicht vor dem 3 m-Stamm kommen, weil er sonst in die 3 m-Lücke gelegt würde und dadurch eine zweite 1 m-Lücke entsteht. Ausser den drei genannten Reihenfolgen gibt es keine Reihenfolgen, die diese Bedingungen erfüllen.

Dies ist Informatik!

Diese Biberaufgabe ist ein Spezialfall des *Behälterproblems* (Englisch auch *bin packing problem*). Beim Behälterproblem müssen Objekte unterschiedlicher Grössen in einer bestimmten Anzahl von Behältern untergebracht werden, die selbst auch wieder unterschiedliche Grössen haben können. Die Objekte sind hier die Baumstämme, die Behälter die Lücken auf dem Weg.

Das Behälterproblem kommt in ganz unterschiedlichen Lebensbereichen vor. Einige Beispiele: (a) In einem Möbellager müssen kleine und grosse Möbel platzsparend untergebracht werden. (b) Eine Spedition kann Geld sparen, wenn sie zum Transport von Gütern durch geschicktes Packen weniger Lastwagen braucht. (c) Das Betriebssystem eines Computers muss Dateien unterschiedlicher Grösse auf der Festplatte speichern. Wenn Dateien gelöscht werden, entstehen Lücken auf der Festplatte. Diese Lücken müssen gefüllt werden, damit kein Speicherplatz verschwendet wird, ganz ähnlich wie auf der Strasse bei dieser Biberaufgabe.

In der Informatik gilt das Behälterproblem als eines der schwersten Probleme; garantiert optimale Lösungen können auch von Computerprogrammen nur für kleine Fälle mit wenigen Objekten und wenigen Behältern gelöst werden. Es gibt aber verschiedene Verfahren und Strategien, mit denen gute Lösungen des Behälterproblems bestimmt werden können. In dieser Biberaufgabe ist die Strategie durch Timeas Regel vorgegeben. Sie legt jeden Baumstamm immer in die erste Lücke von links,

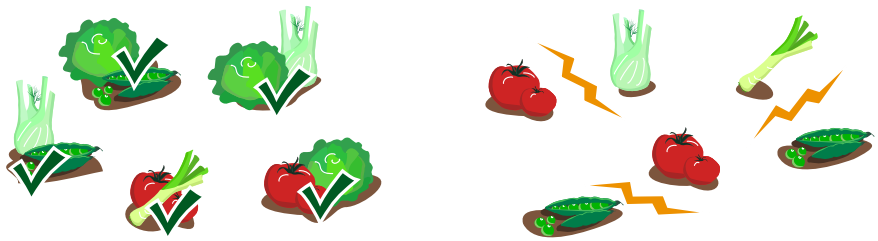
in die er passt. Diese Strategie nennt man *First Fit*. Man sieht am Beispiel dieser Aufgabe, dass diese Strategie zu schlechten Ergebnissen führen kann: Nur wenn die Stämme in einer bestimmten Reihenfolge abgelegt werden, können alle Lücken gefüllt werden.

Stichwörter und Webseiten

- Behälterproblem: `https://de.wikipedia.org/wiki/Behälterproblem`, `https://lamarr-institute.org/de/blog/bin-packing/`
- Speicherverwaltung: `https://de.wikipedia.org/wiki/Speicherverwaltung`
- Fragmentierung: `https://de.wikipedia.org/wiki/Fragmentierung_(Dateisystem)`

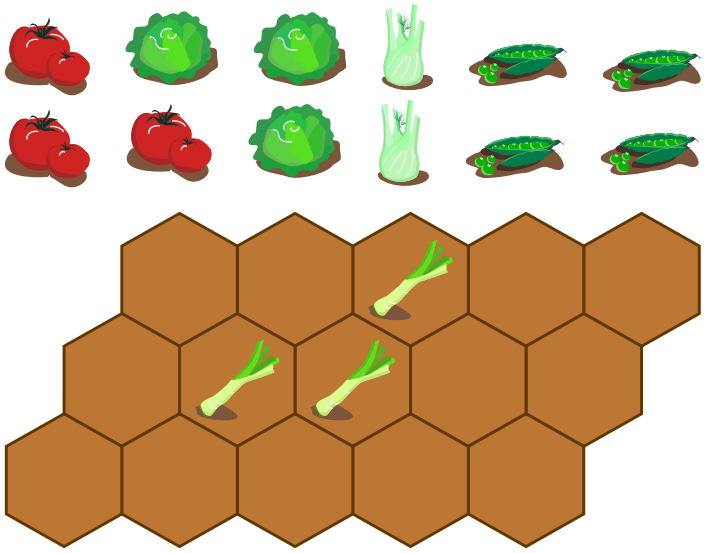
9. Gemüsebeet

Lisa legt ein Gemüsebeet an. Darauf will sie fünf verschiedene Gemüse pflanzen. Manche Gemüse vertragen sich gut miteinander ✔, andere nicht ↯:



Lisa hat das Beet in sechseckige Bereiche aufgeteilt. In jeden Bereich will sie genau ein Gemüse pflanzen.

In drei Bereiche hat Lisa schon Lauch  gepflanzt.

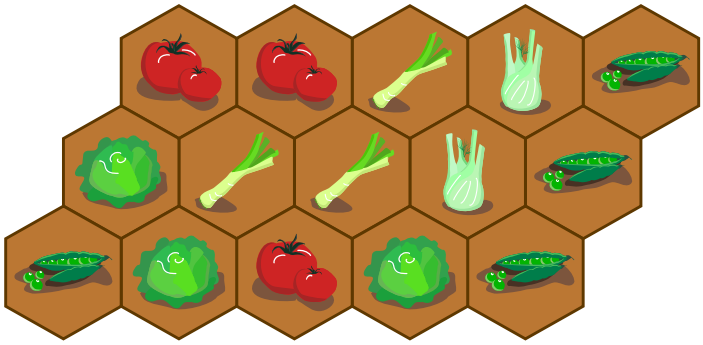


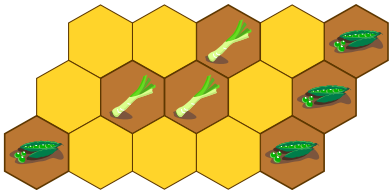
Beim Pflanzen beachtet Lisa folgende Regel: Gemüse, die sich nicht vertragen, dürfen nicht in Bereiche gepflanzt werden, die sich berühren.

Bepflanze alle noch freien Bereiche und beachte Lisas Regel!

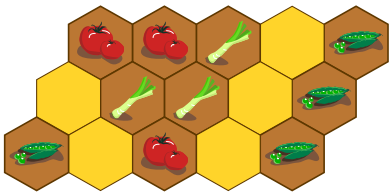
Lösung

So ist es richtig:

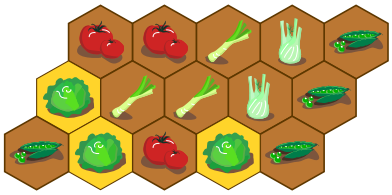




Weil Erbsen sich mit Lauch nicht vertragen, pflanzt Lisa keine Erbsen in die gelben Bereiche. Für die Erbsen bleiben nur die übrigen Bereiche.



Weil Tomaten sich mit Erbsen nicht vertragen, pflanzt Lisa keine Tomaten in die gelben Bereiche. In die übrigen Bereiche kann sie Tomaten pflanzen; Tomaten vertragen sich mit Lauch.



Weil Tomaten sich mit Fenchel nicht vertragen, pflanzt Lisa keinen Fenchel in die gelben Bereiche. Den Fenchel kann sie in die beiden Bereiche zwischen Lauch und Erbsen pflanzen. In die gelben Bereiche kann sie Salat pflanzen: Für Salat ist Lisa keine Unverträglichkeit bekannt.

Dies ist Informatik!

