INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Aufgaben und Lösungen 2023

Schuljahre 11/12/13

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

Herausgeber:

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

SVIA
www.svia-ssie-ssii.ch
schweizerischerverein für informatikinderausbildung//société suisse pour l'informatiquedans l'enseignement//società svizzera per l'informatica nell'insegnamento

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:

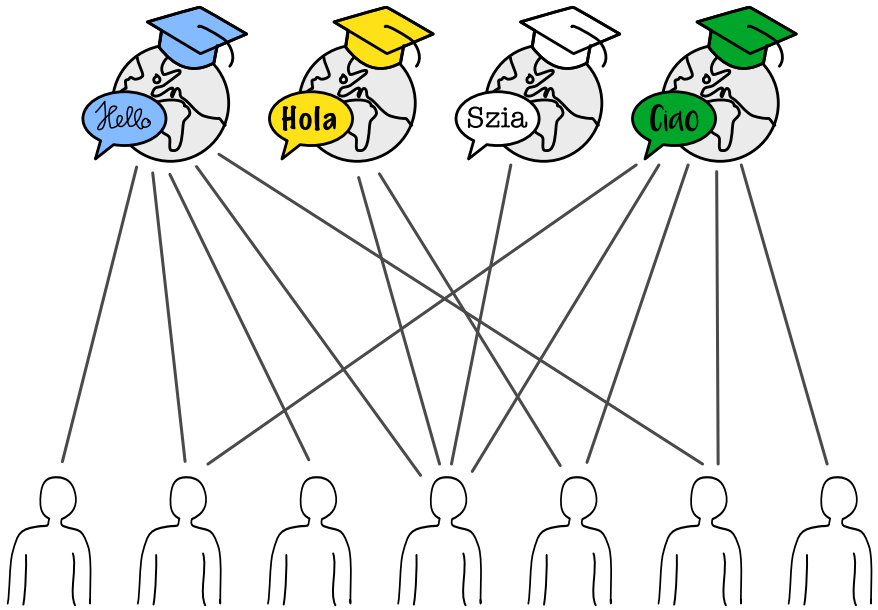
SVIA-SSIE-SSII Schweizerischer Verein für Informatik in der Ausbildung
Informatik-Biber
Nora A. Escherle

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

Inhaltsverzeichnis

1. Sommerkurse

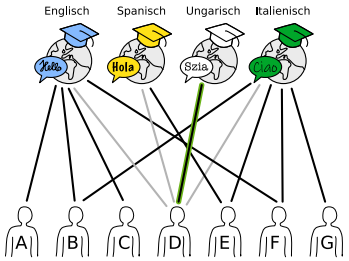
Eine Sprachschule plant vier Sommerkurse. Die Linien im Bild zeigen, welche Lehrperson der Schule für welchen Kurs geeignet ist.



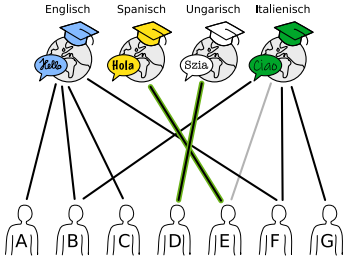
Eine Lehrperson kann nur einen Kurs halten. Trotzdem gibt es mehrere Möglichkeiten, jedem Kurs eine geeignete Lehrperson zuzuordnen.

Ordne jedem Kurs eine geeignete Lehrperson zu. Markiere dazu die Linie zwischen Person und Kurs.

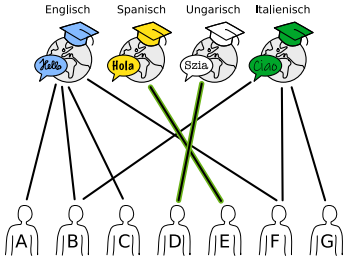
Lösung



D ist die einzige Lehrperson, die für den Ungarischkurs geeignet ist. Sie muss diesem Kurs zugeordnet werden und kann keine weiteren Kurse übernehmen.



E ist jetzt die einzige Lehrperson, die für den Spanischkurs geeignet ist. Sie muss diesem Kurs zugeordnet werden und kann keine weiteren Kurse übernehmen.



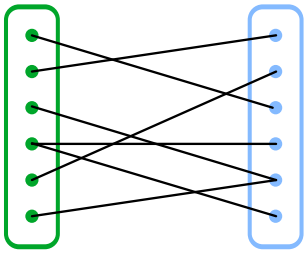
Bei den beiden verbleibenden Kursen (Englisch und Italienisch) kann man recht frei wählen. B und F dürfen aber nur einem Kurs zugeordnet werden, auch wenn sie für beide geeignet sind.

Dadurch gibt es insgesamt 10 Möglichkeiten, jedem Kurs eine geeignete Lehrperson zuzuordnen:

Englisch	**Italienisch**	**Ungarisch**	**Spanisch**
A	B	D	E
A	F	D	E
A	G	D	E
B	F	D	E
B	G	D	E
C	B	D	E
C	F	D	E
C	G	D	E
F	B	D	E
F	G	D	E

Dies ist Informatik!

Ein *Graph* besteht aus *Knoten* (Punkten), die durch *Kanten* (Linien) verbunden sind. Eine spezielle Klasse von Graphen sind *bipartite Graphen*: Die Knoten lassen sich in zwei getrennte Teilmengen teilen, sodass es nur Kanten zwischen Knoten verschiedener Teilmengen gibt.



Die Situation in dieser Biberaufgabe kann durch einen bipartiten Graphen dargestellt werden: Eine Teilmenge besteht aus den Kursen und die andere aus den Lehrpersonen. Bipartite Graphen eigenen sich sehr gut, *Zuordnungsprobleme* zu modellieren und zu lösen. Zuordnungsprobleme begegnen uns häufig im Alltag, z.B. bei Stundenplänen oder bei der Verteilung von Arbeit an Angestellte oder Maschinen. Bei kleineren Problemen ist es einfach möglich, eine optimale Zuordnung zu finden; bei grösseren wird es jedoch relativ schnell sehr komplex. Aus diesem Grund wurden in der Informatik verschiedene Algorithmen entwickelt, um möglichst schnell möglichst viele passende Paare zu finden.

Zum Beispiel wird auch das sogenannte Heiratsproblem mithilfe eines bipartiten Graphen dargestellt. Dabei steht eine Menge von heiratswilligen Männern einer Menge von heiratswilligen Frauen gegenüber. Ziel des Verfahrens ist, unter Berücksichtigung der jeweiligen Wünsche, alle Männer beziehungsweise alle Frauen zu verheiraten. Der englische Mathematiker Philip Hall formulierte im Heiratssatz 1935 die Bedingungen, unter denen so eine Zuordnung möglich ist.

In unserer Variante geht es nicht um diese vollständige Zuordnung, sondern darum, möglichst jeden Knoten der einen Teilmenge (die Kurse) einem Knoten der anderen Teilmenge zuzuordnen.

Stichwörter und Webseiten

- Bitpartierter Graph: `https://de.wikipedia.org/wiki/Bipartiter_Graph`
- Zuordnungsproblem: `https://de.wikipedia.org/wiki/Zuordnungsproblem`
- Programm zur Lösung der Aufgabe:
  `https://www.coding4you.at/dachu_2023/ir02/index.html`

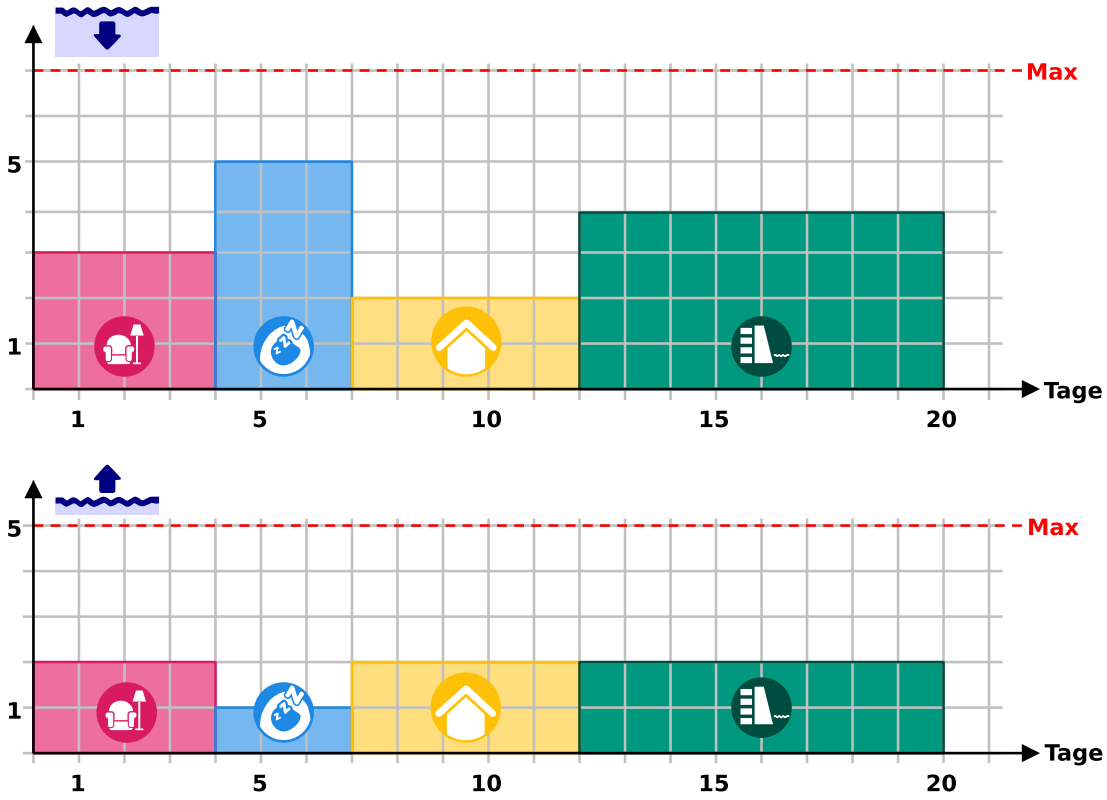
2. Biberburg AG

Eine Biberburg besteht aus 4 Teilen, die alle teilweise unter und teilweise über Wasser liegen. Beim Bau einer Biberburg ist jeder beteiligte Arbeiter entweder nur unter Wasser oder nur über Wasser tätig. Bei jedem Teil wird gleichzeitig über und unter Wasser gearbeitet. Die Tabelle zeigt für jedes Teil, wie lange die Biberbau AG braucht und wie viele Arbeiter unter und über Wasser dafür benötigt werden.

Teile	**Wohnraum**	**Schlafhöhle**	**Dach**	**Damm**
Baudauer	4 Tage	3 Tage	5 Tage	8 Tage
	3 Biber	5 Biber	2 Biber	4 Biber
	2 Biber	1 Biber	2 Biber	2 Biber

Das Dach kann erst gebaut werden, wenn die Schlafhöhle fertig ist! Bei allen anderen Teilen ist die Reihenfolge egal.

Für den Bau einer neuen Burg stehen höchstens 7 Unterwasser-Arbeiter und 5 Überwasser-Arbeiter zur Verfügung. Sie können auch gleichzeitig verschiedene Teile bauen. Hier ist ein Arbeitsplan, mit dem die Biberburg in 20 Tagen fertig wird.

