INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Aufgaben und Lösungen 2018
Schuljahre 11/12/13

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

Herausgeber:
Christian Datzko, Susanne Datzko, Hanspeter Erni

AEIOUHWY	X
BFPV	1
CGJKQSXZ	2
DT	3
L	4
NM	5
R	6

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 2018

Andrea Adamoli, Christian Datzko, Susanne Datzko, Olivier Ens, Hanspeter Erni, Martin Guggisberg, Carla Monaco, Gabriel Parriaux, Elsa Pellet, Jean-Philippe Pellet, Julien Ragot, Beat Trachsler.

Herzlichen Dank an:
Juraj Hromkovič, Urs Hauser, Regula Lacher, Jacqueline Staub: ETHZ
Andrea Maria Schmid, Doris Reck: PH Luzern
Gabriel Thullen: Collège des Colombières
Valentina Dagienė: Bebras.org
Hans-Werner Hein, Ulrich Kiesmüller, Wolfgang Pohl, Kirsten Schlüter, Michael Weigend: Bundesweite Informatikwettbewerbe (BWINF), Deutschland
Chris Roffey: University of Oxford, Vereinigtes Königreich
Anna Morpurgo, Violetta Lonati, Mattia Monga: ALaDDIn, Università degli Studi di Milano, Italien
Gerald Futschek, Wilfried Baumann: Oesterreichische Computer Gesellschaft, Österreich
Zsuzsa Pluhár: ELTE Informatikai Kar, Ungarn
Eljakim Schrijvers, Daphne Blokhuis, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers: Eljakim Information Technology bv, Niederlande
Roman Hartmann: hartmannGestaltung (Flyer Informatik-Biber Schweiz)
Christoph Frei: Chragokyberneticks (Logo Informatik-Biber Schweiz)
Andrea Adamoli (Webseite)
Andrea Leu, Maggie Winter, Brigitte Maurer: Senarclens Leu + Partner

Die deutschsprachige Fassung der Aufgaben wurde ähnlich auch in Deutschland und Österreich verwendet.
Die französischsprachige Übersetzung wurde von Nicole Müller und Elsa Pellet und die italienischsprachige Übersetzung von Andrea Adamoli erstellt.

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Der Informatik-Biber 2018 wurde vom Schweizerischen Verein für Informatik in der Ausbildung SVIA durchgeführt und von der Hasler Stiftung unterstützt.

HASLERSTIFTUNG

Hinweis: Alle Links wurden am 1. November 2018 geprüft. Dieses Aufgabenheft wurde am 19. Januar 2019 mit dem Textsatzsystem LaTeX erstellt.

Vorwort

Der Wettbewerb „Informatik-Biber“, der in verschiedenen Ländern der Welt schon seit mehreren Jahren bestens etabliert ist, will das Interesse von Kindern und Jugendlichen an der Informatik wecken. Der Wettbewerb wird in der Schweiz in Deutsch, Französisch und Italienisch vom Schweizerischen Verein für Informatik in der Ausbildung SVIA durchgeführt und von der Hasler Stiftung im Rahmen des Förderprogramms FIT in IT unterstützt.
Der „Informatik-Biber“ ist der Schweizer Partner der Wettbewerbs-Initiative „Bebras International Contest on Informatics and Computer Fluency“ (`https://www.bebras.org/`), die in Litauen ins Leben gerufen wurde.
Der Wettbewerb wurde 2010 zum ersten Mal in der Schweiz durchgeführt. 2012 wurde zum ersten Mal der „Kleine Biber“ (Stufen 3 und 4) angeboten.
Der „Informatik-Biber“ regt Schülerinnen und Schüler an, sich aktiv mit Themen der Informatik auseinander zu setzen. Er will Berührungsängste mit dem Schulfach Informatik abbauen und das Interesse an Fragenstellungen dieses Fachs wecken. Der Wettbewerb setzt keine Anwenderkenntnisse im Umgang mit dem Computer voraus – ausser dem „Surfen“ auf dem Internet, denn der Wettbewerb findet online am Computer statt. Für die Fragen ist strukturiertes und logisches Denken, aber auch Phantasie notwendig. Die Aufgaben sind bewusst für eine weiterführende Beschäftigung mit Informatik über den Wettbewerb hinaus angelegt.
Der Informatik-Biber 2018 wurde in fünf Altersgruppen durchgeführt:

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

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

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

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

Für weitere Informationen:

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

Inhaltsverzeichnis

1. Computerspiel

Andrea hat ein Computerspiel in der Schule programmiert. Die Spielregeln sind ganz einfach:
Das Spiel besteht aus mehreren Spielrunden. In jeder Spielrunde fällt ein Blatt. Der Biber versucht das Blatt zu fangen, bevor es den Boden erreicht. Um zu gewinnen, muss der Biber 15 Blätter fangen, bevor 4 Blätter den Boden berühren.
Die Länge des Spiels wird in der Anzahl der Spielrunden gemessen.
Im folgenden Beispiel verliert der Biber nach 6 Spielrunden, weil das Maximum von 4 nicht gefangenen Blättern erreicht ist. Die Länge dieses Beispiels beträgt 6 Spielrunden.

Spielrunde	Resultat	Spielstand – Total Anzahl Blätter	
		Gefangen	Nicht gefangen
1	gefangen	1	0
2	nicht gefangen	1	1
3	gefangen	2	1
4	nicht gefangen	2	2
5	nicht gefangen	2	3
6	nicht gefangen	2	4

Wie lange kann ein Spiel maximal dauern?

A) 4 Spielrunden

B) 15 Spielrunden

C) 18 Spielrunden

D) 19 Spielrunden

E) 20 Spielrunden

F) Die Spiellänge ist unbegrenzt.

Lösung

Um das längstmögliche Spiel zu finden, müssen wir alle Situationen kombinieren, in denen das Spiel weitergeht. Dazu kombinieren wir die maximal gefangenen Blätter vor Spielende (14 Spielrunden) mit den maximal nicht gefangenen Blättern vor Spielende (3 Spielrunden). Danach wird entweder ein 15. Blatt gefangen oder ein 4. Blatt verloren. Daher ist die maximale Länge $15+3 = 14+4 = 18$ Runden und die richtige Antwort ist C).
Die Antwort A) „4 Runden“ wäre die Mindestlänge des Spiels (wenn alle Blätter nicht gefangen werden).
Die Antwort B) wäre die Mindestlänge, um das Spiel zu gewinnen (wenn alle Blätter gefangen werden).
Die Antworten D), E) und F) sind falsch, da das Maximum der gefangenen oder das Maximum der nicht gefangenen Blätter vorher erreicht würde.

Dies ist Informatik!

Bei der Programmierung eines Spiels müssen die Regeln klar definiert sein. Die Auswirkungen der Regeln müssen vollständig verstanden werden, so dass das Spiel so aufgebaut werden kann, dass Gewinnen oder Verlieren möglich ist (ausreichende Blätter sind verfügbar), und dass das Spiel weder zu kurz noch zu lang dauert.
Ein Spiel, das aus mehreren Runden besteht, ist ein Prozess. Informatiker sind Spezialisten in der Modellierung und Beschreibung von Prozessen. Eine der Hauptaufgaben besteht darin herauszufinden, was alles passieren kann, und wie lange ein Prozess laufen kann.

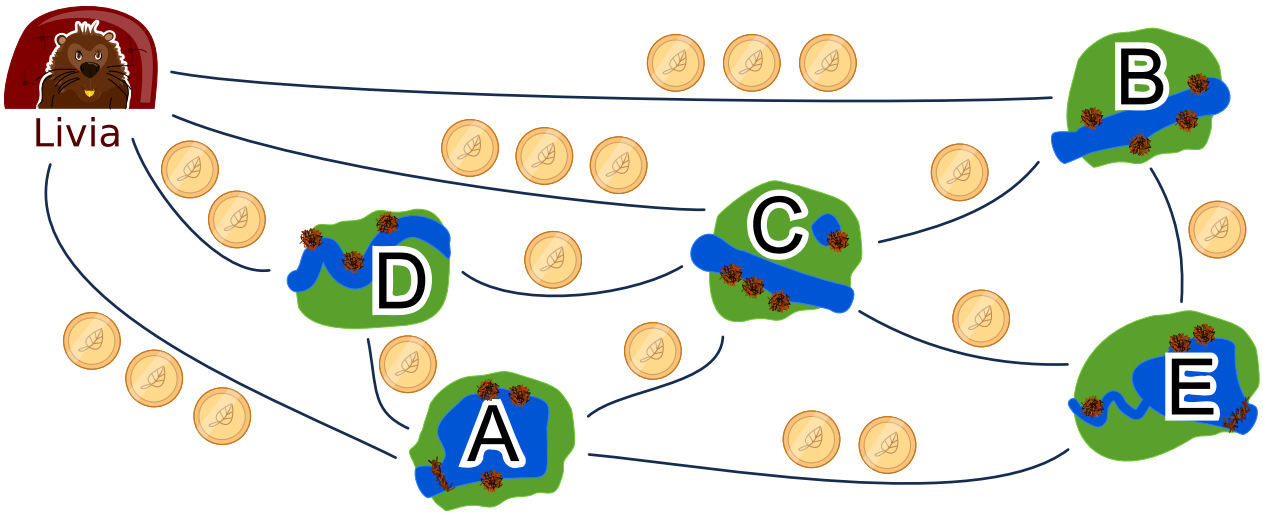
Stichwörter und Webseiten

Analyse, Verifizierung und Validierung von Software

- `https://en.wikipedia.org/wiki/Software_verification`
- `https://de.wikipedia.org/wiki/Verifizierung_und_Validierung`

2. Biberbesuch

Livia möchte alle ihre Freunde in den Dörfern A, B, C, D und E mit öffentlichen Verkehrsmitteln besuchen. Sie besucht alle ihre Freunde auf einer einzigen Reise, ohne ein Dorf mehr als einmal zu besuchen. Am Ende ihrer Reise kehrt sie nach Hause zurück. Der Fahrpreis jeder Linie ist unten angezeigt.