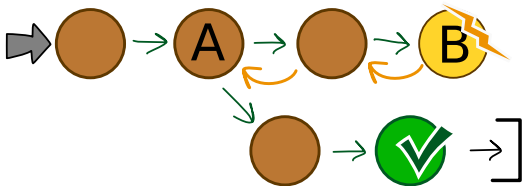
Wer Gemüse so pflanzen will, dass die Ernte möglichst gross wird, muss viele *Bedingungen* beachten: Die einzelnen Sorten haben zum Beispiel unterschiedlichen Bedarf an Platz, Nährstoff und Licht. In dieser Biberaufgabe betrachten wir nur eine Art von Bedingung: die Verträglichkeit zwischen den Gemüsesorten.

Um eine Bepflanzung von Lisas Beet zu finden, die alle Verträglichkeitsbedingungen beachtet, könnte man so vorgehen: Man probiert systematisch alle Kombinationen aus, die Gemüse auf dem Beet zu platzieren. Erst wenn das Beet voll ist, wird geprüft, ob diese Kombination alle Bedingungen erfüllt und eine Lösung für Lisas Problem ist. In der Informatik ist solch ein Ausprobieren aller Kombinationen als *Brute-Force*-Methode bekannt. Bei Problemen mit vielen Kombinationen und nur wenigen Lösungen kann ein Vorgehen nach dieser Methode sehr lange dauern.

Deshalb ist es meist besser, schrittweise vorzugehen und bei jedem Schritt alle Bedingungen zu berücksichtigen. Auf diese Weise haben wir die Lösung für Lisas Problem gefunden, und eine «falsche» Kombination bzw. Bepflanzung des Beets konnte gar nicht entstehen.

Zum Glück liess sich die Lösung auf direktem Weg finden: Es gab immer Bereiche, in die wir einige der noch übrigen Gemüse pflanzen konnten. Das gelingt im Allgemeinen nicht immer.

Wenn man versucht, die Lösung schrittweise zusammenzusetzen, kann es bei einem Schritt A mehrere Möglichkeiten geben, alle Bedingungen zu erfüllen.



Je nach Wahl kann es bei einem späteren Schritt B keine Möglichkeit mehr geben. Dann nimmt man die letzten Schritte solange zurück, bis man beim Schritt A mit den mehreren Möglichkeiten wieder angekommen ist. Dort wählt man eine andere Möglichkeit und versucht damit eine Lösung zu finden.

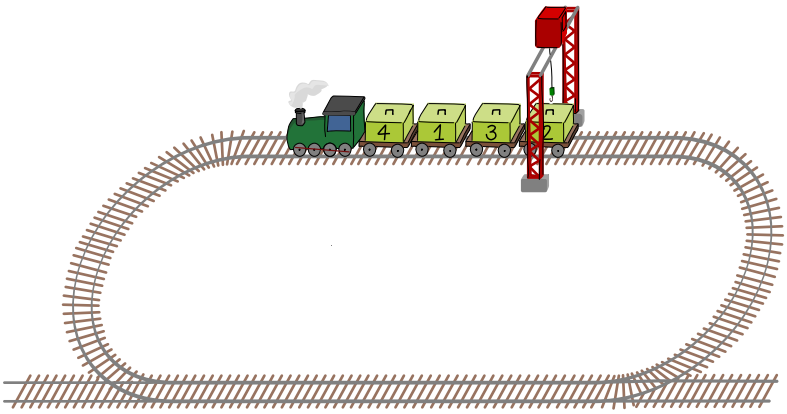
In der Informatik ist diese Rücknahme von Schritten als *Backtracking* bekannt.

Stichwörter und Webseiten

- Brute Force: `https://de.wikipedia.org/wiki/Brute-Force-Methode`
- Backtracking: `https://de.wikipedia.org/wiki/Backtracking`

10. Zug entladen

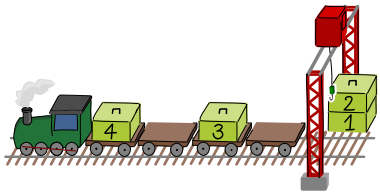
Ein Zug zieht Waggons mit nummerierten Kisten. Der Kran steht an einer festen Position und entlädt die Kisten. Um eine Kiste zu entladen, muss die Kiste direkt unter dem Kran positioniert werden.



Der Kran muss die Kisten von 1 an in der Reihenfolge ihrer Nummern entladen. Der Zug kann nur vorwärts fahren. Wenn er unter dem Kran hindurch gefahren ist, muss er eine Runde fahren, damit weitere Kisten entladen werden können.

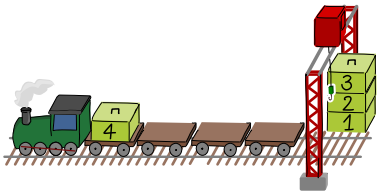
So entlädt der Kran die Kisten 1, 2, 3 und 4:

Runde 1:



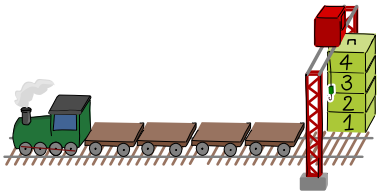
Er überspringt Kiste 4, entlädt Kiste 1, überspringt Kiste 3 und entlädt Kiste 2.

Runde 2:



Er überspringt Kiste 4 und entlädt Kiste 3.

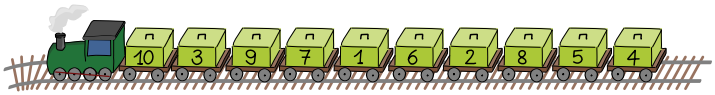
Runde 3:



Er entlädt Kiste 4.

Der obige Zug muss also drei Runden fahren, bis alle Kisten in der richtigen Reihenfolge entladen sind.

Wie viele Runden werden für das Entladen des folgenden Zuges benötigt?



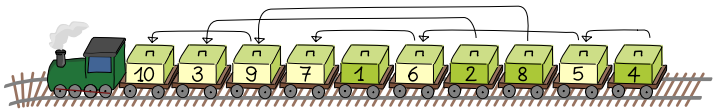
A) 1 Runde
B) 2 Runden
C) 3 Runden
D) 4 Runden
E) 5 Runden
F) 6 Runden
G) 7 Runden
H) 8 Runden
I) 9 Runden
J) 10 Runden

Lösung

Die richtige Antwort ist 7 Runden.

Die vorgeschriebene Reihenfolge für das Entladen ist 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. In der ersten Runde entlädt der Kran die Kisten 1 und 2 zusammen. In der zweiten Runde entlädt der Kran 3 und 4 zusammen, dann 5, dann 6, dann 7 und 8 zusammen, dann 9 und schliesslich 10. Dies entspricht 7 Runden.

Alternativ kann man sich zunutze machen, dass jedes Mal, wenn für eine der Kistennummern in der Folge die nächste Kistennummer links davon steht, eine zusätzliche Entladerunde erforderlich ist.



Da z. B. die 3 links von der 2 steht, wird sie übersprungen, um die 2 zu entladen, so dass eine zusätzliche Runde erforderlich ist, um die 3 unter den Kran zu bringen. Beim gegebenen Zug gibt es sechs solcher Paare (2,3), (4,5), (5,6), (6,7), (8,9) und (9, 10), so dass 6 zusätzliche Runden benötigt werden, also insgesamt 7 Runden.

Dies ist Informatik!

Wenn für eine beliebige Zahl in der Folge 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 die Kiste mit der nächstgrösseren Zahl weiter links auf dem Zug liegt, nennen wir dies eine *Inversion*, das bedeutet Umkehrung. Für jede solche Umkehrung ist eine zusätzliche Runde erforderlich. Wenn wir die Anzahl der Umkehrungen zählen, erhalten wir die Antwort.

Das Zählen von Inversionen in Bezug auf eine gewünschte Folge hat viele Anwendungen. Bei einigen *Sortieralgorithmen*, wie z. B. *Bubble-Sort*, gibt die Anzahl der Inversionen Aufschluss darüber, wie viele Vertauschungen erforderlich sind, um eine bestimmte Folge zu sortieren. Wenn zwei Kunden dieselbe Menge von Artikeln in eine Rangfolge bringen, sagt uns die Anzahl der Inversionen in ihren Ranglisten, wie sehr sich ihre Vorlieben gleichen. Dies wird von Online-Shops genutzt, um «ähnliche» Kunden zu identifizieren und ihnen Produktempfehlungen zu geben.