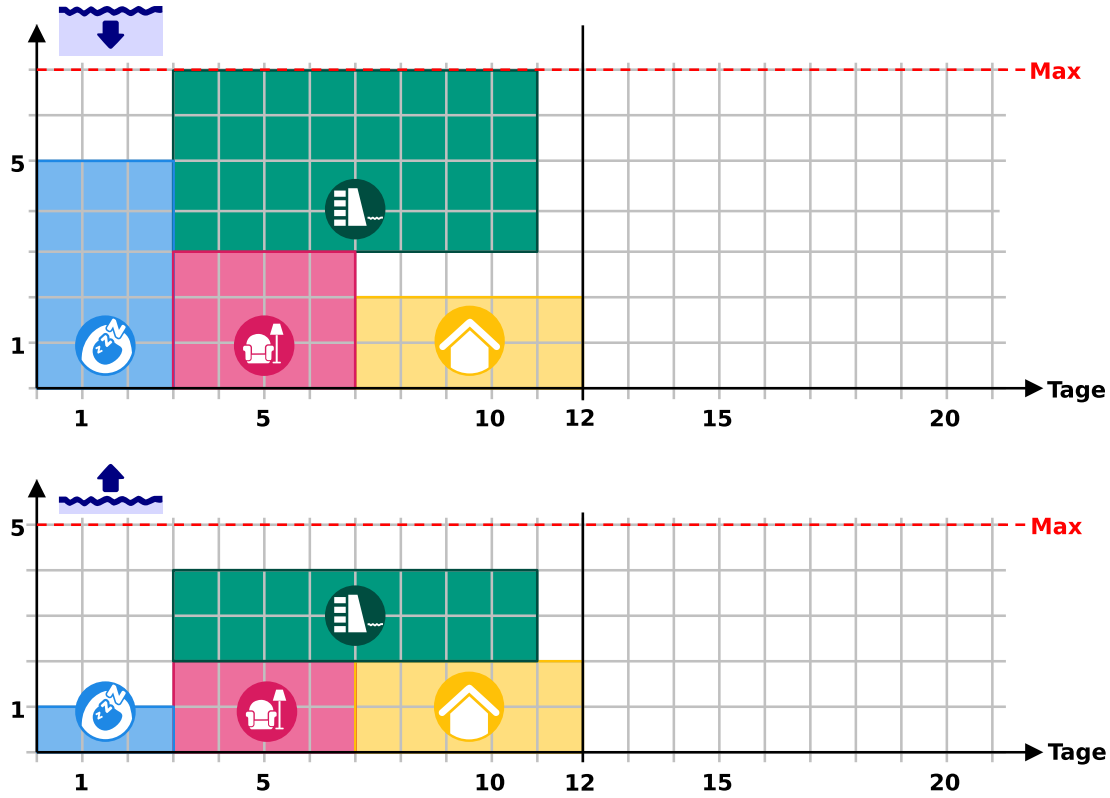


Überlege dir einen Plan, mit dem die Biberburg nach möglichst wenigen Tagen fertig wird. Wie viele Tage sind das?

Lösung

12 Tage ist die richtige Antwort.

Dies ist ein Plan, mit dem die Biberburg in 12 Tagen fertig wird:



Einen solchen Plan mit der kürzesten Bauzeit kann man in zwei Schritten bestimmen:

1. Zuerst muss die Schlafhöhle vor dem Dach eingeplant werden. Da die Schlafhöhle 5 Unterwasser-Arbeiter benötigt, der Damm 3 und der Wohnraum 4, kann die Schlafhöhle – bei der Beschränkung auf 7 Unterwasser-Arbeiter – auch nicht gleichzeitig mit Damm oder Wohnraum gebaut werden. Die Schlafhöhle muss also zuerst gebaut werden und alle drei anderen Teile danach.
2. Damm und Wohnraum können gleichzeitig nach der Schlafhöhle gebaut werden, oder eines der beiden Teile gleichzeitig mit dem Dach. Es ist aber nicht möglich, alle drei Teile gleichzeitig zu bauen, weil sie zusammen $3 + 4 + 2 = 9$ Unterwasser-Arbeiter benötigen – mehr als zur Verfügung stehen. Die kürzeste Bauzeit kann erzielt werden, wenn die beiden Teile mit den kürzeren Bauzeiten (Dach und Wohnraum) hintereinander und der Damm gleichzeitig zu diesen gebaut werden.

Dies ist Informatik!

Einen optimalen, möglichst zügigen Ablauf eines Projekts zu planen, ist eine schwierige Aufgabe, bei der einige Bedingungen zu berücksichtigen sind. Zwischen Teilaufgaben eines Projekts bestehen oft zeitliche Abhängigkeiten; z.B. kann es Teilaufgaben geben die erst nach Beendigung einer anderen

Teilaufgabe begonnen werden können – wie hier bei Dach und Schlafhöhle. Ausserdem braucht jede Teilaufgabe bestimmte Ressourcen wie Arbeitskraft, Zeit und Geräte. Bei der Erstellung von Projektplänen hilft es, wenn man den Plan gut darstellen kann. Die in dieser Biberaufgabe gezeigten Diagramme sind eine Art von Gantt-Diagrammen, die von Henry Gantt (1861−1919) zwischen 1910 und 1915 entwickelt wurden; ähnliche Darstellungen wurden unabhängig von Gantt zur gleichen Zeit auch in Deutschland verwendet. Sie zeigen die Nutzung von Ressourcen (in diesem Fall die beiden Arten von Arbeitskräften) im Zeitverlauf.

Den optimalen Plan für die Biberburg kann man sich im Kopf überlegen und dabei alle erlaubten Möglichkeiten ausprobieren. Bei grösseren Projekten würde das zu lange dauern und zu unübersichtlich werden. Hier können Computerprogramme helfen, und deshalb ist die Erstellung von Zeitplänen (engl. *Scheduling*) ein wichtiges Thema der Informatik. Wie häufig bei schwierigen Problemen wurden Verfahren entwickelt, die statt eines garantiert optimalen Plans einen Plan mit etwas grösserem, aber immer noch sehr gutem Zeitbedarf erstellen. Scheduling wird auch bei der Steuerung von Computern selbst angewandt, deren Prozesse um Ressourcen (Rechenleistung, Speicherzugriff, Zugriff auf externe Geräte wie Speichergeräte, Drucker oder Netzwerkschnittstellen) konkurrieren.

Stichwörter und Webseiten

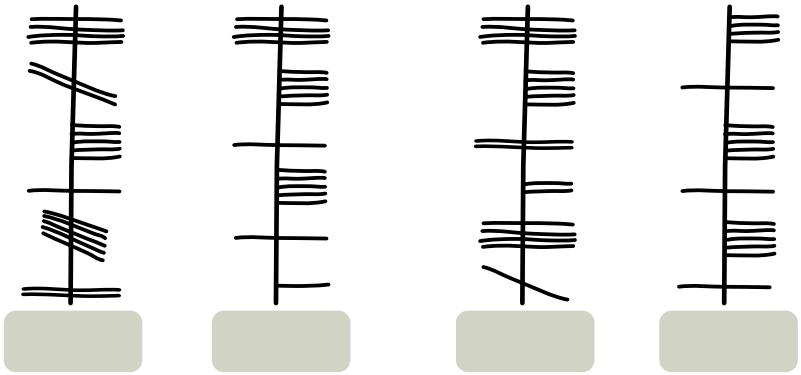
- Schedule: `https://de.wikipedia.org/wiki/Scheduling`
- Gantt-Diagramm: `https://de.wikipedia.org/wiki/Gantt-Diagramm`
- Software für Projektmanagement: `https://de.wikipedia.org/wiki/Projektmanagement-Software`

3. 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.

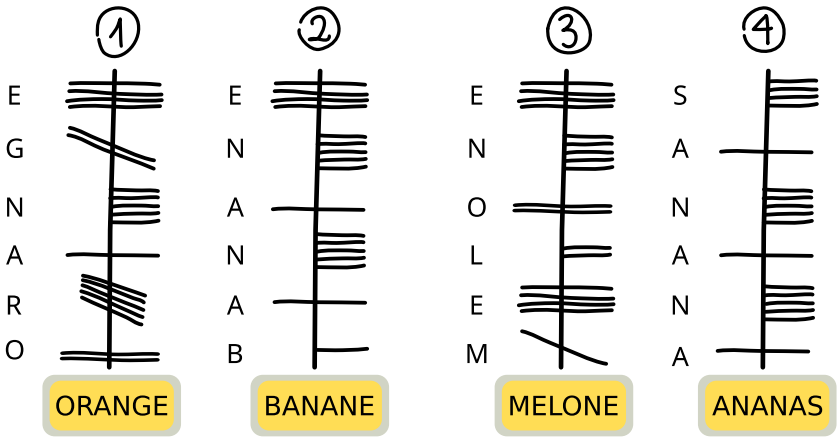
Welches Wort passt zu welchem Ogham-Code?



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.

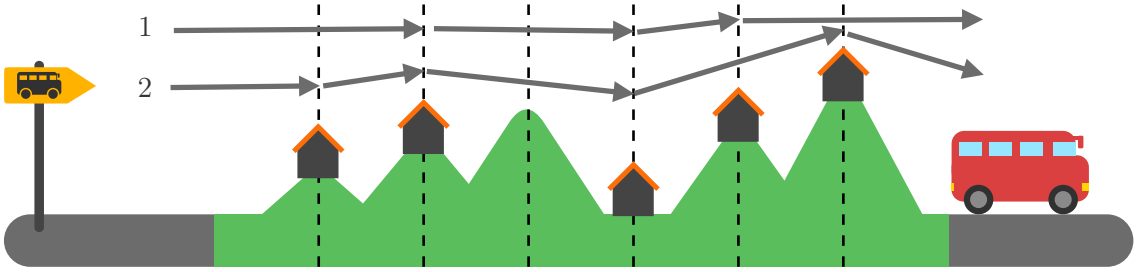
Auch in der Informatik werden Schriftzeichen und Texte entschlüsselt – nachdem sie vorher zur abhörsicheren Datenübertragung verschlüsselt wurden. Dazu werden aber ganz andere Verfahren verwendet als bei der Kodierung von Wörtern in anderen Schriften. Solche einfachen Kodierungen sind insbesondere mit 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`

4. 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

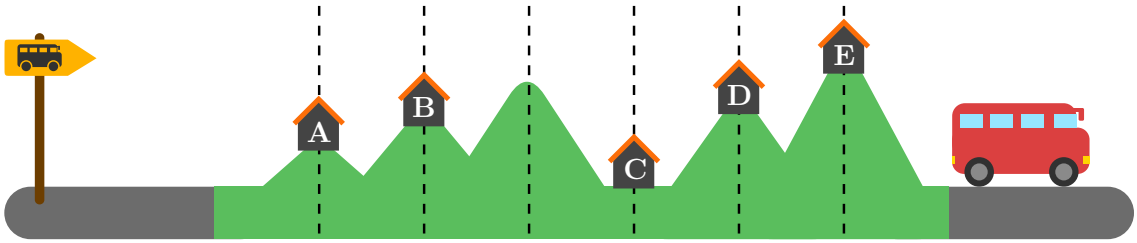
E) 6 Wanderungen

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

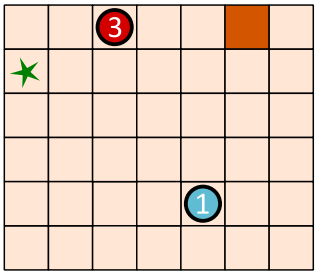
Teilprobleme zusammengesetzt. Dies wird wiederholt, bis die optimale Lösung des vollständigen Problems gefunden ist. Da gefundene Teil-Lösungen häufig zu Lösungen vieler grösserer Teile beitragen, werden sie gespeichert, um wiederholte gleiche Berechnungen einzusparen. Auch beim Zählen von Möglichkeiten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

5. Go-Bots

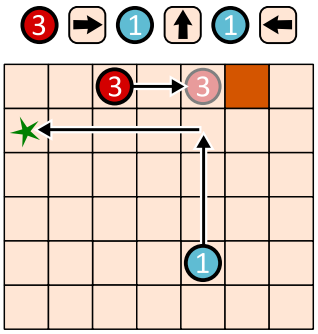
Die Go-Bots sind sehr einfache Roboter. Sie fahren über ein Spielbrett mit Feldern.



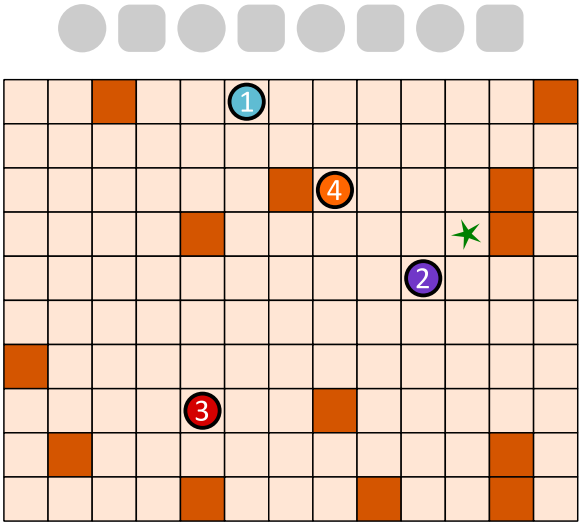
Um sie zu steuern, wählt man zunächst einen der Go-Bots aus. Den schickt man dann mit einem Pfeil-Befehl in eine Richtung: hoch, runter, links und rechts. Der Go-Bot fährt dann stur geradeaus, bis er direkt vor einem Hindernis oder einem anderen Roboter ankommt. Dort bleibt er stehen, bis er einen neuen Befehl bekommt.

Mit einer geschickten Folge von Befehlen sollst du dafür sorgen, dass Go-Bot 1 das Ziel erreicht, also genau dort stehen bleibt.