Ein möglicher Weg, ihre Freunde zu besuchen ist:

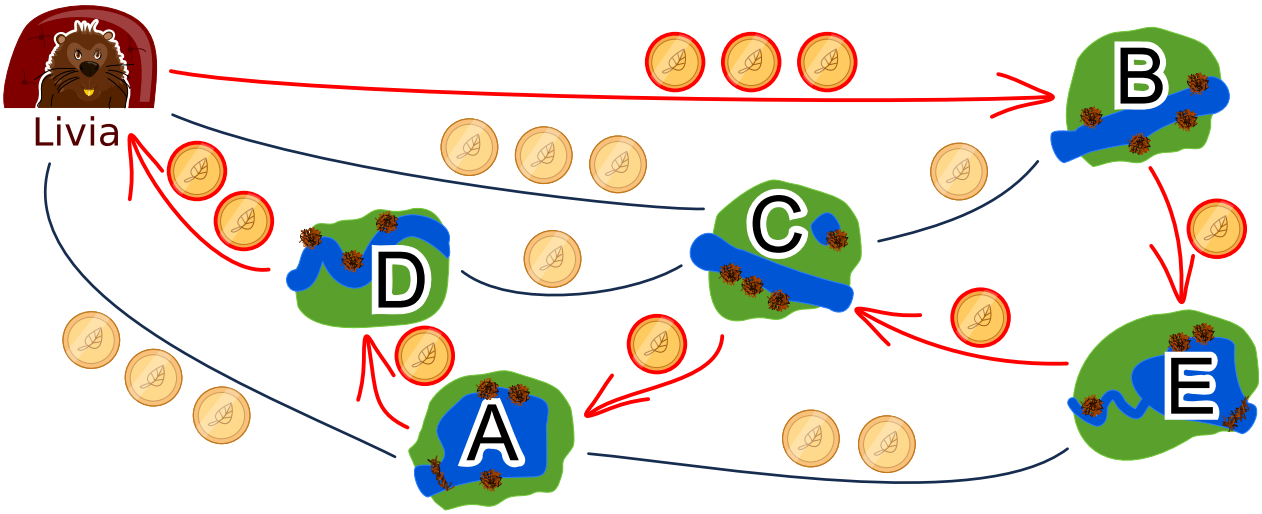
$$\text{Start} \to \text{B} \to \text{E} \to \text{A} \to \text{D} \to \text{C} \to \text{Start}.$$

Dieser Weg kostet $3 + 1 + 2 + 1 + 1 + 3 = 11$ Bibermünzen.
In welcher Reihenfolge muss Livia die Freunde besuchen, damit sie möglichst wenige Münzen bezahlen muss?

Lösung

Es gibt zwei optimale Lösungen:

- Start → B → E → C → A → D → Start
- Start → D → A → C → E → B → Start



Die zwei Lösungen sind bis auf die Richtung gleich und kosten 9 Bibermünzen. Es gibt keine bessere Lösung, denn von Livias Zuhause aus kannst du einmal den Zwei-Bibermünzen-Weg und dann einen Drei-Bibermünzen-Weg gehen. Zu den vier weiteren Knoten gehören vier Wege, die jeweils mindestens eine Bibermünze kosten, was bereits 9 Bibermünzen ergibt.
Alle anderen Lösungen kosten mehr:

- Kosten von 10 Bibermünzen: Start → A → D → C → E → B → Start
- Kosten von 10 Bibermünzen: Start → A → E → B → C → D → Start
- Kosten von 10 Bibermünzen: Start → B → C → E → A → D → Start
- Kosten von 10 Bibermünzen: Start → B → E → A → C → D → Start
- Kosten von 10 Bibermünzen: Start → B → E → C → D → A → Start
- Kosten von 10 Bibermünzen: Start → C → B → E → A → D → Start
- Kosten von 10 Bibermünzen: Start → D → A → E → B → C → Start
- Kosten von 10 Bibermünzen: Start → D → A → E → C → B → Start
- Kosten von 10 Bibermünzen: Start → D → C → A → E → B → Start
- Kosten von 10 Bibermünzen: Start → D → C → B → E → A → Start
- Kosten von 11 Bibermünzen: Start → B → E → A → D → C → Start
- Kosten von 11 Bibermünzen: Start → C → D → A → E → B → Start

Eine Methode, den günstigsten Rundweg zu finden, besteht darin, einen Weg zu gehen, der die minimale Menge an Bibermünzen kostet, und dann von dort aus eine Lösung zu finden.

Dies ist Informatik!

Nach guten oder sogar optimalen Lösungen zu suchen, ist eine der grundlegenden Aufgaben der Informatik. Wir können die Beschreibung dieser Optimierungsaufgabe in einem Graphen visualisieren, in dem Freunde Knoten und die Strassen Kanten sind. Die Aufgabe besteht darin, alle Knoten genau einmal so zu besuchen, dass die Summe der Kantengewichte (die Kosten in Bibermünzen) minimal sind. Dies ist ähnlich dem berühmten Travelling Salesman Problem (TSP).
Diese Art von Problemen sind normalerweise sehr schwierig mit einem Computer zu lösen. Um zu vermeiden, dass jede einzelne Lösung ausprobiert werden muss, kann man eine gute Heuristik verwenden (eine Heuristik ist zum Beispiel, zuerst den kürzesten Weg zu nehmen) und alle Lösungen, die schlechter werden, zu streichen. In diesem Fall erlauben wir, dass jeder Knoten nur einmal besucht werden darf. Wenn wir einen Knoten mehr als einmal besuchen dürfen, wird das Problem tatsächlich schwieriger, weil wir viel mehr Alternativen in Betracht ziehen müssen.

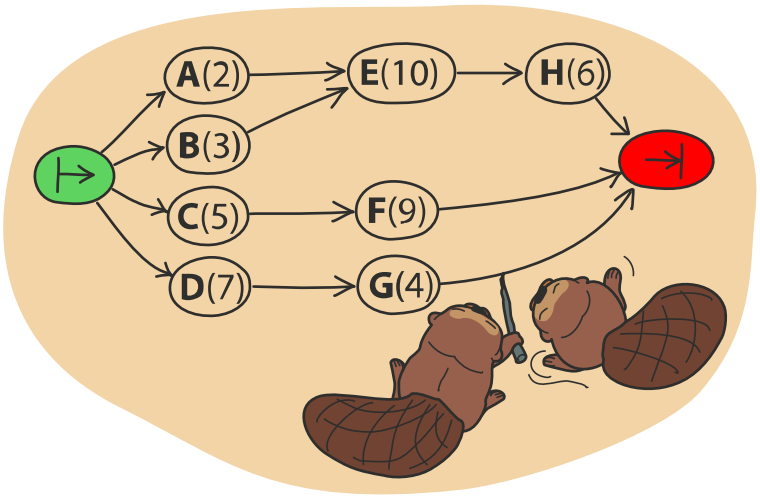
Stichwörter und Webseiten

Optimierung, Problem eines Handlungsreisenden

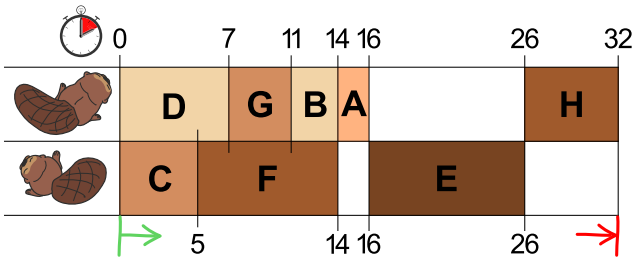
- `https://de.wikipedia.org/wiki/Problem_des_Handlungsreisenden`
- `https://de.wikipedia.org/wiki/Optimierungsproblem`

3. Zwei Biber bei der Arbeit

Zwei Biber bauen einen Damm und müssen dazu acht Aufgaben lösen: Bäume fällen, von den Stämmen die Äste entfernen, Stämme ins Wasser bringen, und so weiter. Für jede Aufgabe gibt es einen Buchstaben als Namen und eine Zahl in Klammern, die die nötige Anzahl der Arbeitsstunden angibt.
Einige Aufgaben können erst dann begonnen werden, wenn bestimmte andere Aufgaben bereits vollständig gelöst worden sind. Diese Abfolge wird durch die Pfeile dargestellt. Die Biber können parallel verschiedene Aufgaben bearbeiten, es kann aber immer nur einer an einer Aufgabe arbeiten.



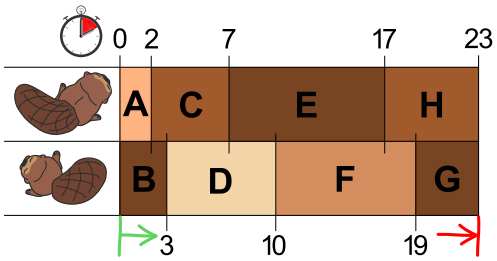
Die Abbildung unten zeigt einen möglichen Arbeitsplan der beiden Biber, der 32 Stunden benötigt. Es geht aber schneller!



Was ist die kürzeste Zeit, in der die Biber einen Damm bauen können?

Lösung

Es sind mindestens 23 Stunden erforderlich.
Das Bild in der Aufgabe zeigt einen möglichen Arbeitsplan der beiden Biber. Dort hat der erste Biber eine lange Pause von 10 Stunden und der zweite Biber zwei Pausen über insgesamt 8 Stunden. Wenn beide die ganze Zeit arbeiten würden, wären sie schneller fertig.