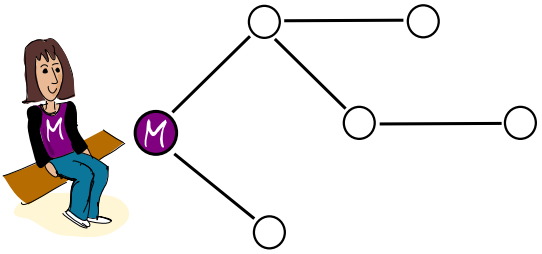
Stichwörter und Webseiten

- Sortierverfahren: `https://de.wikipedia.org/wiki/Sortierverfahren`
- Bubble Sort: `https://de.wikipedia.org/wiki/Bubblesort`
- Einfache Sortierverfahren: `https://hpi.de/friedrich/teaching/units/einfache-sortierverfahren.html`
- Inversion: `https://www.ziemke-koeln.de/unterricht/informatik/gk12/sortieren/suchen_sortieren.htm#Inversion`

11. Martinas Dorf

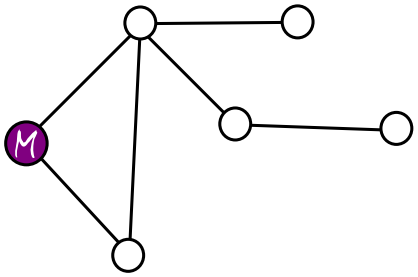
In Martinas Dorf gibt es sechs Häuser. Ausserdem gibt es Wege, über die man von einem Haus zum nächsten gehen kann. Für alle diese Wege benötigt Martina die gleiche Zeit.

Martina hat eine besondere Karte des Dorfs gezeichnet. Sie hat darin Wege eingezeichnet, über die sie am schnellsten zu den anderen Häusern gehen kann.

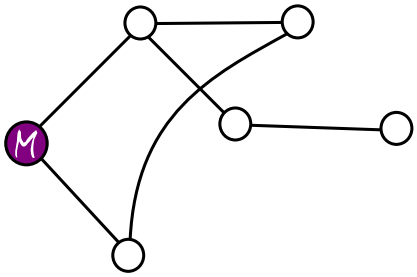


Natürlich gibt es auch eine richtige Karte des Dorfs, mit allen Wegen.

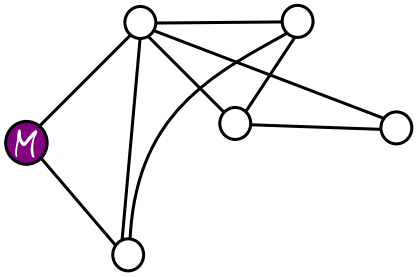
Welche dieser Zeichnungen kann nicht *die richtige Karte sein?*



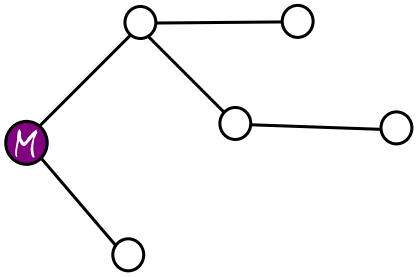
A)



B)



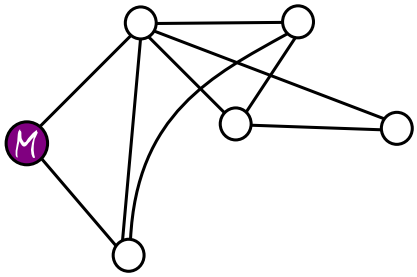
C)



D)

Lösung

Antwort C ist richtig:



Martinas besondere Karte zeigt, dass sie zu dem Haus ganz rechts am schnellsten über drei Wege gehen kann. Wenn C die richtige Karte des Dorfes wäre, dann könnte Martina schneller zu diesem Haus gehen, nämlich über zwei Wege. Also kann C nicht die richtige Karte des Dorfes sein.

Bei den Karten A, B und D gibt es keine Möglichkeit, schneller zu einem der anderen Häuser zu gehen als über die Wege auf Martina besonderer Karte. Diese Karten könnten also richtige Karten des Dorfs sein.

Dies ist Informatik!

Martina ist Informatikerin. Sie hat ihre Karte als *Graph* gezeichnet. Graphen bestehen aus *Knoten* (hier die Häuser), die durch *Kanten* (hier die Wege) verbunden sein können. Sie sind in vielen Bereichen der Informatik geeignet, die Realität zu modellieren – auch hier in dieser Biberaufgabe.

Martina weiss, dass es für Graphen eine ganze Reihe von Algorithmen gibt, beispielsweise die sogenannte Breitensuche, um Aufgaben zu lösen wie «Wie kommt man am schnellsten zu einem anderen Haus?». Vielleicht hat sie ihre besondere Dorfkarte mit Hilfe einer Breitensuche in einem grösseren Graph, der richtigen Karte des Dorfes mit allen Wegen, erstellt.

In der Graphentheorie, die sich mit Graphen und Graph-Algorithmen beschäftigt, entspricht Martinas Karte einem Teilgraph der Gesamtkarte des Dorfes. Martinas Teilgraph hat zwei Besonderheiten:

- Alle Knoten sind direkt (über eine Kante) oder indirekt (über mehreren Kanten) miteinander verbunden.
- Egal welche zwei Knoten man zufällig auswählt, es gibt immer nur genau einen Weg zwischen den beiden.

Ein Graph mit diesen Besonderheiten wird in der Informatik als *Baum* bezeichnet. Martinas Haus stellt die *Wurzel* des Baumes dar. Von der Wurzel aus kann Martina alle anderen Knoten (die anderen Häuser im Dorf) auf einem eindeutigen Weg erreichen. Martinas Teilgraph ist also ein Baum; ausserdem enthält er alle Knoten des gesamten Graphen (der Gesamtkarte des Dorfes) – aber möglicherweise nicht alle Kanten. Ein Teilgraph mit diesen Eigenschaften wird als *Spannbaum* des gesamten Graphen bezeichnet.

In der Informatik gibt es viele Anwendungen für Graph-Algorithmen, vor allem im Zusammenhang mit Netzwerken (Verkehrsnetze, Telekommunikationsnetze . . .), etwa bei der Berechnung von Routen

in Navigationssystemen. Spannbäume können beim Aufbau kostengünstiger Netze eingesetzt werden und hilfreich bei der Lösung besonders schwieriger Probleme sein.

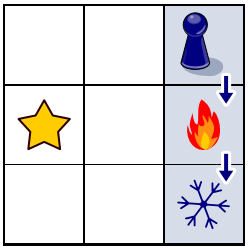
Stichwörter und Webseiten

- Graphentheorie: `https://de.wikipedia.org/wiki/Graph_(Graphentheorie)`
- Baum: `https://de.wikipedia.org/wiki/Baum_(Graphentheorie)`
- Breitensuche: `https://de.wikipedia.org/wiki/Breitensuche`
- Spannbaum: `https://de.wikipedia.org/wiki/Spannbaum`

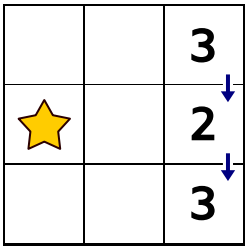
12. Wärmer und Kälter

Nina und Daniel spielen Schatzsuche. Auf einem Spielbrett mit quadratischen Feldern wählt Nina im Kopf ein Feld aus. Dort ist der Schatz versteckt.

Daniel wählt ein Startfeld aus. Von dort geht er schrittweise mit seiner Spielfigur um je ein Feld weiter: nach links, rechts, oben oder unten.