Mit dieser Befehlsfolge erreicht Go-Bot 1 das Ziel:



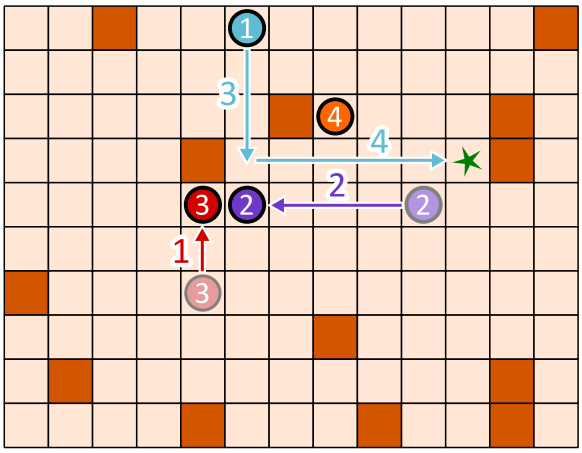
Erstelle eine Befehlsfolge mit vier Pfeilen, mit der Go-Bot 1 das Ziel erreicht!



Lösung

So ist es richtig: 3 ↑ 2 ← 1 ↓ 1 →

Damit Go-Bot 1 durch eine Befehlsfolge mit vier Pfeilen das Ziel erreichen kann, müssen drei Go-Bots kooperieren. Zuerst geht 3 nach oben, bis er vor einem Hindernis stehen bleibt. Damit wird er selbst zum Hindernis für 2 auf seinem Weg nach links. Wenn man nun 1 nach unten schickt, geht er bis 2 und kann von dort aus nach rechts gehen, wo er vor dem Hindernis stehen bleibt - auf dem Ziel.



Wie findet man die richtige Befehlsfolge? Man kann hinten anfangen und sich überlegen, was die letzte Bewegung von Go-Bot 1 zum Ziel sein muss. Es gibt nur zwei Möglichkeiten:
a) Er kommt von links, wie in unserer Lösung.
b) Er kommt von oben. In diesem Fall müsste Go-Bot 4 mit drei Befehlen nach oben rechts bewegt werden, um für 1 als Hindernis zu dienen. Wir benötigten dann $3 + 2 = 5$ Pfeil-Befehle. Gesucht ist aber eine Folge mit vier Befehlen. Also muss Möglichkeit a) korrekt sein, von Go-Bot 1 kommt von links zum Ziel. Dann geht die vorletzte Bewegung des Go-Bots 1 von oben nach unten. Damit er an der richtigen Stelle stehen bleibt, müssen zuvor 2 und 3 wie im Bild bewegt werden.

Dies ist Informatik!

In dieser Biberaufgabe haben mehrere Roboter zusammen gearbeitet, um gemeinsam ein Ziel zu erreichen. Dabei hatten sie unterschiedliche Aufgaben. Der blaue Roboter musste zum Ziel kommen, und die anderen dienten als Hindernisse.

Aufgabenverteilung ist ein wichtiger Aspekt der Robotik. Zum Beispiel arbeiten in einem automatisierten Warenlager unterschiedliche Roboter zusammen, um Waren einzulagern, wiederzufinden und zu transportieren. Dabei werden alle Aktivitäten so koordiniert, dass möglichst wenig nutzlose Ruhezeiten entstehen, alle Transportwege möglichst kurz sind, wenig Energie verbraucht wird und so insgesamt das Warenlager möglichst effizient arbeitet.

Ein besonderes Gebiet der Robotik sind Schwarmroboter. Das sind – wie die Go-Bots – einfache Maschinen, die in einer grossen Gruppe gemeinsam eine Aufgabe lösen. In der Landwirtschaft können

inzwischen Schwärme von Robotern die Aussaat von Mais erledigen, die Entwicklung der Pflanzen und die Bodenbeschaffenheit beobachten und schliesslich sogar das Getreide ernten. Jeder Schwarmroboter ist klein und einfach konstruiert, aber der Schwarm als Ganzes kann Grosses leisten. Dieses Prinzip gilt auch für Multiagentensysteme: Das sind einfache Softwareeinheiten, die gemeinsam komplexe Probleme lösen können. Die Aufgabe der Informatik ist, Algorithmen für eine optimale Koordination und Kooperation von Gesamtsystemen mit mehreren Akteuren – ob Hardware oder Software – zu entwickeln.

Stichwörter und Webseiten

- Multi-Robotersysteme: `https://www.hsu-hh.de/rt/forschung/multirobotersysteme`
- Robotik: `https://www.infineon.com/cms/de/discoveries/grundlagen-robotics/`
- Schwarmroboter
- Industrieroboter: `https://de.wikipedia.org/wiki/industrieroboter`
- Multiagentensystem: `https://de.wikipedia.org/wiki/Multiagentensystem`

6. Emma erledigt

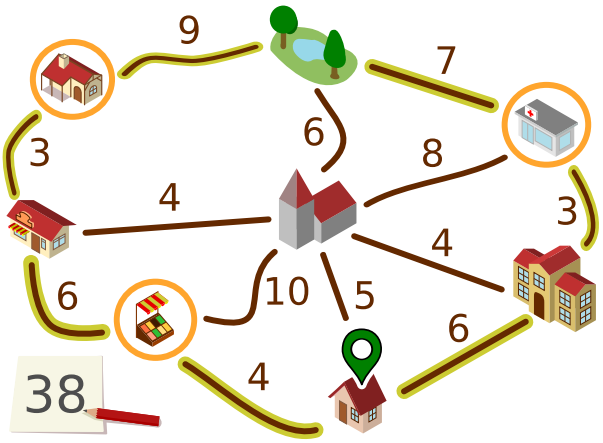
Emma ist zu Hause . Sie soll drei Aufgaben erledigen und zurückkommen:

- 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.

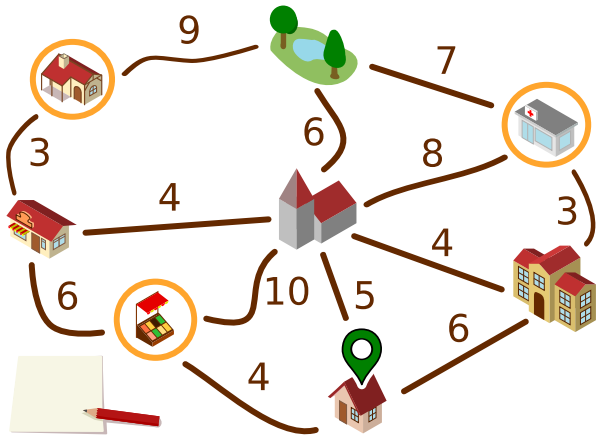
Auf einem Plan hat Emma eingetragen, wie viele Minuten sie für die Strecken zwischen einzelnen Orten ihrer Stadt benötigt. Ausserdem hat sie im Plan markiert, welche Strecke sie auf ihrem Weg geht.

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



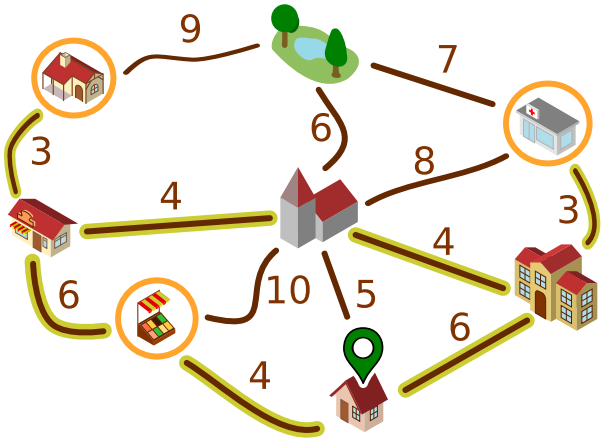
Emma überlegt, ob es noch schneller geht. Vielleicht hilft es, manche Strecken hin und zurück zu gehen?

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

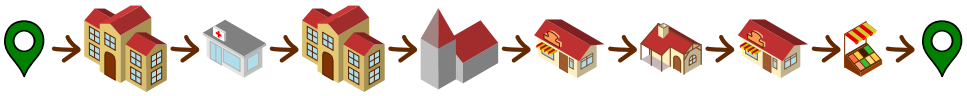


Lösung

So ist es richtig:



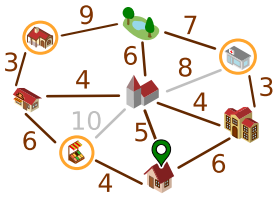
Emma kann so entlang der ausgewählten Strecken gehen (oder in die Gegenrichtung):



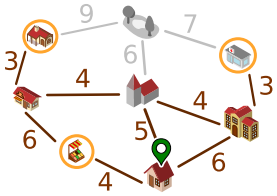
Für diesen Weg braucht sie $6 + 3 + 3 + 4 + 4 + 3 + 3 + 6 + 4 = 36$ Minuten.

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

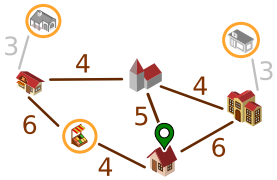
Die grau gezeichneten Strecken können wir ignorieren. Es gibt kürzere Wege zwischen den durch die Strecken verbundenen Orte, nämlich über andere Orte.



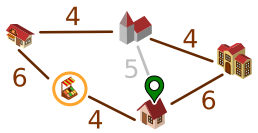
Auch den Park können wir ignorieren. Emma muss nicht zum Park. Zudem gibt es für jeden Weg, der über den Park geht, eine kürzere Alternative.



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.

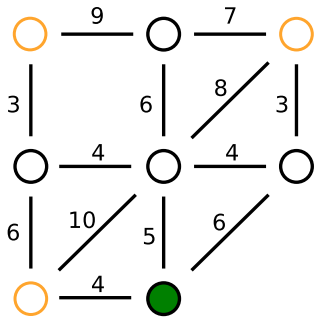


Nun bleibt nur noch der Plan rechts übrig. Start und Ende des Weges ist hier . Diese drei Orte ( ) müssen besucht werden. Der kürzeste Weg, der das erfüllt, geht über alle fünf Orte und entlang aller Strecken ausser der grauen und dauert $4 + 6 + 4 + 4 + 6 = 24$ Minuten. Mit den 12 Minuten von oben macht 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

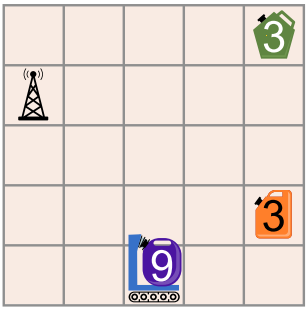
- Weg (in Graphen): `https://de.wikipedia.org/wiki/Weg_(Graphentheorie)`
- Kürzeste Wege: Abenteuer Informatik, Kapitel 1 (`http://abenteuer-informatik.de/dasbuch.html`)

7. Zerobots Mission

Zerobot hat einen austauschbaren Treibstofftank. Zerobot bewegt sich damit in einem Raster: nach oben, unten, rechts und links. Bei jeder Bewegung von einem Rasterfeld zum nächsten sinkt der Füllstand des Tanks um 1.

Auf einigen Feldern sind Austauschtanks; die Zahl darauf zeigt den Füllstand an. Wenn Zerobot ein solches Feld erreicht, tauscht er seinen Tank, egal wie voll der ist: Er nimmt den Austauschtank auf, setzt seinen bisherigen Tank auf dem Feld ab und fährt weiter.

Zerobots aktuelle Position und der Füllstand seines Tanks werden im Bild so angezeigt: 



Alarm: Die Tanks sind fehlerhaft und könnten explodieren!

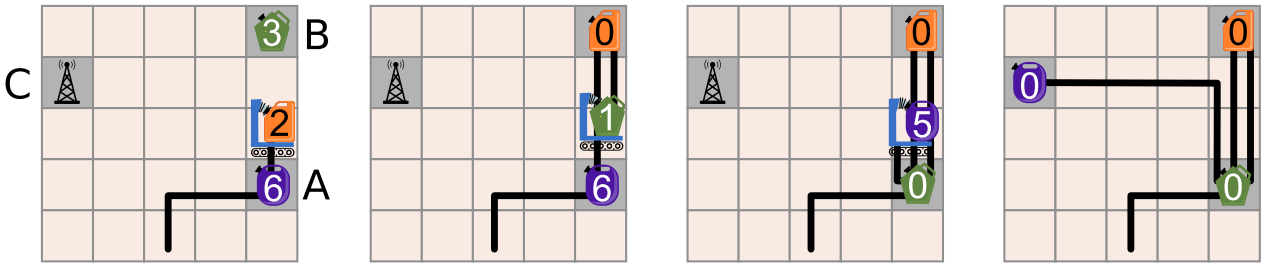
Das ist Zerobots Mission: Er soll so zur Basisstation  fahren, dass am Ende alle Tanks leer sind (Füllstand 0).