Wenn man darauf achtet, dass die beiden grössten Aufgaben E(10) und F(9) nicht von demselben Biber ausgeführt werden, findet man leicht einen Arbeitsplan, der mit 23 Stunden auskommt. Schneller geht es nicht, denn die beiden Biber arbeiten ohne Pause.

Dies ist Informatik!

Um einen kürzesten Arbeitsplan zu finden, wäre eine Möglichkeit, sich an die folgende Regel zu halten: „Wähle unter den noch verfügbaren Aufgaben immer die mit den meisten Arbeitsstunden“.
In der Informatik nennt man eine solche Strategie “greedy” (engl. für „gierig“). Man löst zuerst die Teilaufgaben, die einen möglichst grossen Fortschritt im Hinblick auf die Gesamtlösung des Problems bedeuten.
In vielen Fällen ist “greedy” eine gute Strategie, aber manchmal – wie bei dieser Aufgabe – funktioniert sie nicht so gut. Diese Aufgabe wurde absichtlich so konstruiert, dass die “greedy”-Strategie nicht funktioniert. Das Finden solcher ungünstigen Problemstellungen ist jedoch auch wichtig: In der theoretischen Informatik beispielsweise sucht man für Computerprogramme gezielt nach dem ungünstigsten Fall (“worst case”), um den Zeitbedarf von Algorithmen besser abschätzen zu können.
Eigentlich gäbe es nur einen sicheren Weg, die beste Lösung zu finden: Man probiert alle denkbaren Arbeitspläne aus, die den vorgegebenen Regeln entsprechen. Bei grösseren Projekten kann aber die Anzahl der Möglichkeiten so gross sein, dass die Entscheidung zu viel Zeit benötigt. Da kommt dann eine Strategie wie “greedy” ins Spiel, denn mit ihr kann man einfach Lösungen finden, die zumindest hinreichend gut sind.

Stichwörter und Webseiten

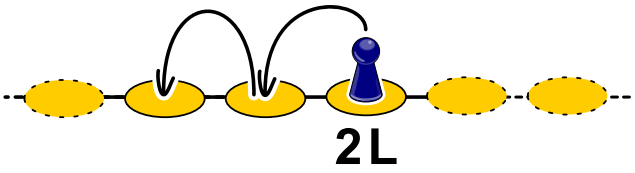
Scheduling, Greedy-Algorithmus

- `https://de.wikipedia.org/wiki/Scheduling`
- `https://de.wikipedia.org/wiki/Topologische_Sortierung`
- `https://de.wikipedia.org/wiki/Greedy-Algorithmus`

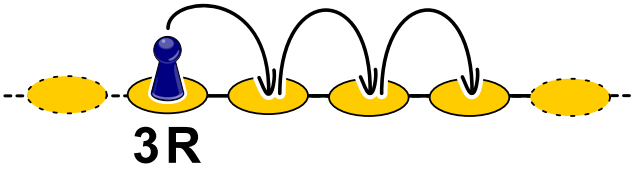
4. Hüpfspiel

Wie bei jedem Hüpfspiel muss man auch hier Felder nach bestimmten Regeln abhüpfen. Bei diesem Hüpfspiel gehört zu jedem Feld eine Regel. Es gibt drei Arten von Regeln:

- nL: n Felder nach links hüpfen, 2L bedeutet also, zwei Felder nach links zu hüpfen:

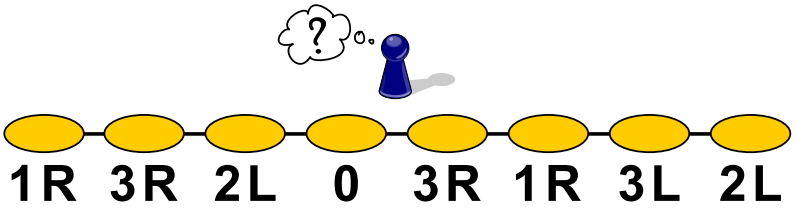


- nR: n Felder nach rechts hüpfen, 3R bedeutet also, drei Felder nach zu rechts hüpfen:



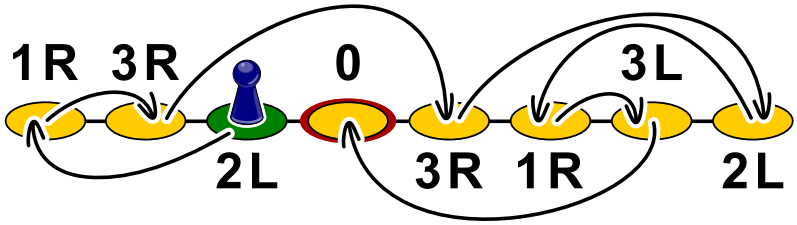
- 0: nicht mehr weiter hüpfen.

Auf welchem Feld muss man starten, damit man nach dem Spiel auf jedem Feld einmal gewesen ist?



Lösung

Wenn man auf dem dritten Feld von links („2L“) startet, wird man am Ende auf jedem Feld einmal gewesen sein.
Man kommt darauf, indem man für das Feld „0“ das Feld sucht, von welchem aus man es erreichen kann. Das ist in diesem Fall das zweite Feld von rechts („3L“). Dieses wiederum erreicht man vom dritten Feld von rechts („1R“), dieses wiederum vom Feld ganz rechts („2L“), dieses vom vierten Feld von rechts („3R“), das vom zweiten Feld von links („3R“), dieses nun vom Feld ganz links („1R“) und zuletzt bleibt nur noch das dritte Feld von links („2L“), das tatsächlich wie gewünscht das Feld ganz links erreichen lässt.



Dieses Markieren der Hüpfwege mit Pfeilen macht aus den Feldern einen gerichteten Graphen, den man nun lediglich von 0 an rückwärts durchgehen muss, um beim dritten Feld von links als Startfeld zu landen.

Dies ist Informatik!

In der Informatik funktioniert das Speicherprinzip *„Verkettete Liste“* ähnlich wie die Hüpffelder: im Arbeitsspeicher werden Objekte mit Verweisen auf die Speicheradresse des nächsten Objekts gespeichert. So kann die Speicherverwaltung ein einzelnes Objekt beliebig im Arbeitsspeicher hinterlegen und muss nicht zeitaufwendig für alle Objekte einen zusammenhängenden Platz im Arbeitsspeicher erstellen. Der Programmierer wiederum muss sich um nichts kümmern, ausser dass er bei der Speicherverwaltung einen Speicher in der entsprechenden Grösse reserviert.
Was passiert aber, wenn Objekte im Speicher nicht mehr gebraucht werden? Während sich früher die Programmierer darum kümmern mussten, dass der Speicher wieder freigegeben wird (was leider häufig schief ging, so dass Programme mit der Zeit immer mehr Speicher verschwendeten und dann irgendwann aus Speichermangel abstürzten), haben moderne Programmiersprachen hierfür eine Art Müllabfuhr, die *„Garbage Collection“*, die regelmässig überprüft, ob Objekte im Speicher noch von anderen Objekten referenziert werden (ob es also noch Verweise auf sie gibt). Da manchmal grosse Strukturen von Objekten nicht mehr referenziert werden, muss genau wie im Beispiel zurückverfolgt werden, von welchem „Startfeld“ aus auf die anderen Objekte verwiesen wird.

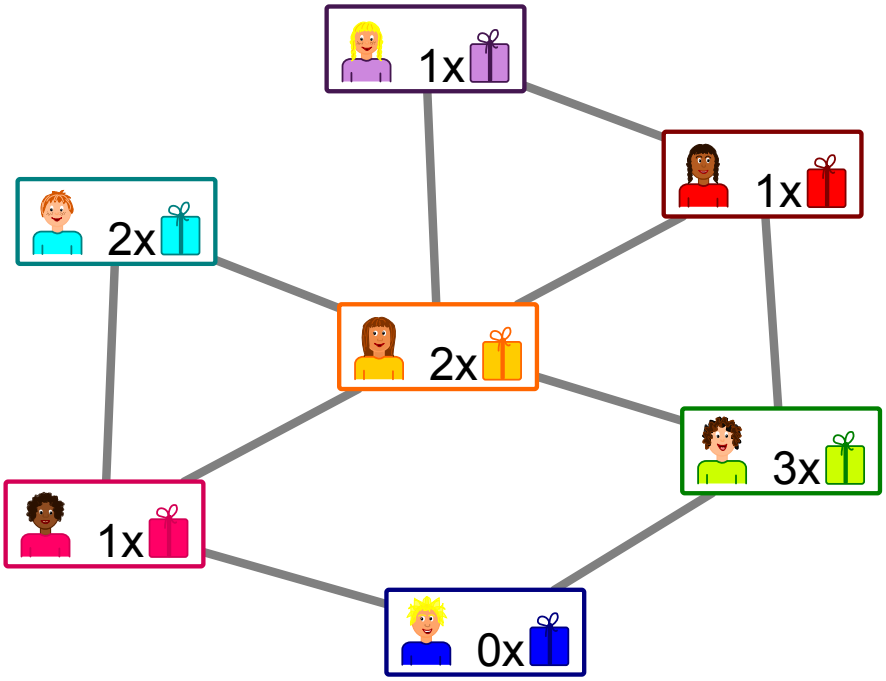
Stichwörter und Webseiten

Schlange, Speicherverwaltung, GOTO

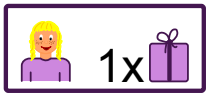
- `https://de.wikipedia.org/wiki/Garbage_Collection`
- `https://de.wikipedia.org/wiki/Erreichbarkeitsproblem_in_Graphen`

5. Geschenke

Das Bild zeigt die Freundschaften zwischen den Kindern in einem Haus. Eine Linie zwischen zwei Freunden bedeutet: Diese Kinder sind Freunde.