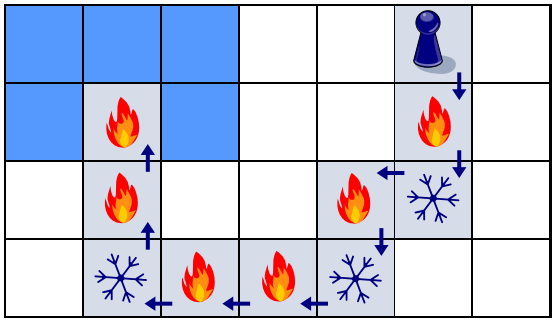
Beim ersten Versuch nehmen sie ein kleines Spielbrett. Nina versteckt den Schatz auf dem Feld mit dem Stern. Daniel startet rechts oben und macht zwei Schritte entlang der Pfeile. Nach jedem Schritt sagt Nina, ob Daniel nun näher am Schatz oder weiter weg vom Schatz ist als vor dem Schritt.



Das Bild rechts zeigt Daniels Entfernungen vom Schatz. Die Entfernung vom Schatz ist die kleinste Anzahl Schritte, mit denen Daniel aktuell zum Schatz gehen könnte.

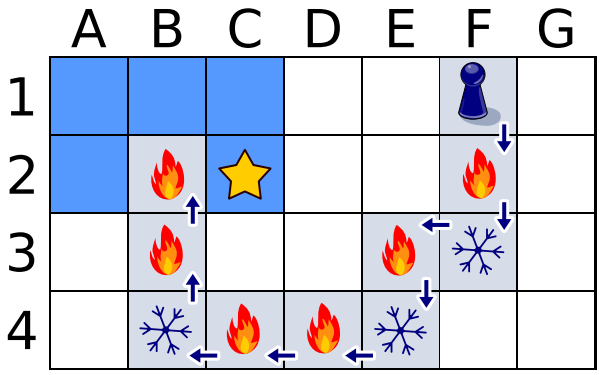
Nun nehmen sie ein grösseres Spielbrett. Nina versteckt den Schatz auf einem der blau markierten Felder. Das Bild zeigt wieder Daniels Schritte und was Nina nach jedem Schritt sagt.

Wo ist der Schatz versteckt?



Lösung

So ist es richtig:



Wir verfolgen Daniels Weg und Ninas Rückmeldungen. Daniel startet in Zeile 1 des Spielbretts. Nach dem ersten Schritt ist er in Zeile 2 und näher am Schatz als in Zeile 1. Nach dem nächsten Schritt ist er in Zeile 3 und wieder weiter weg vom Schatz. Da er in der gleichen Spalte geblieben ist, muss der Schatz auf einem Feld in Zeile 2 sein. Denn egal, in welcher Spalte der Schatz versteckt ist: Den kürzesten Weg zum Schatz von einer anderen Spalte aus hat man, wenn man in der gleichen Zeile ist.

Aber in welcher Spalte ist der Schatz versteckt? Auf seinem weiteren Weg kommt Daniel in Zeile 4 mit einigen Schritten nach links dem Schatz zunächst näher; insbesondere ist er in Spalte 3 näher am Schatz als in Spalte 4. Aber nach dem letzten Schritt in der Zeile ist Daniel in Spalte 2 weiter weg vom Schatz als in Spalte 3. Der Schatz muss also auf einem Feld in Spalte 3 sein. Denn was wir oben für Spalten gesagt haben, gilt auch für Zeilen: Den kürzesten Weg zum Schatz von einer anderen Zeile aus hat man, wenn man in der gleichen Spalte ist.

Dies ist Informatik!

Daniel geht (mit seiner Spielfigur) über das Spielbrett. Von jedem Feld, auf dem er gerade steht, misst Nina die Entfernung zum Feld mit dem Schatz und nutzt dies für ihre Rückmeldung. Üblicherweise wird die Entfernung zwischen zwei Punkten als Länge der geradlinigen Verbindung zwischen den Punkten gemessen (*euklidische Distanz*). Doch die zwei Felder sind genau genommen keine Punkte. Deshalb bemisst Nina die Entfernung zwischen zwei Feldern in der Anzahl von Schritten, die Daniel für einen kürzesten Weg vom einen Feld zum anderen gehen müsste. Dieses *Maß* kann man generell bei Rastern anwenden und ist in der Informatik als *Manhattan-Distanz* bekannt – abgeleitet vom gerasterten Straßenplan des New Yorker Stadtteils Manhattan. Daniel geht (mit seiner Spielfigur) über das Spielbrett. Von jedem Feld, auf dem er gerade steht, misst Nina die Entfernung zum Feld mit dem Schatz und nutzt dies für ihre Rückmeldung. Üblicherweise wird die Entfernung zwischen zwei Punkten als Länge der geradlinigen Verbindung zwischen den Punkten gemessen (*euklidische Distanz*). Doch die zwei Felder sind genau genommen keine Punkte. Deshalb bemisst Nina die Entfernung zwischen zwei Feldern in der Anzahl von Schritten, die Daniel für einen kürzesten Weg vom einen Feld zum anderen gehen müsste. Dieses *Mass* kann man generell bei Rastern anwenden

und ist in der Informatik als *Manhattan-Distanz* bekannt – abgeleitet vom gerasterten Strassenplan des New Yorker Stadtteils Manhattan.

Informatikerinnen und Informatiker wählen die Art der Distanzberechnung zwischen zwei Objekten in Abhängigkeit von der Frage, die sie lösen möchten. Wenn man zum Beispiel die Entfernung zwischen zwei gleich langen Wörtern in einer natürlichen Sprache messen möchte, kann man die Anzahl Stellen, an welchen sich die Wörter unterscheiden, zählen; das ist dann der *Hamming-Abstand* oder die *Hamming-Distanz*. Sind die Wörter unterschiedlich lang, kann man die *Editierdistanz* verwenden. Entfernungen bzw. Distanzen spielen in der Informatik häufig eine Rolle, wenn es darum geht, optimale Lösungen eines Problems zu finden. Egal ob die Lösung eines Problems am schnellsten, am kürzesten oder am günstigsten sein soll: Man muss oft nicht den Algorithmus ändern, sondern nur das Entfernungsmass: Dauer, Länge oder Kosten.

Stichwörter und Webseiten

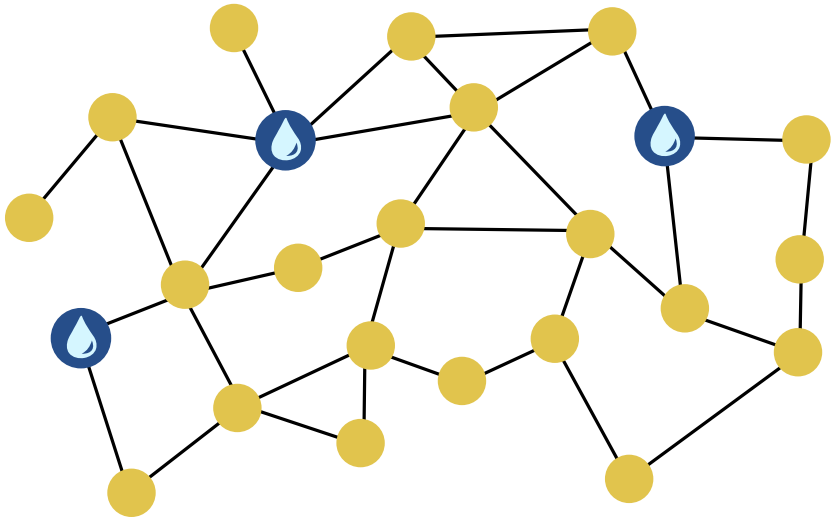
- Manhattan-Distanz: `https://de.wikipedia.org/wiki/Manhattan-Metrik`
- Hamming-Abstand: `https://de.wikipedia.org/wiki/Hamming-Abstand`
- Editierdistanz: `https://de.wikipedia.org/wiki/Levenshtein-Distanz`

13. Brunnen

Der Sommer in der Stadt ist heiss. Die Bürgermeisterin lässt deshalb Brunnen mit Trinkwasser aufstellen.