Wie muss sich Zerobot bewegen, um seine Mission zu erfüllen?

Lösung

So ist es richtig:

Zerobot kann mit 15 Bewegungen so zur Basisstation fahren, dass am Ende alle Tanks Füllstand 0 haben:



Um die richtige Antwort leichter erklären zu können, bezeichnen wir die Felder mit den Austauschtanks und der Basisstation mit den Buchstaben A, B und C:

Zerobot fährt 3 Felder bis A und tauscht  (Füllstand 6) gegen  (Füllstand 3) aus. Dann fährt er 3 Felder bis B und tauscht  (Füllstand 0) gegen  (Füllstand 3) aus. Damit fährt er wieder zu A und tauscht  (Füllstand 0) gegen  (Füllstand 6) aus. Damit fährt er 6 Felder bis zur Basistation C.  hat dann den Füllstand 0. Mission erfüllt!

Ist dies die einzige richtige Lösung? Zerobot muss exakt 15 Bewegungen machen: 15 Bewegungen sind mindestens nötig, um den gesamten verfügbaren Treibstoff von $9 + 3 + 3 = 15$ Einheiten zu verbrauchen, und für mehr Bewegungen reicht der Treibstoff nicht. Um alle Tanks zu leeren, muss er beide Felder mit Austauschtanks besuchen, und A sogar zweimal. Wenn der Zerobot zuerst das Feld B besuchen würde, bräuchte er 17 Bewegungen, um die Basisstation zu erreichen, was nicht möglich ist. Somit ist die gezeigte Reihenfolge der Tanks die einzige richtige Antwort.

Dies ist Informatik!

In dieser Biberaufgabe werden einige grundsätzliche Probleme der autonomen Mobilität angesprochen: Jeder autonome mobile Robotor (wie z.B. ein selbstfahrendes Auto) muss beachten, wie viel Energie in Form von Treibstoff oder Batterieladung zur Verfügung steht, wenn er seine Aktivitäten plant. Auf der einen Seite muss er sicherstellen, dass er rechtzeitig eine Ladestation oder Tankstelle erreicht, bevor sein Energievorrat verbraucht ist. Auf der anderen Seite gibt es Rahmenbedingungen zu beachten. In der Aufgabe ist eine Rahmenbedingung, dass am Ende der Energievorrat komplett verbraucht sein musste. In der Wirklichkeit hat man es vor allem mit anderen Rahmenbedingungen zu tun, wie z.B. die Position und Verfügbarkeit von Ladestationen. Die Software zur Steuerung mobiler Roboter enthält Komponenten, die für die Sicherstellung ausreichender Energie durch Nachladen sorgen (*intelligentes Batterieladungsmanagement*).

Darüber hinaus werden Computerprogramme auch zur Planung und Verwaltung effizienter Netze von Ladestationen verwendet. Informatikerinnen und Informatiker forschen an Lösungen zum charging station placement problem: Ladestationen für mobile Roboter müssen so platziert werden, dass ein Roboter mit einem gewissen Mindestladestand eine der verfügbaren Ladestationen erreichen kann. Für die Kommunikation zwischen Ladestationen und selbstfahrenden Autos wurden Protokolle entwickelt wie z.B. das OCPP (Open Charge Point Protocol).

Stichwörter und Webseiten

- Intelligentes Batterieladungsmanagement: `https://www.researchgate.net/publication/364734487_Intelligent_Battery_-Recharge_Management_for_Mobile_Robots`
- Open Charge Point Protocol: `https://de.wikipedia.org/wiki/OCPP`

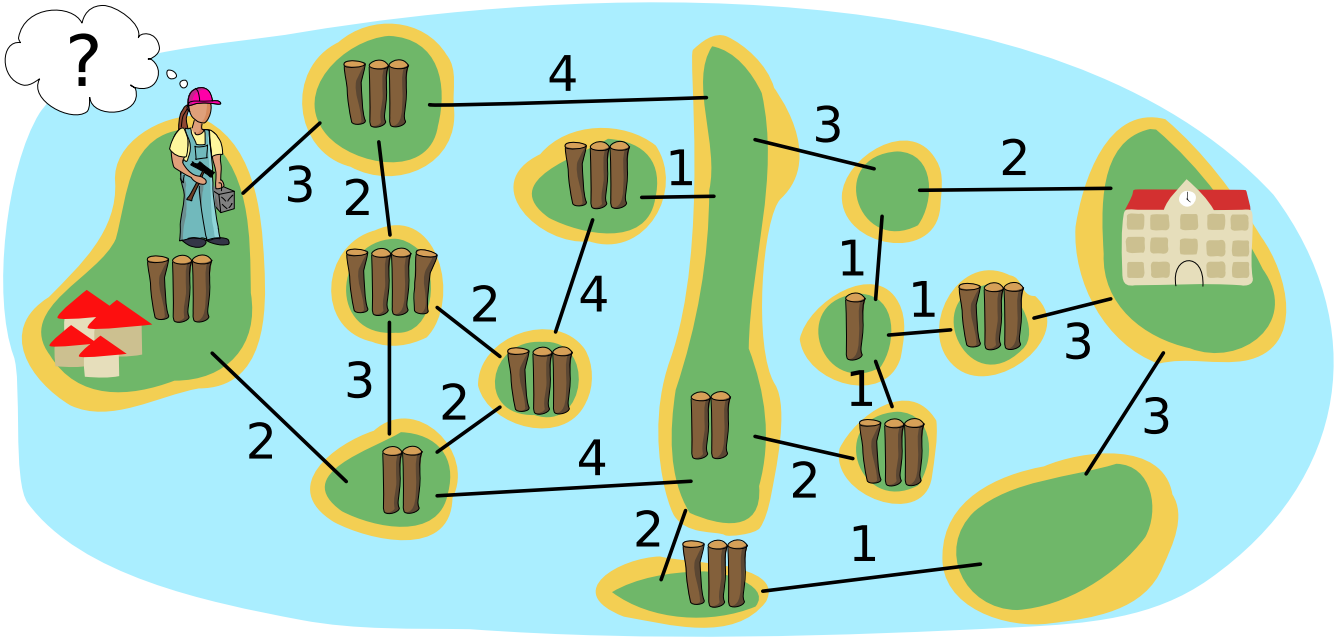
8. Brücken bauen!

Auf der Insel ganz links sind Kinder eingezogen. Bianca soll Brücken bauen, über die die Kinder zur Schule auf der Insel ganz rechts gehen können.

Die Insel-Karte zeigt, wie viele Baumstämme es auf jeder Insel gibt. Diese Baumstämme kann Bianca nehmen, um an den Linien Brücken zu bauen. Die Zahl an einer Linie sagt, wie viele Baumstämme dort für eine Brücke benutzt werden. Sobald es zwischen zwei Inseln eine Brücke gibt, kann Bianca darüber gehen und Stämme, die sie noch hat, mitnehmen. Natürlich kann sie jeden Baumstamm nur für eine Brücke benutzen.

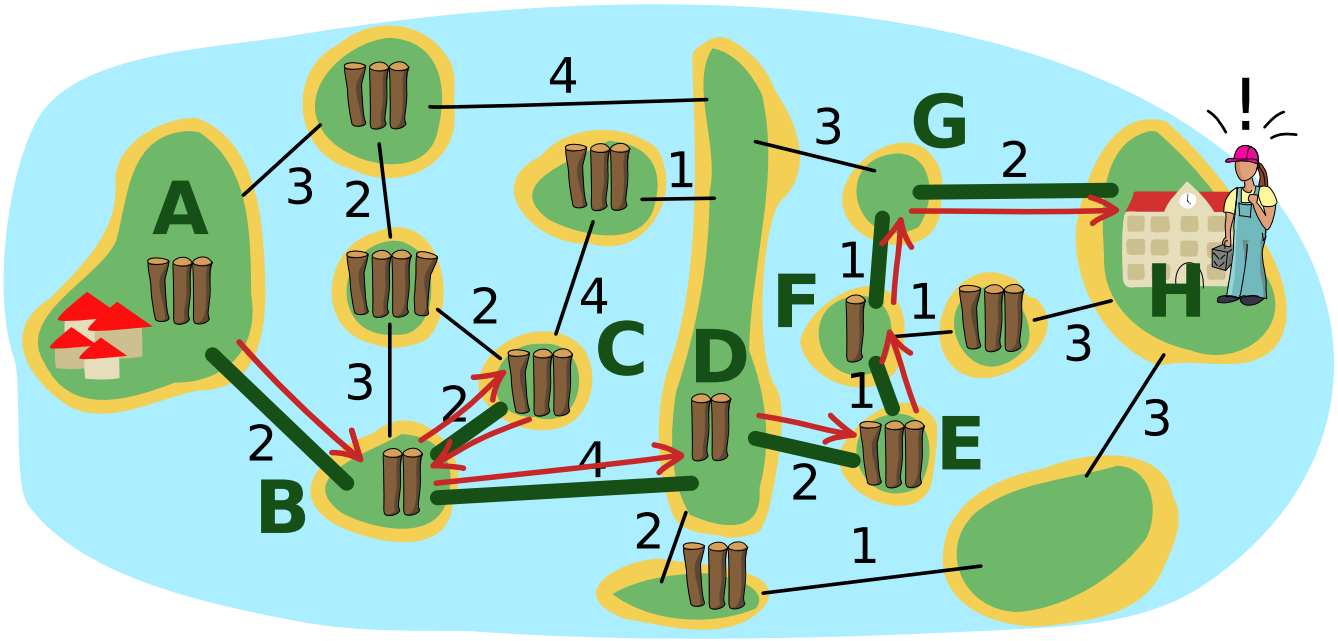
Bianca fängt auf er Insel links an. Ihr Ziel ist, möglichst wenige Baumstämme zu benutzen.

An welchen Linien soll Bianca Brücken bauen, damit sie ihr Ziel erreicht?



Lösung

So ist es richtig:



Die grünen Linien zeigen, wo Bianca Brücken gebaut hat. Die roten Pfeile zeigen, wie Bianca über die Brücken gegangen ist:

- Auf der Insel A nimmt sie die drei Baumstämme und benutzt zwei davon für die erste Brücke. Mit dem verbleibenden Baumstamm geht sie über die Brücke und hat auf der Insel B $3 - 2 + 2 = 3$ Baumstämme. Das sind nicht genug, um eine Brücke zur Insel D zu bauen.
- Deshalb baut sie mit 2 Stämmen eine Brücke zur Insel C. Sie geht über die Brücke, nimmt die 3 Stämme von der Insel C und geht zurück. Nun hat sie $3 - 2 + 3 = 4$ Stämme.
- Damit baut sie eine Brücke zur Insel D, geht über die Brücke und hat dann die 2 Stämme von Insel D.
- Damit baut sie eine Brücke zur Insel E und kann dort 3 Stämme nehmen. Sie baut weitere Brücken zu den Inseln F und G. Auf der Insel E hat sie also 3 Stämme, auf der Insel F $3 - 1 + 1 = 3$ Stämme und auf der Insel G noch 2 Stämme.
- Die reichen genau, um eine Brücke zur Insel H mit der Schule zu bauen.

Insgesamt konnte Bianca also Brücken für einen Weg von Insel A zu Insel H bauen und hat dafür 14 Baumstämme benutzt. Aber geht es auch mit weniger Stämmen? Dazu müssen alle möglichen Wege untersucht werden. Weil die alle über die lange Insel D führen, lässt sich das Problem in zwei Teile zerlegen: Von Insel A zu Insel D, und von Insel D zu Insel H:

- Für die Brücken von Insel A bis Insel D hat Bianca 8 Stämme benutzt und kam ohne Stamm auf Insel D an. Wir notieren ihren Weg so: $2 - [2, 2] - 4$ (von der Insel A über die Linie mit der 2 zur Insel B, dann zwischen B und C hin und zurück über die 2, dann über die 4 zu Insel D). Ein Weg mit weniger Stämmen wäre $3 - 4$, kann aber nur mit Umweg gebaut werden

$(3 - [2, 2] - 4)$, verbraucht also 9 Stämme, wobei Bianca auf Insel D mit einem Stamm im Vorrat ankommt. Alle anderen Wege von Insel A bis D verbrauchen 9 Stämme oder mehr.

- Für die Brücken von Insel D bis H hat Bianca 6 Stämme benutzt. Den direkten Weg $3 - 2$ kann sie nicht bauen, auch nicht mit einem Stamm im Vorrat. Alle anderen Wege von Insel D zu Insel H verbrauchen 6 Stämme oder mehr.