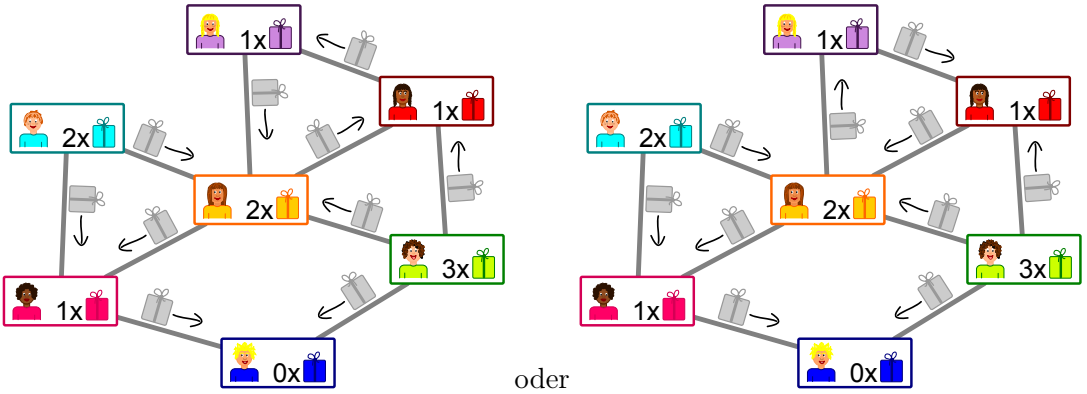
Die Hausbewohner planen ein Kinderfest mit Geschenken. Bei allen Paaren von Freunden soll ein Kind dem anderen Kind ein Geschenk besorgen.

Im Bild steht, wie viele Geschenke das Kind besorgen kann: 

bedeutet, dass das Kind ein Geschenk besorgen kann.
Entscheide für jedes Freundespaar, wer das Geschenk besorgt. Dabei soll kein Kind mehr Geschenke besorgen müssen, als es besorgen kann.

Lösung

Es gibt zwei Möglichkeiten, wie man für alle Paare von Freunden die „Schenkrichtung“ angeben kann, ohne dass ein Kind mehr Geschenke besorgen muss, als es besorgen kann.



Es lohnt sich bei dem Kind unten anzufangen: Es kann kein Geschenk besorgen und erhält deshalb je ein Geschenk von seinen Freundinnen links und rechts. Damit hat die Freundin links ihr Geschenk vergeben und erhält selbst je ein Geschenk von den anderen beiden Kindern, mit denen sie befreundet ist. Für die anderen Freundespaare sind die „Schenkrichtungen“ also klar.
Die einzige Auswahlmöglichkeit, die es noch gibt, ist, ob bei den drei Kindern oben rechts im Uhrzeigersinn oder gegen den Uhrzeigersinn geschenkt wird.

Dies ist Informatik!

Die Freundschaften zwischen den Kindern bilden ein Netzwerk aus Knoten (die Kinder) und Kanten (die Freundschaftsbeziehungen). Ähnlich sind „soziale Netzwerke“ aufgebaut mit Millionen von Benutzenden. Es gibt jedoch einen Punkt, in dem sich diese Systeme grundlegend unterscheiden können: In manchen gibt es wechselseitige „Freundschaften“, in denen die Verbindungen keine Richtung haben, so wie in dieser Aufgabe. In anderen gibt es „Anhänger“ (engl. “follower”), so dass die Verbindungen eine Richtung haben: Wenn du beispielsweise einer berühmten anderen Nutzerin „folgst“, muss sie nicht unbedingt auch dir folgen.
In dieser Aufgabe sollen die Freundschafts-Verbindungen „Schenkrichtungen“ bekommen. Das ist ein neuer Aspekt, denn die „Schenk-Kapazitäten“ der Kinder sind begrenzt und setzen so indirekt der Wahl der Richtungen Grenzen. Das Ziel ist, in jeder Freundschaft ein Geschenk zu schenken, ohne dass die Schenk-Kapazität eines Kindes überschritten wird. In der Informatik gibt es ähnliche Probleme: In einem Netzwerk (etwa die Kabel, die das Internet bilden) sind die Kapazitäten der Verbindungen begrenzt. Innerhalb dieser Grenzen aber ist es am besten, wenn diese Kapazitäten voll ausgenutzt werden.
Das Problem des maximalen Flusses in Netzwerken lässt sich effizient lösen. Da seine Struktur der Struktur unserer Aufgabe gleicht, lassen sich die Lösungsmethoden auch hier anwenden. So etwas kommt in der Informatik häufig vor: Ein Problem wird in ein anderes Problem mit der gleichen Struktur umgewandelt, in der man das Problem bereits einfach lösen konnte.

Stichwörter und Webseiten

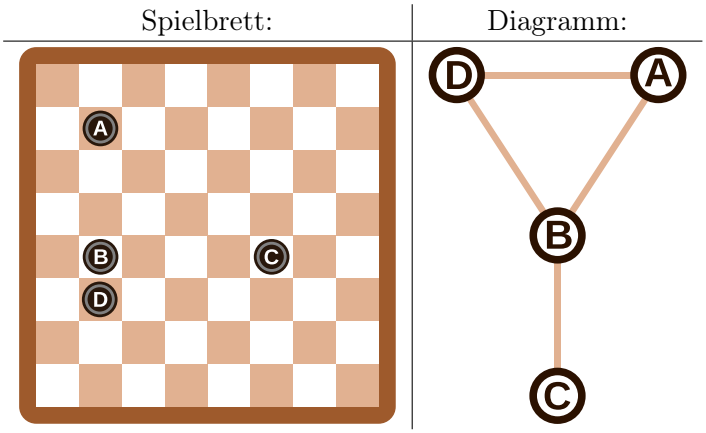
Netzwerkfluss, Problemreduktion

- `https://en.wikipedia.org/wiki/Maximum_flow_problem`

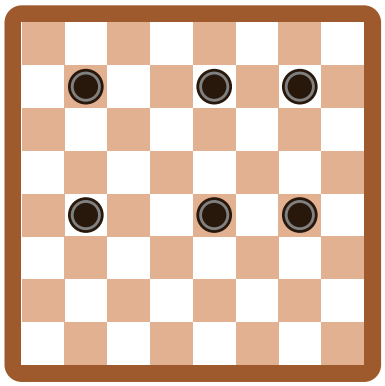
6. Zeilen und Spalten

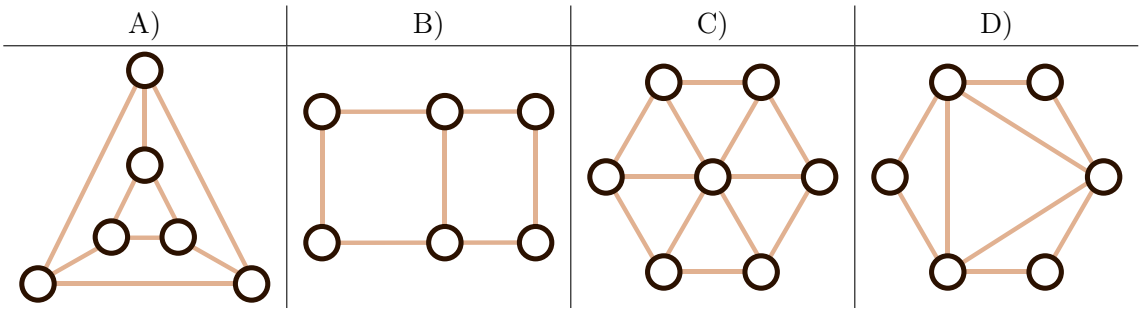
Aus den Spielsteinen auf dem Spielbrett wurde das Diagramm rechts vom Spielbrett so konstruiert, dass...

- ...jeder Spielstein durch einen Kreis dargestellt wird, und...
- ...2 Spielsteine im Diagramm durch eine Linie verbunden sind, wenn sie auf dem Brett in derselben Zeile oder in derselben Spalte liegen.



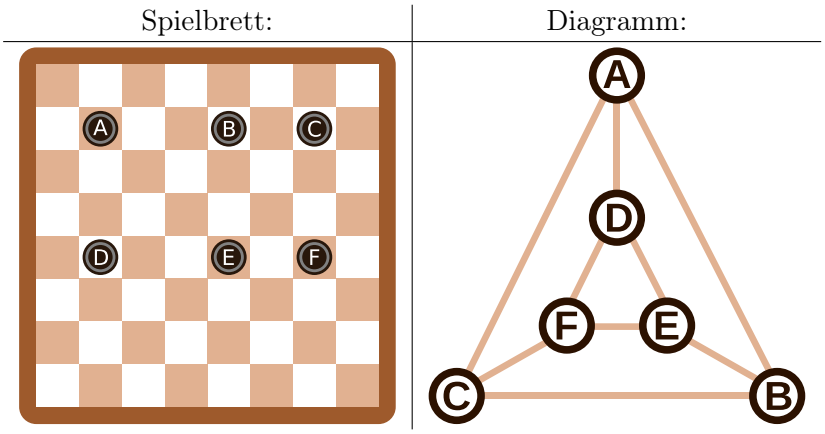
Die Spielsteine auf dem Spielbrett und die Kreise im Diagramm sind in diesem Beispiel mit Buchstaben bezeichnet, damit der Zusammenhang deutlich wird.
Welches Diagramm entspricht dem folgenden Spielbrett mit 6 Spielsteinen?