Die Brunnen sollen so stehen, dass man von jeder Strassenecke aus höchstens zwei Strassenabschnitte gehen muss, um einen Brunnen zu erreichen. Dann ist die Bürgermeisterin zufrieden.

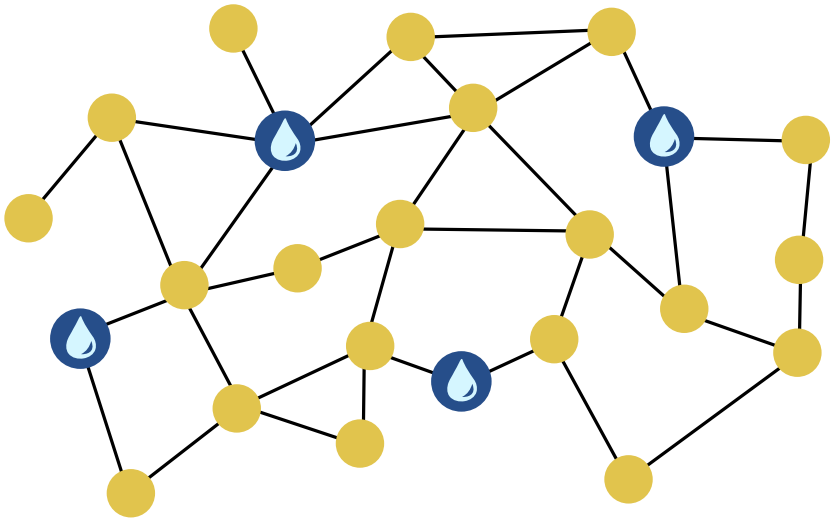
Hier ist ein Stadtplan. Die Linien sind Strassenabschnitte, und die Punkte sind Strassenecken. An drei Ecken stehen bereits Brunnen .



Stelle einen weiteren Brunnen so auf, dass die Bürgermeisterin zufrieden ist.

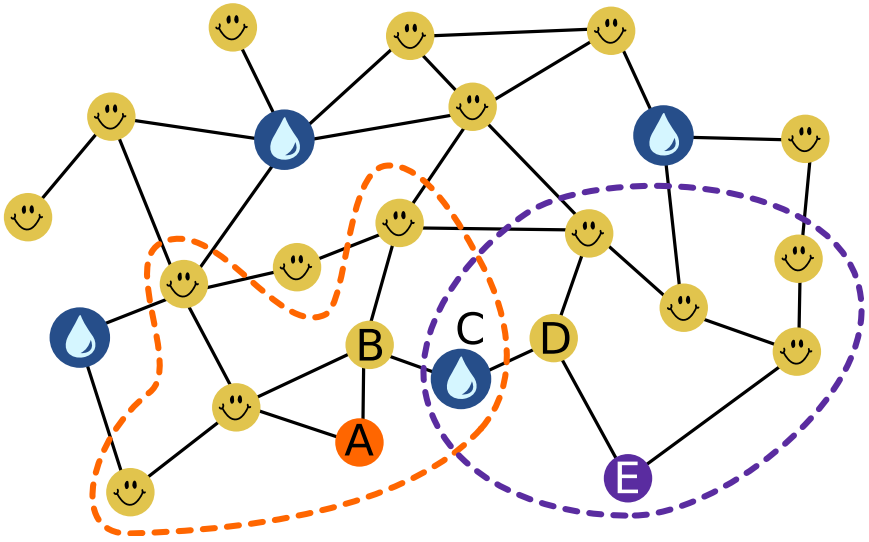
Lösung

So ist es richtig:



Wenn ein weiterer Brunnen unten in der Mitte aufgestellt wird, muss man von jeder Strassenecke aus höchstens zwei Strassenabschnitte gehen, um einen Brunnen zu erreichen. Dann ist die Bürgermeisterin zufrieden.

Wie können wir herausfinden, an welcher Strassenecke ein weiterer Brunnen aufgestellt werden soll? Im Stadtplan markieren wir alle Strassenecken mit einem  , die höchstens zwei Strassenabschnitte von einem der Brunnen entfernt sind, die bereits aufgestellt sind. In Bezug auf diese Ecken kann die Bürgermeisterin bereits zufrieden sein.



Für die fünf übrigen Strassenecken A, B, C, D und E stellen wir einen weiteren Brunnen bei C auf. Damit muss man auch von diesen Ecken höchstens zwei Strassenabschnitte zum nächsten Brunnen gehen.

Die Ecke C ist die einzige Stelle für einen neuen Brunnen, die das ermöglicht: Wenn wir für die Ecken A und E jeweils alle anderen Ecken betrachten, die über zwei Strassenabschnitte erreichbar sind (im Bild mit gestrichtelten Linien umrandet), ist die Strassenecke C die einzige, die diese Bedingung für A *und* E erfüllt.

Dies ist Informatik!

Der Stadtplan kann als *Graph* modelliert werden. Das ist ein für die Informatik wichtiges Werkzeug, um Beziehungen zwischen Objekten zu modellieren und Fragen in Bezug auf diese Beziehungen zu beantworten. Hier kann man die Strassenecken als Objekte und damit *Knoten* des Graphen auffassen. Die Beziehung zwischen zwei Objekten wird im Graph durch *Kanten* modelliert, die man als Verbindungslinien darstellt. Hier bedeutet eine Kante zwischen zwei Strassenecken, dass sie durch einen Strassenabschnitt verbunden sind. Diese Beziehung kann man Nachbarschaft nennen. Kanten können aber auch andere Beziehungen modellieren, wie z.B. Freundschaft.

In dieser Biberaufgabe soll eine Teilmenge der Knoten gefunden werden (zum Aufstellen der Brunnen), so dass jeder Knoten ausserhalb dieser Teilmenge über einen Weg mit einem «Brunnen-Knoten» verbunden ist, der höchstens zwei Kanten lang ist. In der Fachsprache der Informatik würde dies als Suche nach einem «distance−2 dominating set» bezeichnet. Im allgemeinen (für alle Weglängen $k \geq 1$) gehört diese Suche nach einer möglichst kleinen solchen Teilmenge zu den schwierigsten Problemen der Informatik.

Solche «minimum distance k-dominating sets» spielen in der letzten Zeit eine grössere Rolle, insbesondere im Bereich des *Social Computing* (auf Deutsch auch *Sozioinformatik*): Zur automatischen Verarbeitung von Daten über soziale Netzwerke (etwa um die Verbreitung von Fake News zu erkennen) werden die Fan- oder Follower-Beziehungen zwischen den Nutzern als Graph modelliert. Diese Graphen können so gross sein, dass nur eine (möglichst kleine) repräsentative Auswahl von Nutzern betrachtet werden kann - zum Beispiel ein «minimum distance 3-dominating set». Da die wirklich kleinste Auswahl nicht effizient berechnet werden kann, entwickelt die Informatik Verfahren, die in kurzer Zeit möglichst kleine, aber nicht garantiert kleinste Auswahlen berechnen.

Stichwörter und Webseiten

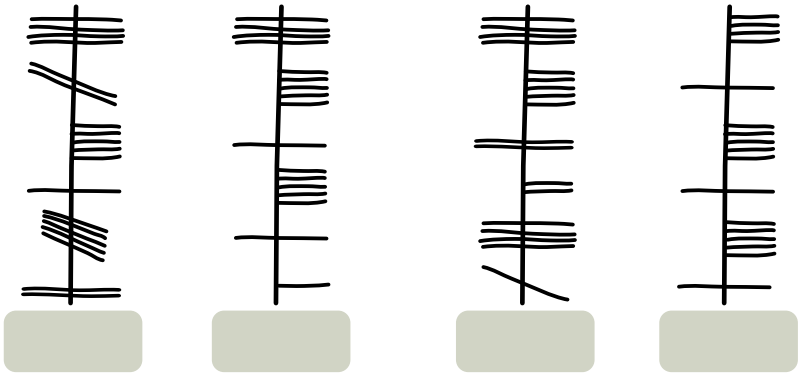
- Minimum Distance k-Dominating Sets: `https://computationalsocialnetworks.springeropen.com/articles/10.1186/s40649-020-00078-5`
- Sozioinformatik: `https://de.wikipedia.org/wiki/Sozioinformatik`

14. Ogham

Sue kennt das alte irische Alphabet Ogham. Jeder Buchstabe besteht aus einem oder mehreren Strichen, die entlang einer langen Linie angeordnet sind. Zwei aufeinander folgende Buchstaben werden durch einen Zwischenraum getrennt.