Es ist also nicht möglich, mit weniger als 14 Stämmen Brücken zu bauen, über die die Kinder von der Dorf-Insel A zur Schul-Insel H gehen können. Mit den von ihr gebauten Brücken hat Bianca also ihr Ziel erreicht.

Dies ist Informatik!

Die Insel-Karte mit den durch Linien angezeigten «Brücken-Bauplätzen» kann als *Graph* modelliert werden: Das ist eine mathematische Struktur, die Objekte (auch Knoten genannt) paarweise miteinander in Relation setzt (die Paare nennt man auch Kanten). In einem Graphen man die Inseln als Knoten und die Linien als Kanten modellieren. Dabei haben die Kanten *Gewichte*, nämlich die Anzahl der für den Brückenbau entlang einer Linie benutzten Baumstämme, aber auch die Knoten (die Anzahl der Stämme auf einer Insel) – das ist eher ungewöhnlich. Für Graphen, bei denen nur die Kanten gewichtet sind, kennt die Informatik mehrere effiziente Algorithmen, die einen kürzesten Weg (über Kanten mit minimaler Summe der Gewichte) zwischen zwei Knoten berechnen können.

Das Problem, das Bianca in dieser Biberaufgabe optimal lösen möchte, ist komplizierter: Sie möchte zwar auch einen kürzesten Weg gehen, hat aber eine Randbedingung: Die Summe der Knotengewichte auf ihrem bisherigen Weg (die Stämme, die sie nehmen konnte) abzüglich der Summe der Kantengewichte auf ihrem Weg (die Stämme, die sie für den Brückenbau benutzt hat) muss grösser sein als das Gewicht der Kante, die sie als nächste gehen bzw. wo sie eine Brücke bauen möchte. Um den optimalen Weg zu finden, müssen hier eventuell alle Möglichkeiten ausprobiert werden. Die Zerlegung des Problems in zwei Teile hilft, die Anzahl der Möglichkeiten zu reduzieren. Und wegen der Randbedingung kann man viele Möglichkeiten ausschliessen, bevor man sie komplett probiert hat. In der Informatik ist ein solches Vorgehen (Probieren und Ausschliessen) als *Backtracking* bekannt (siehe auch die Biberaufgabe «Gemüsebeet»).

Stichwörter und Webseiten

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

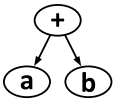
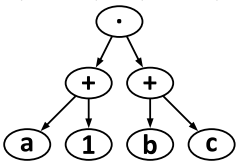
9. Postfix-Notation

Ein mathematischer Ausdruck besteht aus ...

- ... einem *Operator*: +, −, · oder **:**
- ... und den *Operanden*: Zahlen wie 1, 2, ..., Buchstaben wie a, b, ... oder wieder Ausdrücke wie (1 + 2).

Die Struktur eines mathematischen Ausdrucks kann man als *Strukturbaum* darstellen. Dieses Diagramm aus Operatoren und Operanden wird so gezeichnet: Ein Kringel mit dem Operator wird durch Pfeile mit den Strukturbäumen der Operanden verbunden. Das sind im einfachsten Fall Kringel mit einer Zahl oder einem Buchstaben.

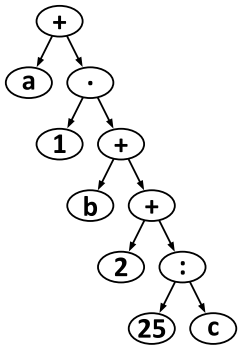
Aus einem Strukturbaum wiederum kann man die *Postfix-Notation* eines mathematischen Ausdrucks ablesen. In dieser Notation werden für jeden Ausdruck zunächst die Operanden und dahinter der Operator geschrieben.

Mathematischer Ausdruck:	a + b	(a + 1) · (b + c)
Strukturbaum:		
Postfix-Notation:	a b +	a 1 + b c + ·

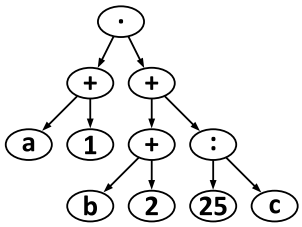
Hier ist die Postfix-Notation eines anderen Ausdrucks:

a 1 + b 2 + · 25 c : +

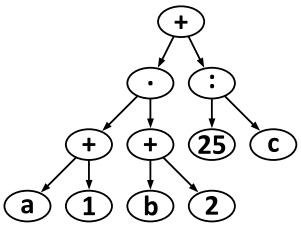
Welchen Strukturbaum hat dieser Ausdruck?



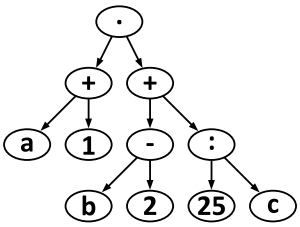
A)



B)

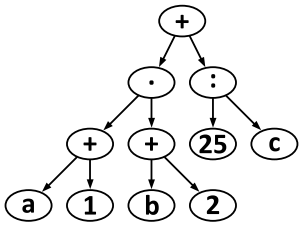


C)



D)

Lösung



Antwort C ist richtig:

Wie in der Aufgabenstellung beschrieben ist, findet sich der zentrale Operator eines mathematischen Ausdrucks im Strukturbaum ganz oben (er ist dessen *Wurzel*) und in der Postfix-Notation ganz hinten. Möchte man den Strukturbaum eines Ausdrucks in Postfix-Notation finden oder erstellen, muss man das in der Postfix-Notation letzte Zeichen oben im Baum suchen, in diesem Fall das +. Nur bei den Bäumen der Antworten A und C findet sich ein + in der Wurzel.

Der Operator + hat zwei Operanden, einer links und einer rechts. In der Postfix-Notation sieht man direkt (am vorletzten Zeichen), dass der rechte Operand des Ausdrucks wiederum ein Ausdruck ist, der den Operator : hat. Im Strukturbaum muss also rechts unter der Wurzel ein : zu sehen sein. Das ist nur im Baum aus Antwort C der Fall. Also muss das die richtige Antwort sein.

Das kann man auch zeigen, indem man den Strukturbaum aus Antwort C vollständig in Postfix-Notation umwandelt:

- Die untersten drei «kleinsten» Teilbäume, die jeweils aus 3 Elementen bestehen, werden zu a 1 +, b 2 + und 25 c :.
- Die beiden linken dieser drei «kleinsten» Teilbäume werden zum linken Operanden des oberen +, so dass die Umwandlung des linken Teilbaum nun a 1 + b 2 + · lautet. Der dritte der «kleinsten» Teilbäume ist bereits der rechte Operand.
- Damit lautet die Postfix-Notation des Strukturbaums aus Antwort C insgesamt so: a 1 + b 2 + · 25 c : +. Das ist genau der in der Aufgabenstellung vorgegebenen Ausdruck.

Dies ist Informatik!

Die *Postfix-Notation*, auch *umgekehrte polnische Notation* (engl. *reverse Polish notation RPN*) genannt, wird oftmals verwendet, um mathematische oder andere Ausdrücke (z. B. in Programmiersprachen) missverständnisfrei und kompakt zu formulieren. Würde man den durch den Strukturbaum aus Antwort C gegebenen Ausdruck in normaler Notation schreiben (also mit Operatoren zwischen den Operanden, deshalb auch *Infix-Notation* genannt), müsste man Klammern setzen (a + 1) · (b + 2) + 25 : c, die man für die Postfix-Notation nicht braucht. Die Postfix-Notation wurde von Jan Łukasiewicz (1878−1956) zunächst als Präfix-Notation eingeführt, mit dem Operator vor den Operanden. So schreibt man u.a. die Anwendung von Funktionen auf: in der Mathematik $f(x, y)$,

beim Programmieren `funktionsname(argument1, argument2, argument3)`. Im Computer wird sie unter anderem beim *Parsen* von Ausdrücken einer Programmiersprache verwendet.

In der jüngeren Vergangenheit haben viele Menschen die Postfix-Notation vor allem bei der Nutzung der ersten wissenschaftlichen Taschenrechner kennengelernt: Dort konnte man damit schnell und zuverlässig und vor allem ohne Klammern komplexe mathematische Ausdrücke eingeben und berechnen lassen. Noch heute gibt es eine eingeschworene Gemeinschaft, die programmierbare Taschenrechner (wie z. B. den HP 35s) mit Postfix-Notation nutzen.

Stichwörter und Webseiten

- Umgekehrte polnische Notation UPN: `https://de.wikipedia.org/wiki/Umgekehrte_polnische_Notation`
- Syntaxbaum: `https://de.wikipedia.org/wiki/Syntaxbaum`
- Jan Łukasiewicz: `https://de.wikipedia.org/wiki/Jan_Łukasiewicz`
- HP 35s: `https://en.wikipedia.org/wiki/HP_35s`

10. Ziffernschloss

Bob hat ein Ziffernschloss an seiner Haustür. Um es zu öffnen, muss man einen Zifferncode eingeben. Alle Ziffern im Code müssen verschieden sein. Aktuell hat der Code fünf Stellen und lautet so:

Bob hat sich den Code notiert, aber ein wenig verschleiert: $n >> c$ bedeutet, dass links von Ziffer c im Code genau n Ziffern stehen, die grösser sind als c. Zum Beispiel notiert Bob mit

$1 >> 3$

dass links von Ziffer 3 genau eine Ziffer steht (nämlich die 4), die grösser ist als 3. Den aktuellen Zifferncode hat er sich insgesamt so notiert:

$0 >> 0$; $3 >> 1$; $0 >> 2$; $1 >> 3$; $0 >> 4$

Ein Code aus nur fünf Ziffern ist Bob zu unsicher. Deshalb überlegt er sich einen neuen Code, aus den Ziffern 0 bis 7. Den neuen Code notiert er sich so:

$3 >> 0$; $2 >> 1$; $4 >> 2$; $4 >> 3$; $1 >> 4$; $1 >> 5$; $1 >> 6$; $0 >> 7$

Wie lautet der neue Code?

Lösung

So ist es richtig:

Um den Code zu bestimmen, schauen wir uns Bobs Notation genauer an, nach und nach für die Ziffern 0 bis 7.

- $3 >> 0$: Links von 0 stehen genau 3 Ziffern, die grösser sind als 0. Die Ziffer 0 muss also an der vierten Stelle des Codes stehen.
- $2 >> 1$: Links von 1 stehen genau 2 Ziffern, die grösser sind als 1. Die Ziffer 1 muss also an der dritten Stelle des Codes stehen.
- $4 >> 2$: Links von 2 stehen genau 4 Ziffern, die grösser sind als 2. Da die kleineren Ziffern 1 und 0 bereits an dritter und vierter Stelle stehen, müssen die 4 grösseren Ziffern an erster, zweiter, fünfter und sechster Stelle stehen. Die Ziffer 2 muss also an der siebten Stelle des Codes stehen.
- $4 >> 3$: Links von 3 stehen genau 4 Ziffern, die grösser sind als 3. Die Ziffer 3 muss also an der achten und letzten Stelle des Codes stehen.
- $1 >> 4$: Links von 4 steht genau 1 Ziffer, die grösser ist als 4. Die Ziffer 4 muss also an der zweiten der verbleibenden Stellen stehen; das ist die zweite Stelle des Codes.
- $1 >> 5$: Links von 5 steht genau 1 Ziffer, die grösser ist als 5. Die Ziffer 5 muss also an der zweiten der nun noch verbleibenden Stellen stehen; das ist die fünfte Stelle des Codes.
- $1 >> 6$: Links von 6 steht genau 1 Ziffer, die grösser ist als 6. Die Ziffer 6 muss also an der zweiten der nun noch verbleibenden Stellen stehen; das ist die sechste Stelle des Codes.
- $0 >> 7$: Es gibt keine Ziffer, die grösser ist als 7. Die Ziffer 7 muss an der letzten freien Stelle, also an der ersten Stelle des Codes stehen.

Dies ist Informatik!

Bob beschreibt in seiner Notation, wie sich der Code zu einer sortierten Folge der verwendeten Ziffern bzw. Zahlen verhält.

Schauen wir uns den fünfstelligen Code noch einmal an: 0 2 4 3 1. Er entsteht, indem man die sortierten Zahlen 0 1 2 3 4 nimmt und deren Positionen verändert. Das Ergebnis nennt man auch *Permutation* (der Zahlen 0 bis 4). In einer Permutation können die Zahlen bzgl. ihrer Sortierung «verdreht» sein. Zum Beispiel steht im Code die 4 vor der 3, während in der sortierten Folge die 3 vor der 4 steht (denn $3 < 4$). Also steht die 3 «falsch» bzgl. der Sortierung. Das bezeichnet man in der Kombinatorik, einem Teilgebiet der Mathematik, als *Inversion* oder auch *Fehlstand*.