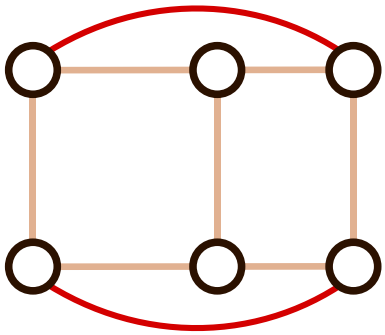


Lösung

Das Diagramm A) ist richtig. Das erkennt man in folgender Grafik, in der die Spielsteine mit Buchstaben markiert sind:



Man kann die Diagramme B), C) und D) mit folgender Beobachtung ausschliessen: Jeder Spielstein hat 2 andere Spielsteine in der selben Reihe und einen anderen Spielstein in der selben Spalte. Das heisst, dass jeder Spielstein im Diagramm mit genau $2 + 1 = 3$ anderen Spielsteinen verbunden sein muss. Das ist bei den anderen Diagrammen nicht der Fall. Zudem hat das Diagramm C) mit 7 Kreisen einen Kreis zu viel.
Das Diagramm B) ist nicht richtig, obwohl es dem Spielbrett ähnelt. Die äusseren 4 Kreise sind nur mit 2 anderen Kreisen verbunden. Um dieses Diagramm richtig zu machen, müsste man 2 zusätzliche Linien wie im folgenden Diagramm einzeichnen:



Dies ist Informatik!

In der Informatik werden solche Diagramme oft verwendet um das Wesentliche von Problemstellungen darzustellen. Solche Diagramme werden *Graphen* genannt. Die Kreise heissen *Knoten* und die Linien werden *Kanten* genannt.
Bei Graphen kommt es nur darauf an, welche Knoten mit welchen anderen Knoten durch Kanten verbunden sind. Die Anordnung der Knoten und auch die Form der Kanten spielt keine Rolle. Der selbe Graph kann daher auf unterschiedliche Weise dargestellt werden, wie wir bereits oben gesehen haben: Sowohl der Graph in Antwort A) als auch das letzte Bild in der Antworterklärung sind korrekte Lösungen und repräsentieren denselben Graph.
Graphen sind eine Form der Abstraktion. Sie repräsentieren das Wesentliche eines Problems. In unserem Fall kann man mit Hilfe des Graphen zum Beispiel das Problem „Was ist die kleinste Anzahl von Spielsteinen, die man entfernen muss, damit keine 2 Spielsteine in derselben Reihe oder derselben Spalte liegen?“ lösen. Eine wesentlicher Teil der Arbeit eines Informatikers liegt darin eine gute Repräsentation der Problemstellung zu finden, die zur Lösung des Problems hilfreich ist.

Stichwörter und Webseiten

Graph

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

7. Büchertausch

Jeder der drei Biber hat einen Tisch mit zwei Büchern. Sie wollen die Bücher durch Vertauschen benachbarter Bücher sortieren. Das machen die Biber gemeinsam in Runden. In jeder Runde darf ein Buch höchstens einmal bewegt werden.
Es gibt zwei verschiedene Typen von Runden, die immer abwechselnd durchgeführt werden:

A. Alle Biber dürfen (aber müssen nicht) die beiden Bücher auf seinem Tisch vertauschen (Beispiel A).

B. Die beiden linken Biber dürfen (aber müssen nicht) das rechte ihrer beiden Bücher mit linken Buch auf dem rechten Nachbartisch vertauschen (Beispiel B).

Die Biber beginnen mit folgender Anfangssituation:

Die erste Runde ist vom Typ A.
Wie viele Runden sind insgesamt mindestens notwendig um die Bücher zu sortieren, d.h. in die Reihenfolge 1, 2, 3, 4, 5, 6 *zu bringen?*

A) drei Runden

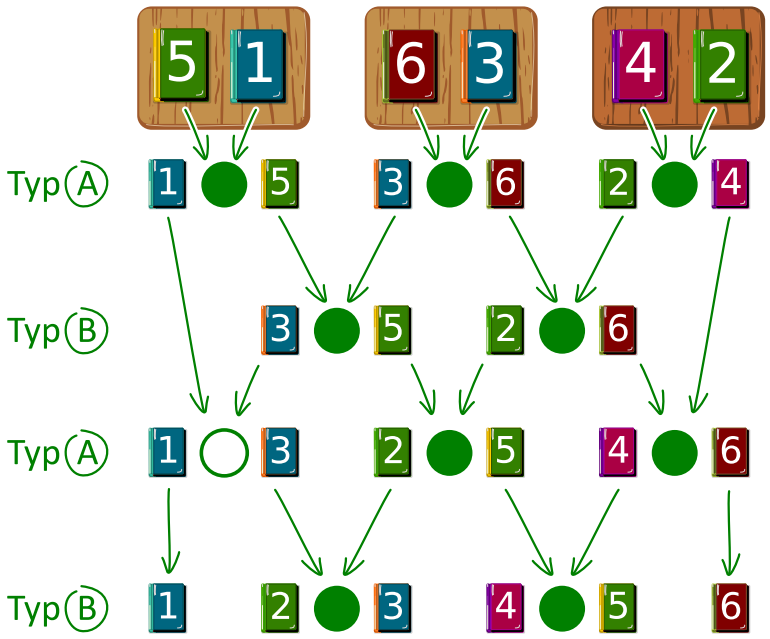
B) vier Runden

C) fünf Runden

D) sechs Runden

Lösung

Die richtige Antwort ist B).
Die Abbildung zeigt, wie die Bücher durch Platztausch sortiert werden können. Die Biber verfolgen eine „greedy-Strategie“ (“greedy” bedeutet „gierig“). Das heisst, sie versuchen bei jedem Schritt der Lösung ein Stückchen näher zu kommen. Sie vergleichen benachbarte Bücher, die gerade getauscht werden dürfen. Wenn diese beiden Bücher schon sortiert sind (das linke Buch hat eine kleinere Nummer als das rechte Buch), dann machen sie nichts. Ansonsten vertauschen sie die Bücher.
In der ersten Runde (Typ A) werden auf jedem Tisch die beiden Bücher vertauscht. In der zweiten Runde (Typ B) werden benachbarte Bücher von Nachbartischen getauscht, in der dritten Runde (Typ A) werden nur auf den beiden rechten Tischen die Bücher getauscht und in der vierten Runde (Typ B) werden alle benachbarten Bücher von Nachbartischen getauscht. Somit sind die Bücher sortiert. Schneller geht es nicht, denn das Buch 5 muss beispielsweise um vier Positionen in vier Runden nach rechts getauscht werden.



Dies ist Informatik!

Das Sortieren in dieser Aufgabe ist ein Beispiel für einen parallelen Algorithmus. Mehrere Akteure arbeiten zur gleichen Zeit an der Lösung eines Problems. Paralleles Sortieren kann durch ein Sortiernetz wie in der Abbildung dargestellt werden. Ein Sortiernetz besteht aus gerichteten Kanten, die durch Pfeile dargestellt werden, und Knoten, die durch die Kreise dargestellt werden.
In jeder Runde werden jeweils die beiden durch einen Kreis markierten Bücher verglichen und bei Bedarf vertauscht. Dabei können Vergleiche von Kreisen nebeneinander zeitgleich stattfinden. Wenn man den Pfeilen eines Buchs von oben nach unten folgt, kann man erkennen, wie es nach einigen Vertauschungen allmählich seine richtige Position in der angestrebten Reihenfolge einnimmt.

Stichwörter und Webseiten

Paralleles Sortieren, Sortiernetz

- `https://en.wikipedia.org/wiki/Sorting_network`

8. Soundex

Donald möchte Wörter nach ihrem Klang codieren. Er macht dazu Folgendes:

- Behalte den ersten Buchstaben bei.
- Streiche von allen anderen Buchstaben A, E, I, O, U, H, W und Y.
- Ersetze die restlichen Buchstaben wie folgt:
  - B, F, P oder V → 1
  - C, G, J, K, Q, S, X oder Z → 2
  - D oder T → 3
  - L → 4
  - M oder N → 5
  - R → 6
- Wenn nun zweimal oder öfters dieselbe Ziffer auftaucht, und die Buchstaben, die zu diesen Ziffern geführt haben, im Original direkt nebeneinander standen, behalte die Ziffer nur einmal. Dies gilt auch, wenn der erste Buchstabe durch diese Ziffer codiert würde, dann wird nur dieser Buchstabe behalten.
- Am Ende werden nur die ersten vier Zeichen (inkl. des ersten Buchstabens) notiert, fülle gegebenenfalls am Ende mit Nullen auf.

Die folgenden Wörter werden so codiert:

Euler → E460
Gauss → G200
Heilbronn → H416
Kant → K530
Lloyd → L300
Lissajous → L222

Welcher Code wird für das Wort „Hilbert“ erstellt?

A) H410

B) B540

C) H041

D) H416

Lösung

Der erste Buchstabe ist ein H, also ist das erste Zeichen des Codes ebenfalls ein H.
Danach werden alle A, E, I, O, U, H, W und Y gelöscht, nun muss also noch Hlbrt übersetzt werden.
Das Ersetzen der Buchstaben durch ihre Entsprechungen ergibt H4163.
Es gibt keine nebeneinanderliegenden Buchstaben mit demselben Code, also muss auch nichts gelöscht werden.
Nun werden die ersten vier Zeichen aufbewahrt, also ist H416 die richtige Antwort.

Dies ist Informatik!