Sue benutzt Ogham als Code. Sie kodiert vier Wörter – ihre liebsten Fruchtsorten:
ANANAS, BANANE, MELONE und ORANGE.

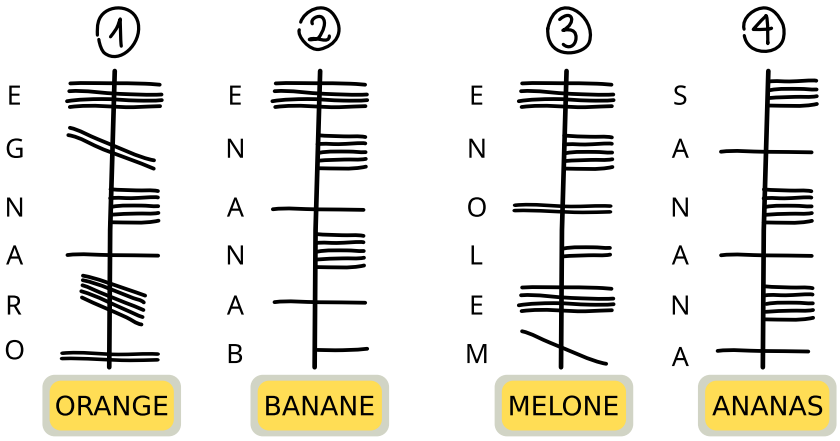
Welches Wort passt zu welchem Ogham-Code?



ANANAS BANANE MELONE ORANGE

Lösung

So ist es richtig:



Es gibt verschiedene Möglichkeiten, die richtige Zuordnung zu bestimmen. Auf jeden Fall aber muss herausgefunden werden, in welcher Richtung die Buchstaben entlang der Linie geschrieben werden. Dabei hilft das besonders markante Wort ANANAS. Darin kommt der Buchstabe A dreimal vor, mit jeweils einem anderen Buchstaben dazwischen.

Nur im Ogham-Code 4 kommt ein Buchstabe dreimal vor, und auch dort ist jeweils ein Buchstabe dazwischen. Code 4 ist also der einzige, zu dem das Wort ANANAS passt. So erkennt man, dass man in Ogham Wörter von unten nach oben schreibt und dass der dreifach vorkommende Buchstabe A in Ogham als horizontaler Strich durch die Linie geschrieben wird.

Dieser Ogham-Buchstabe A kommt nur im Code 2 zweimal vor. Auch wegen der aus ANANAS bekannten Kodierung von N (fünf horizontale Striche rechts von der Linie) und der Anordnung der weiteren Buchstaben passt nur BANANE zu diesem Code. ORANGE passt nur zum Code 1; unter anderem, weil man dort den Ogham-Buchstaben A genau einmal findet. Nun ist nur noch Code 3 übrig; er muss also das Ogham-Wort für MELONE sein und enthält die von den übrigen Wörtern bekannten Ogham-Buchstaben E und N an den passenden Stellen.

Dies ist Informatik!

In dieser Biberaufgabe muss ein unbekannter Text entschlüsselt bzw. dechiffriert werden. Das ist hier nicht sehr schwierig, weil der entschlüsselte *Klartext* bekannt ist. Ausserdem ist der unbekannte Text auf gleiche Weise in Buchstaben und Wörter eingeteilt wie der bekannte Text. Beim Dechiffrieren eines geheimen Textes bzw. eines Textes in unbekannter Schrift, dessen Klartext nicht bekannt ist, hilft es in diesem Fall oft, sich Gedanken über die Häufigkeit von Buchstaben und Wörtern zu machen und auf dieser Grundlage zu versuchen, sie im Text zu finden. Auf diese Weise sind einige antike Alphabete und Schriften entschlüsselt worden. Schwierig wird es aber, wenn die Schriftzeichen im unbekannten Text nicht so einfach den Buchstaben und Wörtern der bekannten Sprache zuzuordnen sind wie im Fall von Ogham. Dann hilft oft nur der Abgleich mit bekannten Texten oder Schriften, wie in dieser Biberaufgabe. Zum Beispiel wurden die ägyptischen Hieroglyphen jahrhundertelang

nicht entschlüsselt, bis durch einen Zufall ein Stein mit Hieroglyphen und zwei bekannten Schriften gefunden wurde, der Stein von Rosetta. Auf dem Stein fand sich dreimal der gleiche Text. Der war zwar in verschiedenen Sprachen geschrieben, enthielt aber immer dieselben Namen. So konnten wesentliche Elemente der Hieroglyphen entschlüsselt werden. Das gilt aber nicht für alle Schriften: Noch immer sind die etwa 650 Schriftzeichen der Maya-Kultur nicht vollständig entschlüsselt, genau so wenig wie die Schriften Linearschrift A und Linearschrift B aus der Mittelmeerregion.

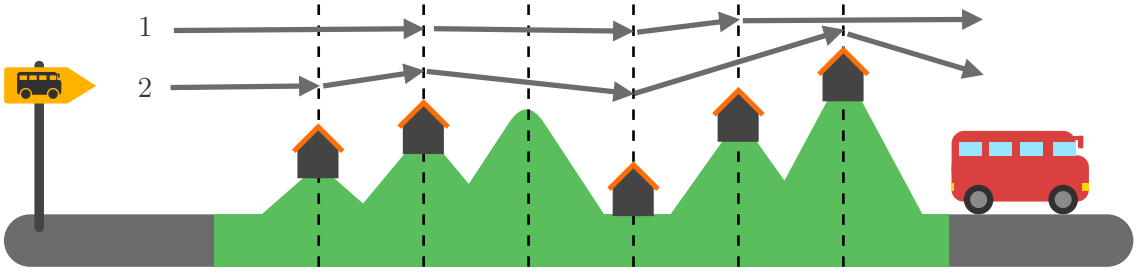
Auch in der Informatik werden Schriftzeichen und Texte entschlüsselt – nachdem sie vorher zur abhörsicheren Datenübertragung verschlüsselt wurden. Dazu werden aber ganz andere Verfahren verwendet als bei der Kodierung von Wörtern in anderen Schriften. Solche einfachen Kodierungen sind insbesondere mit Hilfe von Computern zu leicht zu entschlüsseln, meist mit Hilfe der oben schon genannten Überlegungen zur Häufigkeit von Buchstaben und Wörtern.

Stichwörter und Webseiten

- Kryptographie: `https://de.wikipedia.org/wiki/Kryptographie`
- Kryptoanalyse: `https://de.wikipedia.org/wiki/Kryptoanalyse`
- Ogham: `https://de.wikipedia.org/wiki/Ogham`

15. Wanderungen

Mia mag Wanderurlaube, bei denen sie jede Nacht an einem anderen Ort übernachtet. Für ihren nächsten Urlaub hat Mia eine Karte der Region. Die Karte zeigt Mias Startpunkt , ihr Ziel und alle Orte, an denen sie übernachten kann .



Mia hat die Region mit gestrichelten Linien in Abschnitte eingeteilt. Sie kann immer nur einen oder zwei Abschnitte an einem Tag wandern. Zwei verschiedene Wanderungen, die sie machen kann, hat sie bereits in die Karte eingetragen:

- Wanderung 1 hat drei Übernachtungsorte
- Wanderung 2 hat vier Übernachtungsorte

Mia kann aber noch andere Wanderungen machen.

Wie viele verschiedene Wanderungen kann Mia insgesamt machen? Zähle die Wanderungen 1 und 2 mit.