Bobs Code ist also eine Permutation, und seine Notation gibt für jede Zahl an, wie oft sie darin «invertiert» ist: Die 0 steht richtig, die 1 ist Teil von 3 Inversionen ($3 >> 1$: drei grössere Zahlen stehen vor der 1), die 2 steht richtig, die 3 ist einmal invertiert, die 4 steht richtig.
Die Folge dieser Inversionszahlen heisst *Inversionsssequenz*. Die 0 steht richtig, die 1 ist Teil von

3 Inversionen (3 >> 1: drei größere Zahlen stehen vor der 1), die 2 steht richtig, die 3 ist einmal invertiert, die 4 steht richtig. Die Folge dieser Inversionszahlen heißt *Inversionsssequenz*. (Die Summe der Inversionszahlen beschreibt übrigens den Grad der Unsortiertheit einer Permutation; vergleiche dazu auch die Biberaufgabe «Zug entladen».)

Wir haben nun drei Folgen – den Code (bzw. die Permutation), die sortierte Folge und die Inversionssequenz – und fassen sie in dieser Tabelle zusammen:

Code / Permutation	0	2	4	3	1
Sortierte Folge	0	1	2	3	4
Inversionssequenz	0	3	0	1	0

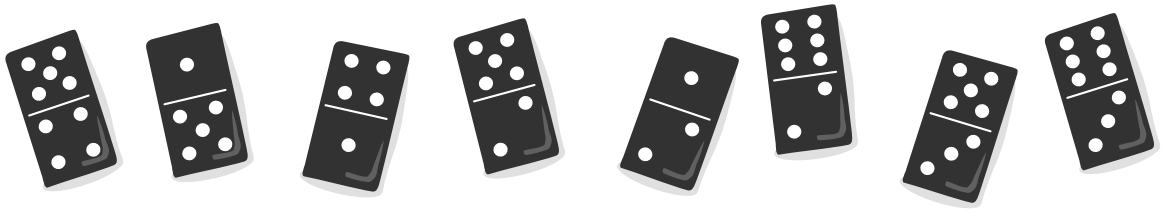
Die Beschreibung der Lösung hat gezeigt, dass es einen effizienten Algorithmus gibt, der aus der Inversionssequenz die zugehörige Permutation berechnet. Es genügt, die Inversionssequenz einmal durchzugehen. Die Informatik beschäftigt sich häufig mit kombinatorischen Problemen und kennt viele Algorithmen zur Lösung solcher Probleme. Sie können bei der automatischen Lösung von Rätseln verwendet werden (wie etwa Sudokus), aber auch bei vielen «ernsthaften» Problemen. Meist sind sie deutlich komplizierter als der Algorithmus zur Lösung dieser Biberaufgabe.

Stichwörter und Webseiten

- Permutation: `https://de.wikipedia.org/wiki/Permutation`
- Inversion bzw. Fehlstand: `https://de.wikipedia.org/wiki/Fehlstand`

11. Domino

Jeder Dominostein hat zwei Felder. Auf jedem Feld sind 1 bis 6 Punkte. Du hast diese acht Steine:



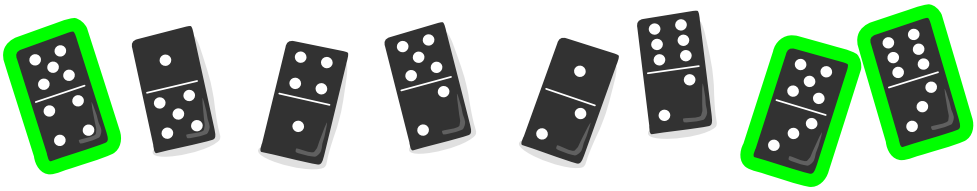
Alle acht Steine sollst du so in eine Reihe legen, dass auf den angrenzenden Feldern zweier benachbarter Steine immer gleich viele Punkte sind.

Du kannst mehrere solcher Reihen legen. Es gibt aber Steine, die du auf keinen Fall an den Anfang oder das Ende deiner Reihe legen kannst.

Welche Steine sind das?

Lösung

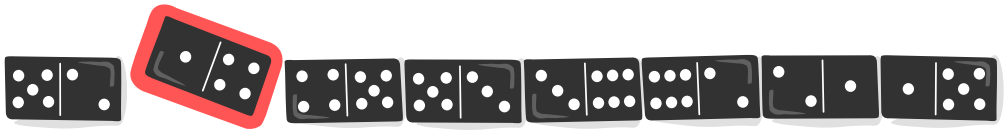
Drei der acht Steine können nicht an den Anfang oder das Ende der Reihe gelegt werden:



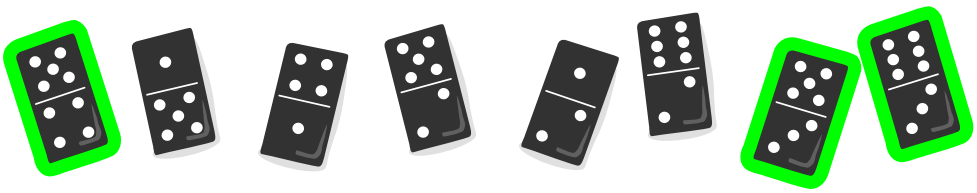
Um die Aufgabe zu lösen, untersuchen wir die Augenzahlen (die Punkte auf Dominosteinen nennt man auch *Augen*, wie bei einem Würfel) der 16 Felder der Dominosteine. Wir halten fest, wie häufig die einzelnen Augenzahlen vorkommen, und ob die Häufigkeit eine gerade oder eine ungerade Zahl ist:

Augenzahl	**Häufigkeit**	**Gerade/Ungerade**
	3	ungerade
	3	ungerade
	2	gerade
	2	gerade
	4	gerade
	2	gerade

Felder mit Augenzahlen, die mit gerader Häufigkeit vorkommen, müssen paarweise mitten in der Reihe liegen oder gleichzeitig an Anfang und Ende. Felder mit Augenzahlen, die mit ungerader Häufigkeit vorkommen, können aber nicht alle mitten in der Reihe liegen: Es kann nämlich nicht für jedes Feld mit dieser Augenzahl ein passendes angrenzendes Feld gefunden werden; das geht nur bei gerader Häufigkeit. Hier siehst du eine Reihe, in die ein Stein mit der dreimal vorkommenenden Augenzahl 1 nicht mehr mitten hinein passt.

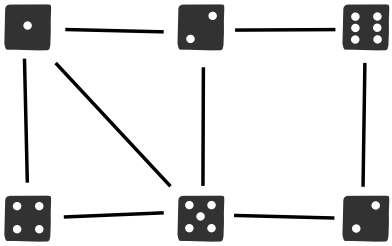


Da es auf den acht Steinen dieser Biberaufgabe Felder mit ungerader Häufigkeit gibt, müssen Steine mit solchen Feldern außen liegen. Steine, die zwei Felder mit gerader Häufigkeit haben, können also nicht an den Anfang oder das Ende der Reihe gelegt werden. Das sind folgende Steine:

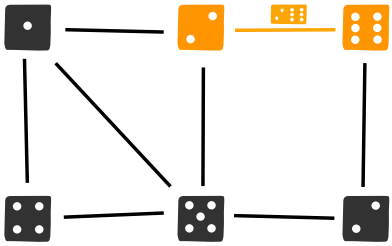


Dies ist Informatik!

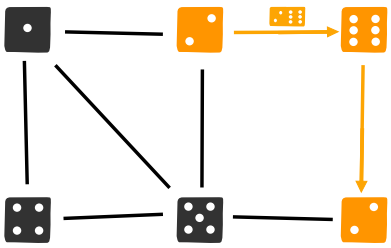
Es gibt mehrere Möglichkeiten, die acht Dominosteine dieser Biberaufgabe in eine korrekte Reihe zu legen. Um sich hier eine bessere Übersicht zu verschaffen, verwenden Informatiker sogenannte *Graphen*:



Im Graphen oben sind Quadrate (sogenannte *Knoten*) ersichtlich, die die sechs Augenzahlen der Dominosteine aufzeigen. Die acht Linien (sogenannte *Kanten*), stellen die acht Dominosteine dar; jede Linie verbindet zwei Felder. Der Dominostein 2–6 wird beispielsweise durch die folgende Kante dargestellt:



Um die Aufgabe zu lösen, müssen alle acht Dominosteine passend aneinander gereiht werden. Dabei ist nach dem Legen des ersten Dominosteins bereits klar, mit welcher Augenzahl der zweite Stein beginnen muss, denn angrenzende Felder zweier Steine sollen ja immer die gleiche Augenzahl haben. Im Graph erkennt man dies dadurch, dass Dominosteine genau dann nebeneinander gelegt werden dürfen, wenn deren Kanten sich bei demselben Knoten treffen. Die Steine 2–6 und 6–3 können beispielsweise aneinander gelegt werden, da sie beide die Augenzahl 6 enthalten:



Das Aneinanderreihen der Dominosteine lässt sich als *Weg* (eine Abfolge von Kanten) durch den Graphen verstehen. Dieser Weg soll sämtliche Kanten *genau einmal* besuchen, um sicherzustellen, dass die acht Dominosteine einerseits alle verwendet werden, andererseits aber auch nicht mehrmals zum Einsatz kommen. Ein Weg, der *jede Kante genau einmal* besucht, wird *Eulerweg* genannt. Der Name geht auf Leonhard Euler, einen Schweizer Mathematiker und den Erfinder der Graphentheorie, zurück. Euler konnte zeigen, dass in einem zusammenhängenden Graphen ein Eulerweg genau dann

existiert, wenn maximal zwei Knoten eine ungerade Anzahl von diesem Knoten ausgehender Kanten haben.

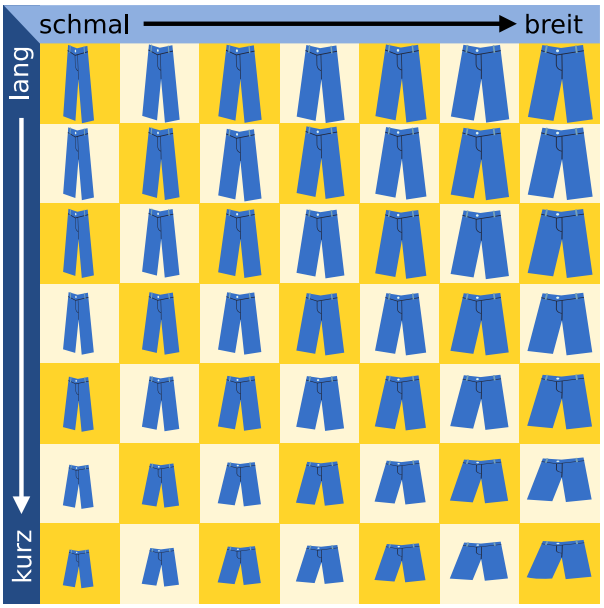
Stichwörter und Webseiten

- Graph, Knoten, Kanten: `https://de.wikipedia.org/wiki/Graph_(Graphentheorie)`
- Eulerweg: `https://de.wikipedia.org/wiki/Eulerkreisproblem`

12. Anprobieren

Christian braucht neue Hosen. Im Geschäft gibt es seine Lieblings-Hose in sieben Längen und sieben Breiten. Hosen in allen 49 Grössen sind im Regal, nach Länge und Breite sortiert.

Weil Christian seine richtige Grösse nicht weiss, muss er sie durch Anprobieren herausfinden. Bei jeder Anprobe merkt Christian, ob die Hose passt oder ob er eine kürzere, längere, schmalere oder breitere Hose braucht. Damit eine Hose passt, müssen Länge und Breite stimmen.



Der Verkäufer stöhnt: Bei 49 Grössen die richtige zu finden – das kann dauern.

Doch Christian ist eine Methode eingefallen, die richtige Grösse in jedem Fall nach möglichst wenigen Anproben zu wissen.

Wie viele Anproben braucht er mit dieser Methode höchstens, bis er die richtige Grösse weiss?