Das Soundex-Verfahren, genauer gesagt der amerikanische Soundex, wurde bereits vor 100 Jahren von Robert C. Russel und Margaret King Odell entwickelt und patentiert. Es wurde dazu verwendet, ähnlich klingende Wörter in der englischen Sprache insbesondere auch ähnliche Namen von Personen zu finden. Das funktioniert, weil die Gruppen von Buchstaben, die demselben Code zugeordnet werden, ähnlich klingen: B, F, P und V sind Lippenlaute, C, G, J, K, Q, S, X und Z sind Gaumenlaute und Zischlaute, D und T sind Zahnlaute, L ist ein langer Fliesslaut, M und N sind Nasenlaute und R ist ein kurzer Fliesslaut.
Da es sehr einfach ist und nicht nur in der englischen Sprache relativ gute Resultate gibt, wird es häufig zur phonetischen Suche, also zur Suche nach ähnlich klingenden Wörtern verwendet. Es ist auch als Standard in vielen Datenbanken eingebaut.
Die Beispiele oben stammen von Donald Knuth, einem der ganz grossen Informatiker des 20. Jahrhunderts, der bis heute an seinem Buch „The Art of Computer Programming“ arbeitet. Im Band 3 „Sorting And Searching“ findet sich das beschriebene Verfahren.

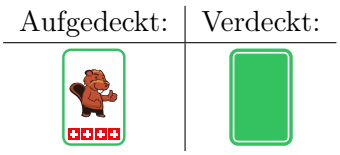
Stichwörter und Webseiten

Phonetische Suche, Soundex

- `https://www.functions-online.com/soundex.html`
- `https://de.wikipedia.org/wiki/Soundex`
- `https://de.wikipedia.org/wiki/Laut`
- `https://www-cs-faculty.stanford.edu/~knuth/taocp.html`
- `http://www.highprogrammer.com/alan/numbers/soundex.html`

9. Karten drehen

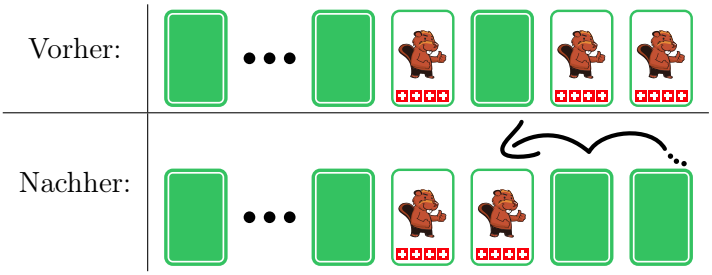
Jemand schenkt dir einen Satz gleicher Spielkarten. Die Karten sehen so aus:



Mit diesen Karten kannst du „Drehen“ spielen. Dafür legst du eine Reihe Karten vor dir aus.
In einem Spielzug gehst du diese Karten von rechts nach links so durch:

- Ist die aktuelle Karte aufgedeckt, drehe sie um.
- Ist die aktuelle Karte verdeckt, drehe sie um. Damit ist der Spielzug beendet, die übrigen Karten bleiben unverändert.

Ein Spielzug könnte zum Beispiel sein:



Die beiden rechten verdeckten Karten drehst du um. Die nächste Karte ist verdeckt. Du deckst sie auf und damit ist der Spielzug beendet.
Diesmal beginnst das Spiel mit 16 verdeckten Karten.

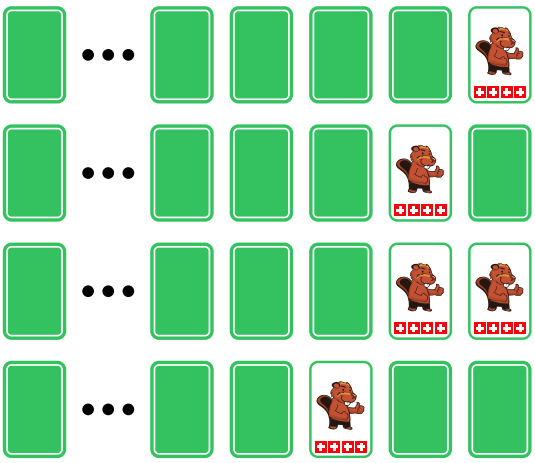
Wie viele Karten sind nach 16 Spielzügen aufgedeckt?

Lösung

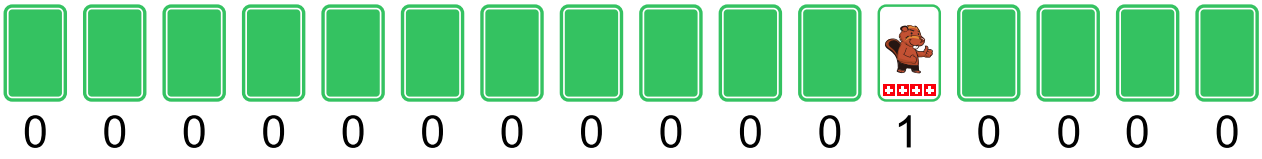
Es ist genau eine Karte aufgedeckt.
Man kann sich die Reihe von gleichen Karten, die entweder verdeckt oder aufgedeckt sein können, als Binärzahl vorstellen. Binärzahlen bestehen nur aus den Ziffern 0 und 1. Eine verdeckte Karte stellt beispielsweise die Ziffer 0 dar und eine aufgedeckte Karte die Ziffer 1.
Analog zum üblichen Zehnersystem gibt jede Stelle einer Binärzahl an, ob die passende Zweierpotenz in den Wert der Zahl einzurechnen ist oder nicht. Ist z. B. die dritte Stelle (von rechts) einer Binärzahl mit einer 1 besetzt, ist die dritte Zweierpotenz zum Wert der Zahl zu addieren – also 2^2, denn $1 = 2^0$.
Binärzahlen werden auf diese Weise um 1 hochgezählt. Man beginnt mit der Ziffer ganz rechts:

- Ist die aktuelle Ziffer eine 0, mache daraus eine 1. Damit ist die gesamte Zahl um 1 hochgezählt.
- Ist die aktuelle Ziffer eine 1, mache daraus eine 0 und gehe zur nächsten Ziffer nach links für den Übertrag.

Das entspricht genau einem Spielzug im Spiel „Drehen“. Ein Spielzug erhöht also den Wert der Binärzahl, die durch die Kartenreihe dargestellt wird, um 1. Die verdeckten Karten zu Beginn stellen die Binärzahl dar, die nur aus Nullen besteht, also den Wert 0 hat.
Das folgende Bild zeigt die Ergebnisse der ersten vier Spielzüge, die also den Zahlen von 1 bis 4 entsprechen. Man kann sehen, dass bei den Zahlen 1, 2 und 4 (also den Zweierpotenzen 2^0, 2^1 und 2^2) genau eine Karte aufgedeckt ist. Das heisst, dass bei einer Binärzahl, die eine Zweierpotenz als Wert hat, nur an der Stelle, die dieser Zweierpotenz entspricht, eine 1 ist.



Nach 16 Spielzügen erhalten wir also die Darstellung einer Binärzahl mit dem Wert 16. Da $16 = 2^4$ ist, hat diese Binärzahl an der fünften Stelle von rechts eine 1 und sonst nur Nullen: 0000000000010000.
In der Darstellung mit den Karten ist also genau diese eine Karte aufgedeckt:



Dies ist Informatik!

Die kleinste Speichereinheit heutiger Computer kann nur zwei Werte unterscheiden: AN oder AUS, WAHR oder FALSCH, 0 oder 1. Alle Daten, die in einem Computer gespeichert und verarbeitet

werden, müssen wir als Reihen binärer Ziffern sehen, letztlich also als Binärzahlen. Deshalb haben Operationen auf Binärzahlen für die Informatik eine grosse Bedeutung.
Seitdem es Computer gibt, wird das Abarbeiten solche Operationen möglichst effizient gebaut. Es gibt Operationen, die zwei Binärzahlen miteinander verknüpfen, wie etwa die Rechenoperationen Addition oder Multiplikation. Es gibt aber auch Operationen, die eine einzelne Binärzahl verändern, etwa das Verschieben aller Ziffern um eine Position nach links oder eben das Hochzählen, also die Addition um 1 wie in dieser Aufgabe.
Gute Prozessoren zeichnen sich dadurch aus, dass sie solche Operationen schnell und energiesparend ausführen, und zwar millionenfach pro Sekunde. Es ist dann die Aufgabe des Programmierers und seiner Werkzeuge, komplizierte Abläufe, auch diese einfachen Operationen, zu reduzieren, so dass der Benutzer beliebige Programme verwenden kann.

Stichwörter und Webseiten

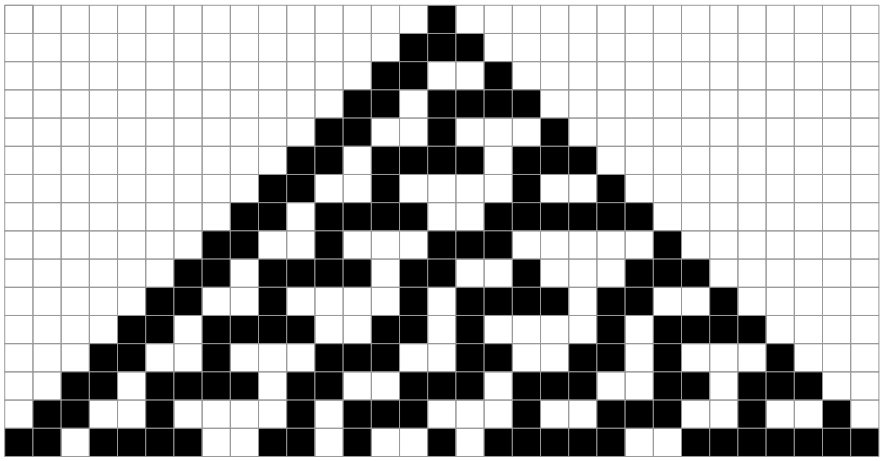
Binärzahlen

- `https://de.wikipedia.org/wiki/Dualsystem`
- `https://de.wikipedia.org/wiki/Synchronzähler`
- `https://de.wikipedia.org/wiki/Asynchronzähler`