A) 2 Wanderungen

B) 3 Wanderungen

C) 4 Wanderungen

D) 5 Wanderungen

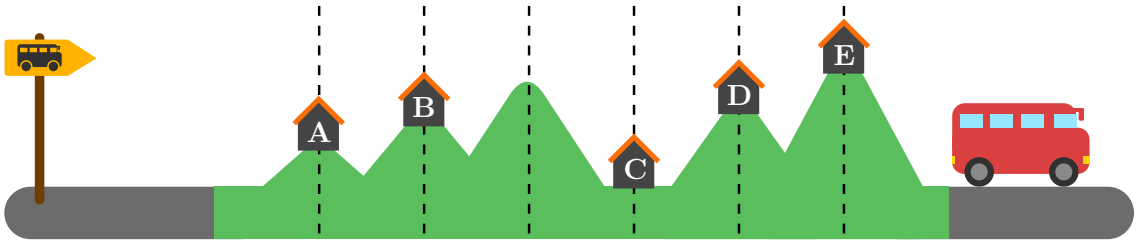
E) 6 Wanderungen

F) 7 Wanderungen

G) 8 Wanderungen

Lösung

Die richtige Antwort ist E) 6 Wanderungen.



Zuerst stellen wir fest, dass Mia in **B** und **C** übernachten muss, weil die Entfernung zwischen diesen beiden Orten die grösste Entfernung (2) ist, die sie an einem einzigen Tag zurücklegen kann. Für den Weg von **B** nach **C** hat Mia also nur eine Möglichkeit.

Nun können wir die Möglichkeiten für die anderen Teilstücke ihres Weges ermitteln: Vom Startpunkt () bis **B** kann Mia entweder in einem Stück durchwandern oder zwischendurch in **A** übernachten; das sind zwei Möglichkeiten (wie in den Wanderungen 1 und 2). Von **C** zum Ziel () muss Mia drei Abschnitte wandern, und sie kann nach jedem Abschnitt übernachten. Deshalb kann sie den gesamten Weg in alle drei Kombinationen von 1 und 2 Abschnitten aufteilen:

C → **D** → **E** →  ;
C → **E** →  ;
C → **D** →  .

Die Gesamtzahl aller Wanderungen, die Mia machen kann, ist also $2 \times 1 \times 3 = 6$.

Dies ist Informatik!

Manchmal kann die Zahl aller Möglichkeiten, eine gegebene Aufgabe zu erledigen, sehr gross sein. Zum Beispiel gibt es etwa 14 Millionen Möglichkeiten, 6 verschiedene Zahlen aus den Zahlen 1 bis 49 auszuwählen. Und es gibt etwa eine halbe Milliarde Möglichkeiten, die Zahlen von 1 bis 12 in unterschiedlicher Folge aufzuschreiben. Dafür braucht dann auch ein Computer ein wenig Zeit.

Wie gut, dass es in dieser Biberaufgabe nach dem dritten Abschnitt keinen Übernachtungsort gibt und das Zählen aller Wanderungen, die Mia machen kann, in drei Teile aufgeteilt werden kann. Das Zählproblem wird sozusagen in drei kleinere Zählprobleme zerlegt. In der Informatik wird die Technik der *Problemzerlegung* (engl.: *decomposition*) beim Entwurf von Algorithmen häufig verwendet. Dieses Lösungsprinzip ist auch als *Divide and Conquer* (auf Deutsch auch «Teile und herrsche») bekannt.

Nach diesem Prinzip funktionieren zum Beispiel einige wichtige Sortieralgorithmen. Auch die *dynamische Programmierung*, eine Methode zur algorithmischen Lösung von von Optimierungsproblemen (beschrieben 1957 von Richard Bellman), basiert auf diesem Prinzip: Wenn man erkennt, dass die optimalen Lösungen eines Problems sich aus den optimalen Lösungen von Teilproblemen zusammensetzen, kann man dies nutzen, um sozusagen «klein anzufangen»: Zunächst werden die Lösungen für die kleinsten Teilprobleme direkt berechnet und anschliessend zu Lösungen für die nächstgrösseren

Teilprobleme zusammengesetzt. Dies wird wiederholt, bis die optimale Lösung des vollständigen Problems gefunden ist. Da gefundene Teil-Lösungen häufig zu Lösungen vieler grösserer Teile beitragen, werden sie gespeichert, um wiederholte gleiche Berechnungen einzusparen. Auch beim Zählen von Möglichkeiten kann dynamische Programmierung sehr hilfreich sein.

Stichwörter und Webseiten

- Problemzerlegung, Decomposition
- Divide and Conquer / Teile und herrsche:
  `https://de.wikipedia.org/wiki/Teile-und-herrsche-Verfahren`
- dynamische Programmierung:
  `https://de.wikipedia.org/wiki/Dynamische_Programmierung`

16. Emma erledigt

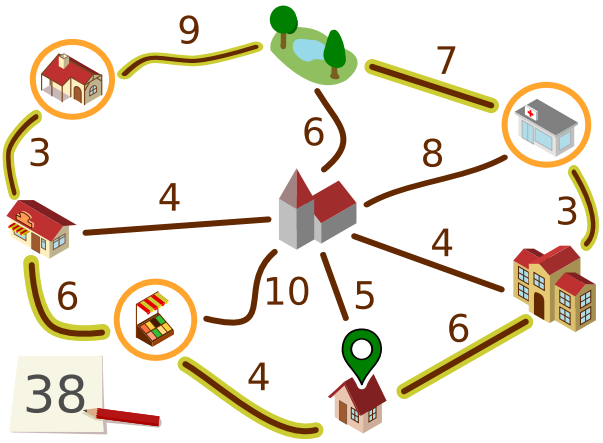
Emma ist zu Hause. Sie soll drei Aufgaben erledigen und zurückkommen:

- beim Kiosk ein Päckchen abholen,
- auf dem Markt Obst kaufen und
- in der Apotheke ein Medikament besorgen.

Emma weiss nicht, wie lange sie in jedem Geschäft brauchen wird. Aber zumindest ihr Weg soll so kurz wie möglich sein.

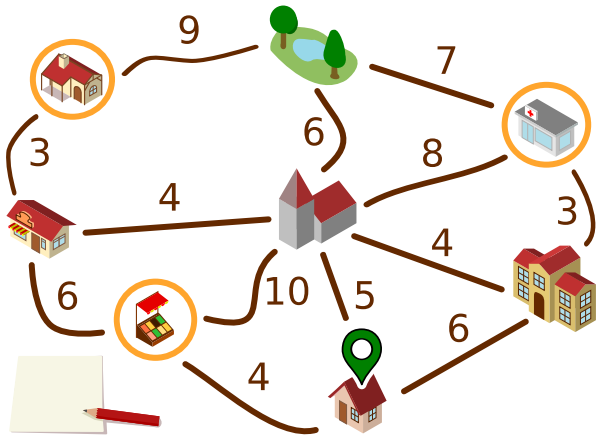
Auf einem Plan hat Emma eingetragen, wie viele Minuten sie für die Strecken zwischen einzelnen Orten ihrer Stadt benötigt. Ausserdem hat sie im Plan markiert, welche Strecke sie auf ihrem Weg geht.

Für diesen Weg benötigt Emma insgesamt $6 + 3 + 7 + 9 + 3 + 6 + 4 = 38$ Minuten.



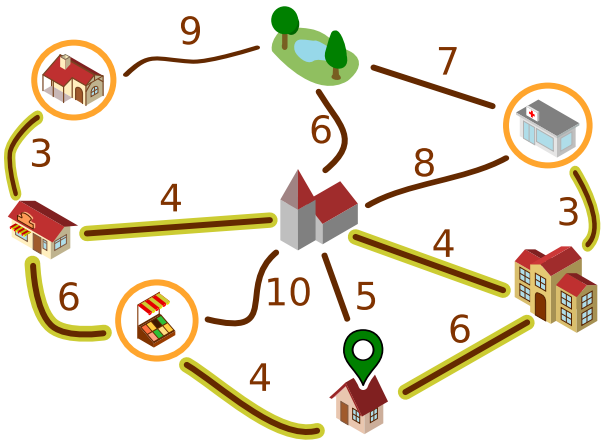
Emma überlegt, ob es noch schneller geht. Vielleicht hilft es, manche Strecken hin und zurück zu gehen?