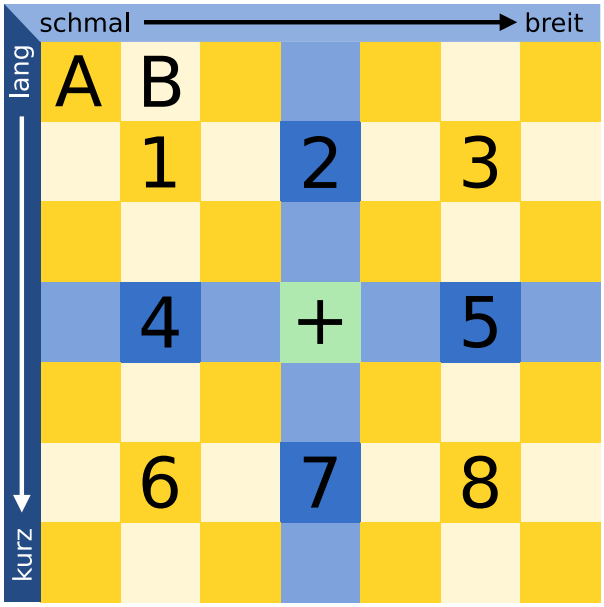
Lösung

2 ist die richtige Antwort.

Natürlich kann Christian Glück haben und direkt bei der ersten Anprobe die Hose in der richtigen Grösse erwischen. Aber auf Glück kann er sich nicht verlassen und geht nach dieser Methode vor:

Zuerst probiert er die Hose in der Mitte an (Position + im Bild). Bei der Anprobe prüft er Länge und Breite der Hose.

- Wenn Länge und Breite stimmen, hat der die Hose mit der richtigen Grösse gefunden.
- Wenn die Hose zu kurz und zu breit ist, ist die passende Hose in Bereich 1.
- Wenn die Hose zu kurz ist aber die richtige Breite hat, ist die passende Hose in Bereich 2.
- Wenn die Hose zu kurz und zu schmal ist, ist die passende Hose in Bereich 3.
- Dies kann man für die Bereiche 4 bis 8 fortführen.



Nehmen wir an, die Hose mit der richtigen Grösse ist in Bereich 1. Christian wählt für die zweite Anprobe die Hose in der Mitte von Bereich 1.

Nun gibt es wieder mehrere Möglichkeiten:

- Wenn die Hose passt, hat er die richtige Grösse gefunden.
- Wenn die Hose immer noch zu kurz und zu breit ist, weiss Christian, dass die Hose an Position A die richtige Grösse hat.
- Wenn die Hose zu kurz ist, aber die passende Breite hat, weiss Christian, dass die Hose an Position B die richtige Grösse hat.
- Dies kann man für die anderen Positionen rund um die Mitte von Bereich 1 fortführen.

Weil in jedem nummerierten Bereich das mittlere Regalfach in jeder Richtung nur ein Nachbarfach hat, sind keine weiteren Anproben notwendig. Christian braucht also in jedem Fall höchstens zwei Anproben, um die richtige Grösse zu wissen.

Dies ist Informatik!

Die Methode, die Christian bei der Anprobe anwendet, heisst in der Informatik binäre Suche. Der Begriff *binär* kommt vom lateinischen Wort bis (zweimal). Bei der binären Suche nach einem Objekt in einer Folge sortierter Objekte wird deren mittleres Objekt mit dem gesuchten verglichen. Wenn das mittlere Objekt nicht schon das gesuchte ist, weiss man immerhin, in welcher Hälfte der Folge sich das gesuchte Objekt befindet und durchsucht diese Hälfte wieder binär. In jedem Schritt wird die Folge also in zwei Teile geteilt – deshalb «binär». Auf diese Weise kommt man sehr schnell beim gesuchten Objekt an. Bei 1.000 Objekten werden etwa 10 Suchschritte benötigt, bei 1.000.000 Objekten etwa 20. Allgemein kann man sagen: Bei n Objekten werden etwa $\log(n)$ Schritte benötigt; die Funktion log ist der «Zweier-Logarithmus» oder der Logarithmus zur Basis 2. Weil die binäre Suche so schnell ist, wird sie in Computerprogrammen häufig für die Suche in sortierten Daten verwendet.

In dieser Biberaufgabe ist der Suchraum, nämlich die Hosen im Regal, in zwei Dimensionen (Länge und Breite) sortiert. Deshalb kann Christian die binäre Suche gleich auf beide Dimensionen anwenden. Dann teilt sich die Suchmenge in einem Schritt nicht in 2, sondern gleich in 8 Teile auf - falls Christian nicht direkt die richtige Grösse erwischt hat.

Stichwörter und Webseiten

- Binäre Suche: `https://de.wikipedia.org/wiki/Binäre_Suche`
- Suchverfahren: `https://de.wikipedia.org/wiki/Suchverfahren`

13. Konflikt-Detektor

Anna und Ben wollen einen «Konflikt-Detektor» bauen, der anzeigt, ob sie eine unterschiedliche Meinung haben.

Sie verwenden Einheiten, die in zwei Zuständen sein können: Ja und Nein. Zwei Einheiten können mit einem Kabel verbunden werden.

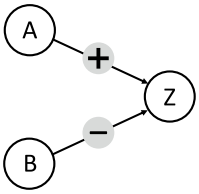
Wenn eine Einheit im

- Zustand Ja ist, sendet sie über alle ausgehenden Kabel ein Signal.
- Zustand Nein ist, sendet sie kein Signal.

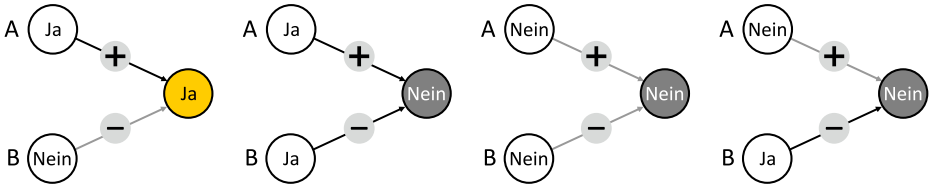
Die Kabel werden so eingestellt, dass sie ein Signal als positives (+) oder negatives (–) Signal an die rechts angeschlossene Einheit übermitteln.

Eine angeschlossene Einheit geht in den Zustand Ja, wenn sie mehr positive als negative Signale empfängt, und sonst in den Zustand Nein. Als Eingabe setzt Anna den Zustand der Einheit A und Ben den Zustand der Einheit B.

Zuerst bauen Anna und Ben diese Maschine:

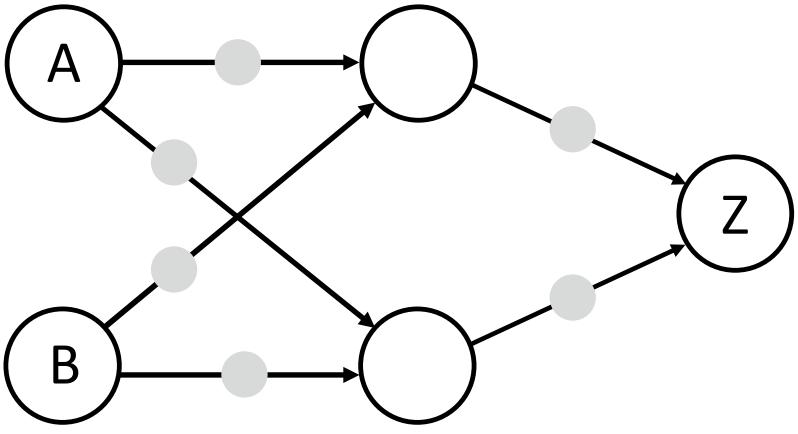


Sie bemerken, dass die Einheit Z nur dann Ja ist, wenn A Ja und B Nein ist. Das ist nicht das, was sie wollen.



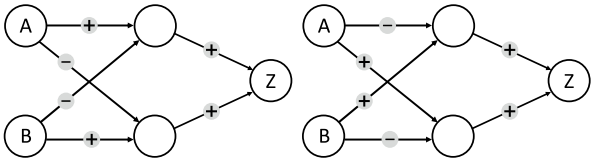
Dann bauen Anna und Ben eine grössere Maschine (unten im Bild) und sind sicher, dass sie der Konflikt-Detektor sein kann: Z soll nur dann Ja sein, wenn A und B in unterschiedlichen Zuständen sind (Ja und Nein bzw. Nein und Ja). Ansonsten soll Z im Zustand Nein sein. Jetzt müssen nur noch die Kabel richtig eingestellt werden.

Stelle für jedes Kabel ein, ob es ein Signal positiv (+) oder negativ (–) übermittelt, damit der Konflikt-Detektor korrekt arbeitet.

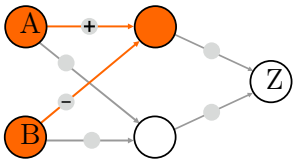


Lösung

Diese beiden Antworten sind richtig:

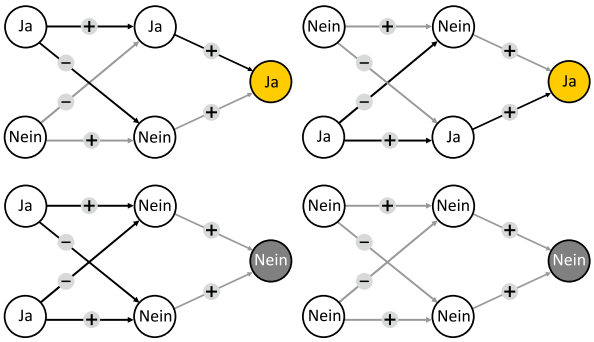


Im Konflikt-Detektor muss die Ausgabe-Einheit genau bei zwei unterschiedlichen Eingaben (A=Ja und B=Nein sowie A=Nein und B=Ja) auf Ja sein. Z kann nur Ja sein, wenn über die zwei eingehenden Kabel mehr positive als negative Signale ankommen. Mindestens eines der Kabel muss also ein positives Signal (+) übermitteln. Nehmen wir einmal an, nur das obere Kabel, das zu Z führt, wird auf + gestellt. Dann muss die Einheit oben Mitte beide gewünschten Eingabekombinationen erkennen können, also in beiden Fällen Ja sein. Zusammen mit den Eingabeeinheiten A und B bildet diese Einheit aber genau so eine Maschine, wie Anna und Ben sie zu Beginn gebaut haben. Sie kann nur in genau einem der gewünschten Fälle Ja sein, und zwar, wenn eines ihrer Kabel auf + und das andere auf – gestellt wird:



Es wird also für jeden der gewünschten Eingabefälle eine eigene Einheit in der Mitte benötigt, eine für A=Ja und B=Nein, die andere für A=Nein und B=Ja. Die Kabel zur ersten Einheit müssen auf + (Kabel von A) und – (B) gestellt werden, die Kabel zur anderen Einheit auf – (A) und + (B). Welche Einheit in der Mitte welchen Fall übernimmt, ist egal; deshalb gibt es bei den Kabeln von A und B zur Mitte zwei Möglichkeiten. Wenn nun jede Einheit in der Mitte in genau einem gewünschten Fall Ja ist, müssen beide Kabel von der Mitte zu Z auf + gestellt werden; nur dann ist Z=Ja in genau beiden gewünschten Fällen.

Für die erste richtige Antwort zeigt das Bild unten die Funktion des Konflikt-Detektors. Man sieht: Die obere Einheit in der Mitte erkennt den Fall A=Ja und B=Nein, die untere den Fall A=Nein und B=Ja. Die jeweilige Einheit sendet ein positives Signal zu Z, und Z ist Ja. Für die anderen Eingaben (A=Ja und B=Ja sowie A=Nein und B=Nein) sind beide mittleren Einheiten Nein, Z empfängt kein positives Signal und ist damit auch Nein.



Dies ist Informatik!

Der Konflikt-Detektor verarbeitet zwei Eingabewerte (Ja oder Nein) und liefert die Ausgabe Ja genau dann, wenn die beiden Eingabewerte unterschiedlich sind. Diese logische Funktion nennt man Exklusiv-Oder (XOR, Kontravalenz). Die erste in dieser Biberaufgabe beschriebene Maschine von Anna und Ben (zwei Schalter und eine Ausgabe-Einheit) ist eine vereinfachte Version eines *Perzeptrons*, das Frank Rosenblatt im Jahr 1957 beschrieben hat. Die Ausgabe-Einheit bildet eine Nervenzelle (Neuron) nach, die Eingabesignale verarbeiten kann und ein Ausgabesignal erzeugt. Mit einem Perzeptron kann man zwar die logischen Operationen Und und Oder implementieren, nicht aber das Exklusiv-Oder. Dazu benötigt man eine weitere Schicht von Schalteinheiten wie in der Lösung dieser Aufgabe. Erst in den 1980er Jahren hat man das erkannt (z.B. Rumelhart, Hinton & Williams 1986) und war dann (später) in der Lage, künstliche neuronale Netze zu programmieren, die ähnlich wie das menschliche Gehirn arbeiten und z. B. Kamerabilder auswerten und Objekte erkennen können. Die Informatik hat Methoden entwickelt, wie grosse neuronale Netze mit vielen Schichten und Einheiten ihre Berechnungen effizient durchführen können. Solche Netze bilden die Grundlage vieler aktueller KI-Systeme.