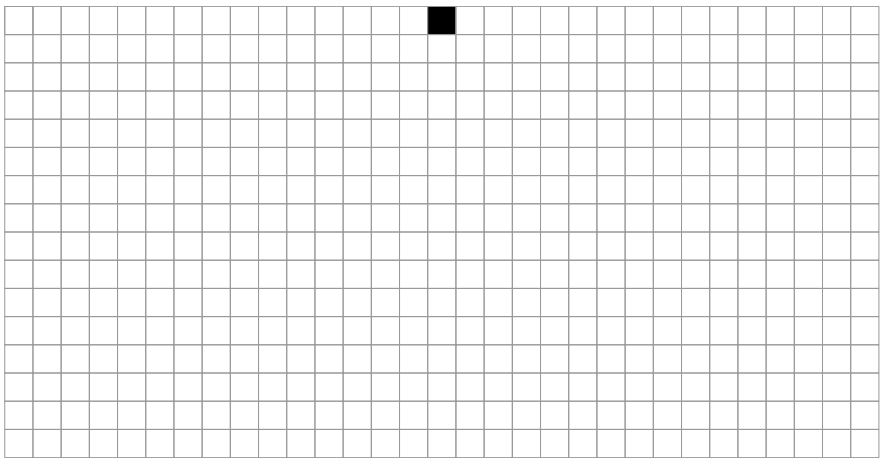
10. Plättlimuster

Tina soll Plättli auf eine Fläche legen, die 31 Plättli breit und 16 Plättli hoch ist. Tina möchte, dass die Plättli nach einem Satz einfacher Regeln gelegt werden.
Eine einfache Regel ist immer so aufgebaut, dass drei Plättli nebeneinander bestimmen, wie das Plättli mittig darunter aussehen soll. Ein Satz einfacher Regeln besteht aus acht einfachen Regeln: für jede mögliche Kombination von Plättli eine einfache Regel (am Rand werden einfach weisse Plättli angenommen):

Tina startet oben in der Mitte mit einem schwarzen Plättli, alle anderen Plättli der ersten Reihe sind weiss. Wenn sie ihre Regeln anwendet, sieht die Fläche so aus:



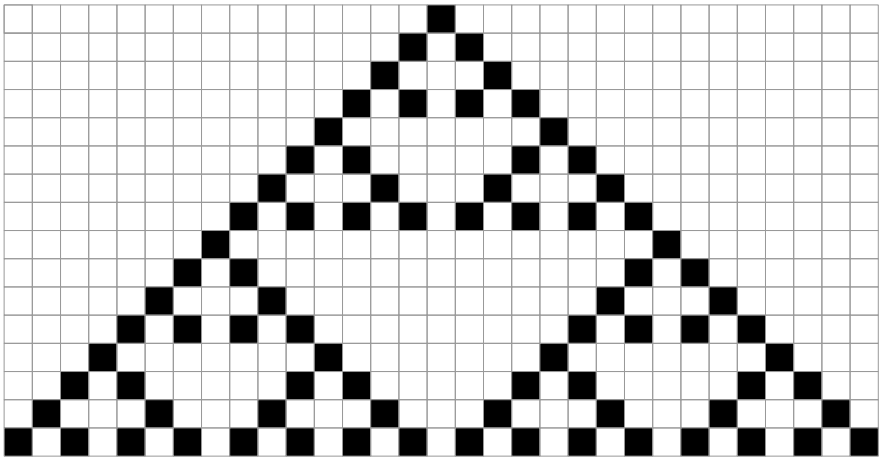
Erstelle einen eigenen Satz einfacher Regeln, so dass in der letzten Reihe immer abwechselnd ein schwarzes und ein weisses Plättli liegt.

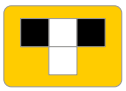


Lösung

Für diese Aufgabe gibt es viele Lösungen. Eine mögliche Lösung ist die folgende:

Sie erzeugt das folgende Muster:



Wenn man sich das Muster genau anschaut, sieht man, dass es eigentlich neun „Dreiecke“ sind: die dritte erzeugte Linie besteht aus abwechselnd schwarzen und weissen Plättli, so dass immer die Regel  angewendet wird ausser am linken und rechten Rand der Gesamtfigur, von wo aus sich neue Dreiecke bilden können.
Wenn man das Ergebnis einer Regel als W für ein weisses Plättli und S für ein schwarzes Plättli abkürzt, kann man jede der 256 möglichen Sätze von Regeln (W oder S für jedes der 8 einzelnen Regeln, also $2^8 = 256$) mit einem Code beschreiben. Das Beispiel in der Aufgabe hätte dann den Code WWWSSSSW.
Die folgenden siebzehn Codes erzeugen in der 16. Zeile ein schwarz-weiss-Muster der Plättli:

SWSSWWSS
SSSSSWSW
WSSSSWSW
SWSSSWSW
WWSSSWSW
SSWSSWSW
WSWSSWSW
SWWSSWSW
WWWSSWSW
SSSSWWSW
WSSSWWSW
SWSSWWSW
WWSSWWSW
SSWSWWSW
WSWSWWSW
SWWSWWSW
WWWSWWSW (die Beispiellösung)

Dies ist Informatik!

Diese Regeln in dieser Aufgabe ähneln Conways Spiel des Lebens (engl. *Conway's Game of Life*), das der englische Mathematiker John Horton Conway im Jahre 1970 veröffentlicht hat. Es basiert auf einem zweidimensionalen zellulären Automaten. Ein zweidimensionaler zellulärer Automat sieht aus wie ein Boden aus Plättli. Jedes Plättli ist eine „Zelle", deren Zustand von den acht Nachbar„zellen" abhängt. Der neue Zustand jeder Zelle wird nach einfachen Regeln aus den (alten) Zuständen der Nachbarzellen berechnet. So kann die z.B. Vermehrung und das Absterben von Lebewesen in einem Gebiet simuliert werden. In diesem Fall ist es ein reduzierter Regelsatz, da eine „Zelle" nur von den drei „Zellen" darüber abhängt.

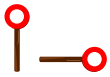
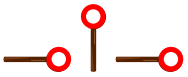
Stichwörter und Webseiten

Muster, Zellulärer Automat

- `http://web.stanford.edu/~cdebs/GameOfLife/`
- `https://en.wikipedia.org/wiki/Rule_90`
- `https://de.wikipedia.org/wiki/Zellulärer_Automat`
- `https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens`

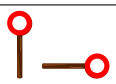
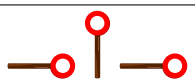
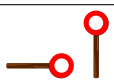
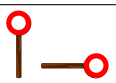
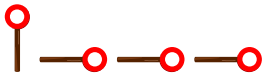
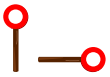
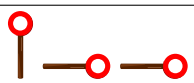
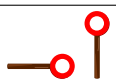
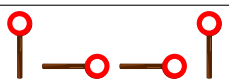
11. Wo ist das Segelflugzeug?

Jana und Robin spielen mit ihrem Segelflugzeug. Einer von ihnen lässt es von einem kleinen Hügel fliegen und der andere holt es nach jeder Landung ab. Leider wurde das Gras der Wiese schon länger nicht mehr gemäht. So sieht man das gelandete Segelflugzeug nur noch vom Hügel und nicht mehr von der Wiese aus. Jana und Robin müssen sich daher gut verständigen können. Dazu haben Sie einen Signalcode vereinbart.

links	rechts	in Richtung Hügel	in Richtung Tal
			

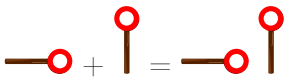
Leider gibt es ein Problem mit diesem Signalcode. Wenn man zum Beispiel die folgenden Befehle sendet, ...

... kann dies „links – in Richtung Hügel – links“ bedeuten, aber auch „links – rechts – links – links“. *Jana und Robin haben sich vier neue Signalcodes überlegt. Welchen Signalcode können Sie widerspruchsfrei nutzen?*

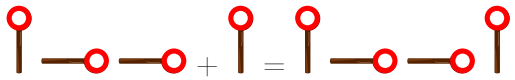
	links	rechts	in Richtung Hügel	in Richtung Tal
A)				
B)				
C)				
D)				

Lösung

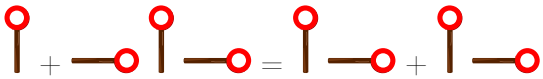
Die richtige Antwort ist C). Man kann das beispielsweise daran erkennen, dass jeder Befehl mit  anfängt und danach 0, 1, 2 oder 3 mal  folgt und  nicht wieder vorkommt. Somit weiss man, dass bei jedem  ein neuer Befehl startet und man muss nur zählen, wie häufig  folgt.
Die Antwort B) ist kein guter Signalcode, da „links“ gefolgt von „rechts“ die gleichen Signale verwendet wie „in Richtung Hügel“:



Die Antwort D) ist kein guter Signalcode, da „links“ gefolgt von „in Richtung Tal“ dasselbe bedeutet wir „in Richtung Hügel“:

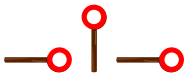


Bei der Antwort A) ist es etwas schwieriger. Wenn man „in Richtung Tal“ und dann „links“ signalisiert, bedeutet das dasselbe wie zweimal „rechts“:



Dies ist Informatik!