Bestimme den kürzesten Weg, den Emma gehen kann, um ihre drei Aufgaben zu erledigen.

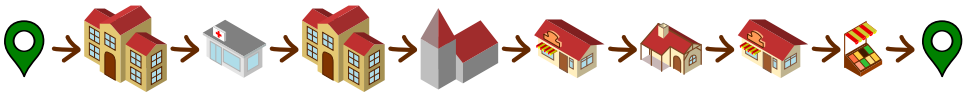


Lösung

So ist es richtig:

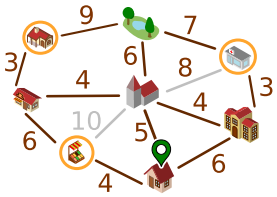


Emma kann so entlang der ausgewählten Strecken gehen (oder in die Gegenrichtung):

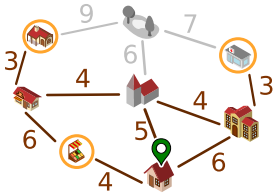


Für diesen Weg braucht sie $6 + 3 + 3 + 4 + 4 + 3 + 3 + 6 + 4 = 36$ Minuten.

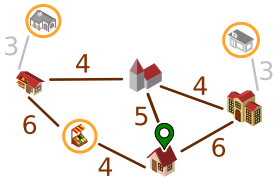
Nun wollen wir begründen, warum es keinen noch kürzeren Weg geben kann. Dazu benutzen wir eine vereinfachte Darstellung des Plans.



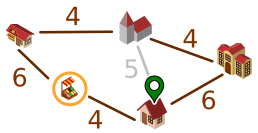
Die grau gezeichneten Strecken können wir ignorieren. Es gibt kürzere Wege zwischen den durch die Strecken verbundenen Orte, nämlich über andere Orte.



Auch den Park können wir ignorieren. Emma muss nicht zum Park. Zudem gibt es für jeden Weg, der über den Park geht, eine kürzere Alternative.

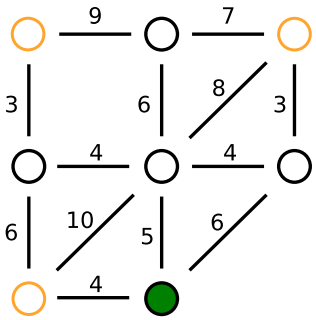


Emma muss zur Apotheke und zum Kiosk gehen. Dorthin kommt sie jeweils nur von der Bäckerei bzw. der Schule. Sie muss jeweils die Strecke zwischen diesen Orten hin- und her gehen. Das dauert jeweils $3 + 3 = 6$, ingesamt also 12 Minuten. Das merken wir uns und fassen nun die beiden Orte oben mit denen darunter zu einem zusammen.
Nun bleibt nur noch der Plan rechts übrig. Start und Ende des Weges ist hier . Diese drei Orte ( ) müssen besucht werden. Der kürzeste Weg, der das erfüllt, geht über alle fünf Orte und entlang aller Strecken ausser der grauen und dauert $4 + 6 + 4 + 4 + 6 = 24$ Minuten. Mit den 12 Minuten von oben macht das 36 Minuten. Die vorherigen Überlegungen zeigen, dass es keinen kürzeren Weg geben kann.



Dies ist Informatik!

Für die Begründung der richtigen Antwort wurde eine vereinfachte Darstellung des Plans benutzt. Es wäre möglich gewesen, den Plan noch deutlich abstrakter darzustellen:



Diese Darstellung enthält alle für Emmas Weg wichtigen Informationen, nämlich

- Objekte: die Orte, wobei die für den Weg wichtigen Orte markiert sind;

- und Beziehungen zwischen den Objekten: die Strecken zwischen den Orten, für die jeweils eine Länge angegeben ist.

Ein wichtiges Werkzeug zur Modellierung von Beziehungen zwischen Objekten sind *Graphen*. Graphen bestehen aus Knoten (für die Objekte) und Kanten (Paare von Objekten, für die Beziehungen). Emmas Plan lässt sich als *gewichteter Graph* modellieren, bei denen die einzelnen Beziehungen mit Zahlenwerten (den *Gewichten*) versehen werden.

Die Informatik interessiert sich für Fragen, die in Bezug auf Graphen gestellt werden können, und für Algorithmen, mit denen man die Fragen beantworten kann. Eine für gewichtete Graphen bedeutsame Frage lautet: Was ist der kürzeste (oder schnellste) Weg zwischen zwei Knoten? Die «Graphen-Frage» in dieser Biberaufgabe ist ähnlich: Was ist der kürzeste Rundweg von einem Knoten aus, bei dem eine Menge anderer Knoten besucht werden? Die Informatik kennt viele Algorithmen, die kürzeste Wege in Graphen effizient bestimmen können. Solche Algorithmen werden zum Beispiel in Software zur Routenplanung implementiert.

Stichwörter und Webseiten

- Weg (in Graphen): `https://de.wikipedia.org/wiki/Weg_(Graphentheorie)`
- Kürzeste Wege: Abenteuer Informatik, Kapitel 1 (`http://abenteuer-informatik.de/dasbuch.html`)

A. Aufgabenautoren

Somayah Albaradei
Laila Alharthi
Aldrich Ellis Catapang Asuncion
Leonardo Barichello
Liam Baumann
Wilfried Baumann
Javier Bilbao
Diego César
Sarah Chan
Marios Omar Choudary
Eimear Colreavy
Kris Coolsaet
Valentina Dagienė
Darija Dasović
Christian Datzko
Justina Dauksaite
Nora A. Escherle
Gerald Futschek
Bence Gaál
Emily Gates
Juan Gutiérrez
Josefine Hiebler
Mathias Hiron
Alisher Ikramov
Hyun-seok Jeon
Merel Kämper
David Khachatryan
Vaidotas Kinčius
Mhairi King
Jia-Ling Koh
Sophie Koh
Víctor Koleszar
Taina Lehtimäki
Angélica Herrera Loyo
Carlos Luna
Yong Mao
Yoshiaki Matsuzawa
Madhavan Mukund
Natalia Natalia
Tom Naughton
Marika Parviainen
Jean-Philippe Pellet
Zsuzsa Pluhár
Wolfgang Pohl
Estela Ramić
Chris Roffey
Kirsten Schlüter
Eljakim Schrijvers
Giovanni Serafini
Alieke Stijf
Marianne Thut
Monika Tomcsányiová

Svetlana Unković

Florentina Voboril

Michael Weigend

Manuel Wettstein

Kyra Willekes

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

C. Sponsoring

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/`

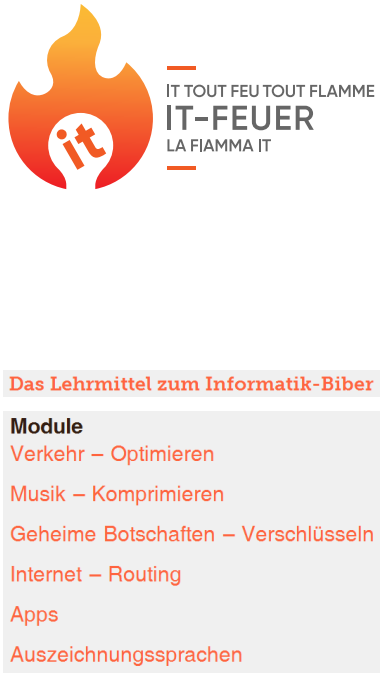
i-factory (Verkehrshaus Luzern)
Die i-factory bietet ein anschauliches und interaktives Erproben von vier Grundtechniken der Informatik und ermöglicht damit einen Erstkontakt mit Informatik als Kulturtechnik. Im optischen Zentrum der i-factory stehen Anwendungsbeispiele zur Informatik aus dem Alltag 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

D. 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.



`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.