Stichwörter und Webseiten

- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533−536 : `http://www.cs.toronto.edu/~hinton/absps/naturebp.pdf`
- Perzeptron, Überblick: `https://de.wikipedia.org/wiki/Perzeptron`
- Tutorial zur Programmierung eines Perzeptrons: `https://neuromant.de/2018/11/25/Tutorial_Das-Perzeptron/`
- Exklusive-Oder (Kontravalenz): `https://de.wikipedia.org/wiki/Kontravalenz`

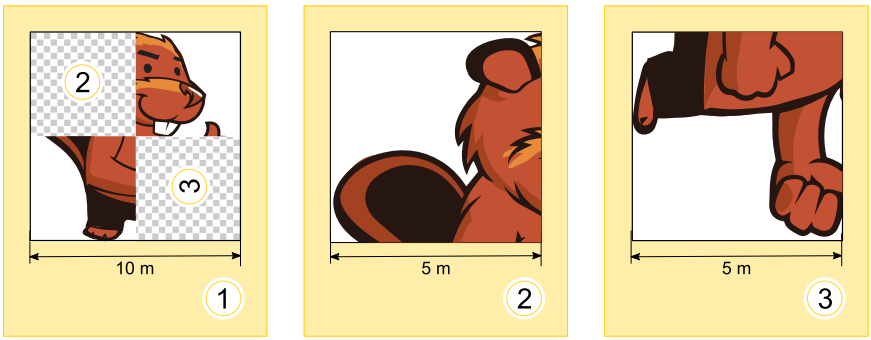
14. Rekursiv malen

Tina und Ben helfen bei der Vorbereitung einer Sonderausstellung im Informatik-Museum. Auf den Boden eines Ausstellungsraums sollen sie ein 16×16 Meter grosses Bild malen.

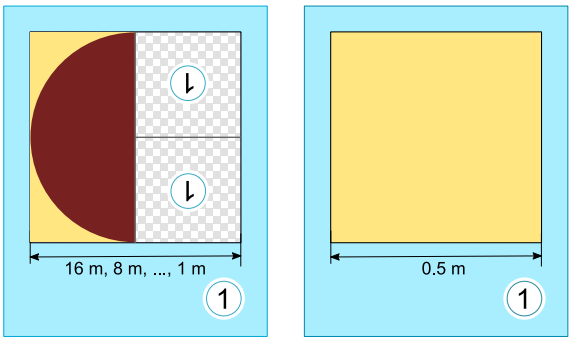
Vom Künstler bekommen sie einen Satz Malanweisungskarten in dessen berühmter Malkartensprache, mit Hinweisen zu den Bildelementen, Massen und Drehungen.

Auf manchen Malanweisungskarten sind nummerierte Felder, die auf andere Karten verweisen.

Hier ein Beispiel aus einem früheren Malkartenprojekt. Wenn man diese drei Karten richtig ausführt, entsteht ein Bild des Bibers:

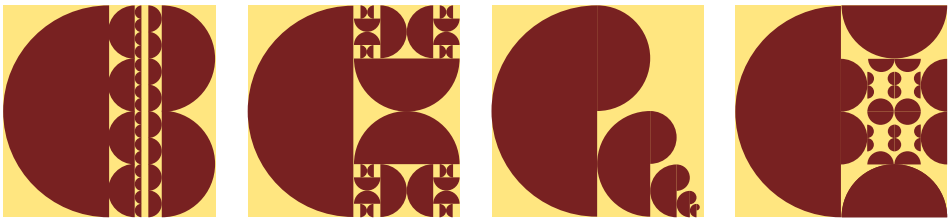


Für die Sonderausstellung bekommen Tina und Ben nun diese zwei Karten:



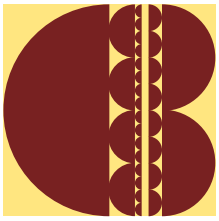
Ben runzelt die Stirn. «Wie soll das gehen? Die linke Karte verweist auf sich selbst, und ausserdem haben beide Karten dieselbe Nummer!» Tina lacht: «Wir kriegen das hin! Zuerst verwenden wir nur die linke Karte. Die rechte Karte wird uns später anweisen, wann wir mit dem Malen aufhören sollen.»

Wie wird der Boden des Ausstellungsraums aussehen?



A) B) C) D)

Lösung

Antwort A ist richtig: 

Die linke Malanweisungskarte besagt, dass die linke Hälfte des Bodens mit einer Halbkreisfläche gefüllt werden soll, deren runde Seite bei normaler Orientierung nach links zeigt. Für die rechte Hälfte soll dieselbe Malanweisungskarte zwei Mal verwendet werden. Die Orientierungen der Bildelemente muss den Orientierungen der Einsen entsprechen.

In beiden Hinweisen auf Karte 1 ist die 1 um 180° gedreht, nach unten. Deshalb müssen die dort eingesetzten Bildelemente ebenfalls so gedreht werden, so dass die Rundung der dann gemalten Halbkreisflächen genau in die Gegenrichtung zeigt. Bei der ersten Verwendung der linken Karte 1 (bei der Breite 16 m) zeigt die Rundung der Halbkreisfläche nach links, bei 8 m nach rechts, bei 4 m wieder nach links usw. Bei 0,5 m wird die zweite Karte 1 verwendet: Ben und Tina malen die verbleibende freie Fläche noch aus und können danach aufhören.

Auf diese Weise entsteht genau das Bild von Antwort A.

Dies ist Informatik!

Die erste Malanweisungskarte 1 in dieser Biberaufgabe verweist auf sich selbst. Sie ruft Ben und Tina sozusagen dazu auf, sie selbst erneut anzuwenden, mit einer geringeren Breite. In der Informatik werden Anweisungen, die sich selbst aufrufen, als «rekursiv» bezeichnet. Der Begriff kommt von lateinisch «recurrere» (deutsch: zurücklaufen). Rekursion ist ein mächtiges Konzept. Für manche komplexen Aufgaben kann man kurz und überschaubar eine rekursive Anweisung zu ihrer Lösung formulieren.

Eine rekursive Anweisung muss eine Bedingung enthalten, die festlegt, wann die Rekursion abgebrochen werden soll. Sonst arbeitet die Rekursion solange weiter, bis irgendeine Ressource erschöpft ist. Etwa der Computerspeicher oder die Geduld des Benutzers. In dieser Biberaufgabe hat die zweite Karte 1 diese Funktion: Unter der Bedingung, dass noch eine Fläche der Breite 0,5 m bemalt werden soll, wird sie «aufgerufen». Da sie keinen Verweis auf eine andere Karte enthält, bricht sie die Rekursion ab.

Stichwörter und Webseiten

- Programmieren: `https://de.wikipedia.org/wiki/Programmierung`
- Rekursion: `https://de.wikipedia.org/wiki/Rekursion`
- Abbruchbedingung: `https://de.wikipedia.org/wiki/Abbruchbedingung`

15. Zerteile den Code

In einem speziellen Code für Texte wird jeder Buchstabe durch ein Codewort aus den Ziffern 0 bis 9 kodiert. Dabei gilt diese Regel: Kein Codewort darf mit dem Codewort eines anderen Buchstabens beginnen.

Der Buchstabe `X` wird beispielsweise durch 12 kodiert. Nun kann `Y` durch 2 kodiert werden, denn 12 beginnt nicht mit 2 (und 2 nicht mit 12). Jetzt kann `Z` durch 11 kodiert werden; denn weder 12 noch 2 beginnen mit 11 und 11 beginnt weder mit 12 noch mit 2. 21 wäre jedoch nicht als Codewort für `Z` erlaubt, weil es mit 2, also dem Codewort von `Y` beginnt.

Das Wort `MEMORY` wird durch die Ziffernfolge 12112233321 kodiert.

Teile die Ziffernfolge in die Codewörter der einzelnen Buchstaben!

Lösung

So ist es richtig: 1 21 1 22 33 321

Man beginnt links am Anfang der Ziffernfolge. Falls `M` durch 12 kodiert würde, hätte `E` zwangsläufig das Codewort 1, denn dahinter käme wieder 12 für das zweite `M`. Das würde jedoch der Regel widersprechen: Das Codewort für `M` würde dann mit 1 beginnen, dem Codewort für E. Längere Anfangsstücke der Ziffernfolge (121, 1211, 12112 etc.) können auch nicht das Codewort für `M` sein, weil dieses Codewort zwei Mal vorkommen muss, diese Stücke aber jeweils nur einmal in der Ziffernfolge enthalten sind. Folglich ist das Codewort für `M` die Ziffer 1.

Nun muss das Codewort für `E` folgen - und dahinter wieder das für M, also die 1. Somit kann das Codewort für `E` nur eine der folgenden Ziffernfolgen sein: 2, 21 oder 211223332. Es kann nicht 2 sein; dann würde das Wort mit MEMM beginnen. Es kann nicht 211223332 sein; dann wäre das Wort insgesamt nur MEM. Folglich ist das Codewort für `E` die Ziffernfolge 21. Nun ist klar, dass 1 21 1 die Kodierung für MEM ist.

Der Rest der Ziffernfolge, also 2233321, kodiert die Buchstaben ORY. Die 2 alleine kann nicht das Codewort für `O` sein, sonst hätten wir `OO` zu Beginn. Das Codewort für `O` muss also mindestens die 22 beinhalten. Am Ende wiederum sind 1 und 21 schon als Codewörter für `M` bzw. `E` vergeben. Das Codewort für `Y` muss also mindestens die Folge 321 sein. Zwischen 22 und 321 steht 33. Das muss das Codewort für `R` sein: Das einzig andere noch möglich Codewort für `R` wäre 3. Dann müsste 3321 das Codewort für `Y` sein – und würde mit dem Codewort für `R` beginnen; das ist gegen die Regel. Die Aufteilung des hinteren Teils ist also 22 33 321.

Dies ist Informatik!

Der Code, der in dieser Biberaufgabe beschrieben wird, ist ein Beispiel für einen *Präfixcode*. Ein Präfix ist eine Zeichenfolge zu Beginn einer anderen Zeichenfolge. Bei einem Präfixcode darf kein Codewort Präfix eines anderen Codeworts sein. Das heisst: kein Codewort darf mit einem anderen Codewort beginnen.

Bei Präfixcodes haben die Codewörter unterschiedliche Länge. Der Vorteil der Präfix-Regel ist, dass man keine Trennsymbole zwischen Codewörtern benötigt. Man kann immer erkennen, an welcher Stelle das nächste Codewort beginnt. Wenn man kurze Codewörter für häufig vorkommende Buchstaben wählt, kann man Texte sehr effizient kodieren und grosse Textmengen platzsparend speichern.

Die Huffman-Kodierung ist eine Methode, einen optimalen Präfixcode zu finden. Sie ist weit verbreitet und steckt z.B. hinter bekannten Datenformaten wie JPEG und MP3.

Stichwörter und Webseiten

- Präfixcode: `https://de.wikipedia.org/wiki/Präfixcode`
- Huffman-Kodierung: `https://de.wikipedia.org/wiki/Huffman-Kodierung`
- Kryptographie: `https://de.wikipedia.org/wiki/Kryptographie`

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

A. Aufgabenautoren

Eslam AbdElAal
Akram Ahmed
Nursultan Akhmetov
Khairul Anwar
James Atlas
Leonardo Barichello
Wilfried Baumann
Tim Bell
Leonardo Cavalcante
Zaheer Chothia
Eimear Colreavy
Raluca Constantinescu
Kris Coolsaet
Valentina Dagienė
Christian Datzko
Justina Dauksaite
Nora A. Escherle
Georgios Fesakis
Gerald Futschek
Hans-Werner Hein
Tracy Henderson
Thomas Ioannou
Filiz Kalelioğlu
Merel Kämper
Gohar Khachatryan
Jihye Kim
Víctor Koleszar
Taina Lehtimäki
Michael Weigend
Dario Malchiodi
Yong Mao
Anna Morpurgo
Jalil Nedaeepour
Özgür Özdemir
Zsuzsa Pluhár
Wolfgang Pohl
Ilya Posov
Sergey Pozdniakov
JP Pretti
Omar Colon Reyes
Chris Roffey
Karima Sayeh
Margareta Schlüter
Eljakim Schrijvers
Vipul Shah
Rostyslav Shpakovych
Jacqueline Staub
Alieke Stijf
Ahto Truu
Laura Ungureanu
Jiří Vaníček
Michael Weigend

 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.

Das Lehrmittel zum Informatik-Biber

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

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

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

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

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

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