Wenn Computer Daten über ein Kabel oder über Funk senden, so tun sie dies durch schnelle Signalfolgen, wobei für jedes Einzelsignal zwei Möglichkeiten bestehen. Man kann sich das als Strom ein oder aus vorstellen (oft sind die Dinge natürlich auch etwas komplexer). Genau dies machen Jana und Robin. Auch sie verwenden zwei Einzelsignale für Ihre Nachrichten.
Diese Art und Weise, in der Nachrichten oder Befehle in Signale umgesetzt werden, wird als (binärer) Code bezeichnet. In diesem Fall ein Code mit variabler Länge, weil die Anzahl der für eine Nachricht oder einen Befehl verwendeten Signale nicht immer gleich sein muss.
Es ist wichtig, dass der Empfänger einer codierten Nachricht die Signale zurück in die ursprüngliche Nachricht übersetzen kann, ohne dass er einen Fehler macht. Mit andere Worten, du musst vorsichtig sein, wenn du einen solchen Code entwirfst. Codes, die wir als „gut“ bezeichnen, werden als eindeutig decodierbare Codes bezeichnet.
Eine spezielle Art von eindeutig decodierbaren Codes ist der Präfix-Code. Dies ist ein Code, bei dem keines der gültigen Codewörter mit der vollständigen Folge von Signalen eines anderen Codeworts beginnt, zum Beispiel:

links	rechts	in Richtung Hügel	in Richtung Tal
			

Präfix-Codes haben schöne Eigenschaften: Sie können ziemlich kurz gehalten werden und sie sind leicht zu entschlüsseln. Sie werden nicht nur für Kommunikationszwecke verwendet, sondern kommen auch in mehreren Komprimierungsalgorithmen.

Stichwörter und Webseiten

Code, Präfix-Code, Signalscheibe

- `https://de.wikipedia.org/wiki/Präfixcode`
- `https://de.wikipedia.org/wiki/Optische_Telegrafie`

12. Probenplan

Fünf Tänzer proben für einen Auftritt: Alex, Bojan, Coco, Deniz und Emil.
Beim Auftritt bilden die Tänzer nacheinander diese Paare:

- Alex – Bojan
- Coco – Alex
- Emil – Deniz
- Alex – Emil
- Coco – Deniz
- Bojan – Coco
- Deniz – Alex
- Coco – Emil

Morgen sollen die Paare nacheinander proben. Dabei soll der Zeitplan so erstellt werden, dass immer ein Mitglied eines Paares zum nächsten Paar gehört und direkt weitermachen kann. Da es sonst zu ermüdend wäre, soll jedoch kein Tänzer dreimal nacheinander proben. Zum Beispiel probt nach dem Paar Alex – Bojan ein anderes Paar, zu dem entweder Alex oder Bojan gehört, also Coco – Alex, Alex – Emil, Bojan – Coco oder Deniz – Alex.
Einer der Tänzer stellt fest, dass er auf jeden Fall später zur Probe kommen kann: Er wird auf keinen Fall zum ersten Paar auf dem Zeitplan gehören.
Welcher Tänzer ist das?

A) Alex

B) Bojan

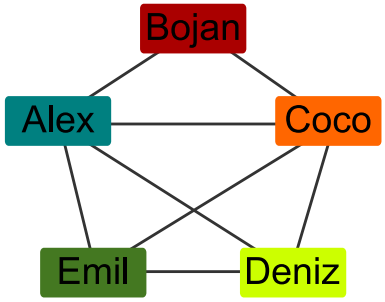
C) Coco

D) Deniz

E) Emil

Lösung

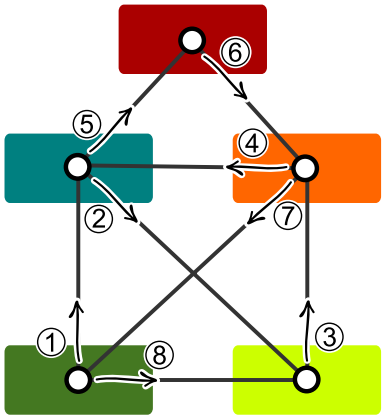
B) Bojan ist die richtige Antwort.
Im folgenden Diagramm ist für jeden Tänzer ein Viereck mit seinem Namen zu sehen. Ausserdem sind zwei Vierecke mit einer Linie verbunden, wenn die beiden Tänzer ein Paar bilden. Ein gültiger Probenplan, ist dann ein „Weg“ durch das Diagramm: Dieser beginnt bei einem Viereck und geht genau einmal an jeder Linie entlang. Ein direktes Zurück gibt es nicht, denn sonst würde ein Tänzer ja dreimal nacheinander proben.



Dabei muss der Weg jedes Viereck, zu dem er zwischendurch kommt, auch wieder verlassen. Bei solchen Vierecken gehen eine gerade Anzahl an Linien aus (die Tänzer sind also in einer geraden Anzahl von Paaren beteiligt). Die Vierecke, bei denen der Weg beginnt bzw. endet, haben eine ungerade Anzahl Linien aus (die Tänzer sind also an einer ungeraden Anzahl von Paaren beteiligt). Von den Vierecken von Deniz und Emil gehen eine ungerade Anzahl Linien aus (nämlich jeweils drei Linien). Nur diese beiden können also zum ersten oder letzten Paar gehören. Bojan ist der einzige Tänzer, der weder mit Deniz noch mit Emil ein Paar bildet. Damit kann nur er auf keinen Fall zum ersten Paar des Probenplans gehören.

Dies ist Informatik!

Das Bild oben zeigt, wie die Tanzpaare als *Graph* dargestellt werden. Ein Graph besteht aus *Knoten* (hier: die Tänzer) und *Kanten* (hier: die Paar-Bildungen der Tänzer). Er ist eine sehr vielseitige Struktur, die bei vielen Informatik-Aufgabenstellungen zur Modellierung verwendet wird, z. B. bei Verkehrs- oder Kommunikationsnetzen. In dieser Aufgabe bilden die Tänzer ein Paar-Netzwerk.
In vielen Netzwerken kann es nötig sein, auf einem Weg von einem Start- zu einem (unterschiedlichen) Zielknoten alle Verbindungen abzugehen. Dabei stellt sich die Frage, z. B. aus Gründen der Effizienz, ob es möglich ist, einen solchen Weg so zu gestalten, dass jede Verbindung nur einmal begangen wird. Ein solcher Weg, der jede Kante genau einmal durchläuft, wird zu Ehren von Leonhard Euler, dem Erfinder der Graphentheorie, als Eulerweg bezeichnet. Euler hatte bewiesen, dass in einem zusammenhängenden Graphen ein Eulerweg existiert, wenn genau zwei Knoten eine ungerade Anzahl Verbindungen mit anderen Knoten haben (und alle anderen Knoten demnach eine gerade Anzahl von Verbindungen haben). Nur diese beiden Knoten können Anfangs- oder Endpunkt des Eulerwegs sein.



Das Paar-Netzwerk dieser Biberaufgabe hast Du übrigens wahrscheinlich schon einmal gesehen. Wenn man es ein wenig dreht und die Knoten verkleinert, erkennt man das *Haus des Nikolaus*. Wer das Haus vom Nikolaus in einem Zug zeichnet, geht also einen Eulerweg entlang der Linien des Hauses.

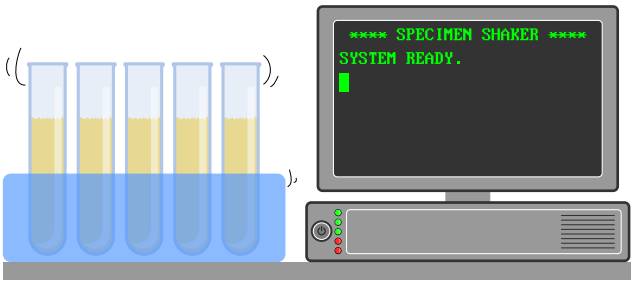
Stichwörter und Webseiten

Graph, Eulerweg

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

13. Labor

In einem medizinischen Labor muss eine Probe eines Patienten regelmässig geschüttelt werden. Dafür benutzt das medizinische Labor eine Maschine, die ein Programm ausführt. Das Programm wird Zeile für Zeile ausgeführt.
Die Maschine wird mit dem folgenden Programm programmiert:




```
1 SPEICHERE 0 ALS N
2 ERHÖHE N UM 1
3 GEHE ZU ZEILE 6
4 WENN N GLEICH 60 IST, DANN GEHE ZU ZEILE 8
5 SPEICHERE 0 ALS N
6 ERHÖHE N UM 1
7 GEHE ZU ZEILE 2
8 WIEDERHOLE N MAL SCHÜTTLE
9 ENDE
```

Die Befehle sind:

- `SPEICHERE` *`Zahl`* `ALS` *`Name`*: Speichert die Zahl *`Zahl`* unter dem Namen *`Name`*.
- `ERHÖHE` *`Name`* `UM 1`: Liest die unter *`Name`* gespeicherte Zahl, addiert 1 und speichert die erhöhte Zahl unter *`Name`*.
- `GEHE ZU ZEILE` *`Zeilennummer`*: Führt das Programm ab der Zeile *`Zeilennummer`* weiter.
- `WENN` *`Name`* `GLEICH` *`Zahl`* `IST, DANN` *`Befehl`*: Vergleicht die Zahl, die unter *`Name`* gespeichert ist, mit der Zahl *`Zahl`*. Wenn beide gleich sind, führt es den Befehl *`Befehl`* aus, sonst nicht.
- `WIEDERHOLE` *`Name`* `MAL` *`Befehl`*: Führt den Befehl *`Befehl`* so häufig aus wie die Zahl, die unter *`Name`* gespeichert ist.
- `SCHÜTTLE`: Schüttelt die Probe einmal.
- `ENDE`: Beendet das Programm.

Wie häufig wird die Maschine die Probe schütteln?

A) Die Probe wird nie geschüttelt.

B) Die Probe wird einmal geschüttelt.

C) Die Probe wird 60 mal geschüttelt.

D) Die Maschine wird nicht aufhören, die Probe zu schütteln.

Lösung

Das Programm springt immer wieder von der Zeile 3 zur Zeile 6 und von der Zeile 7 zur Zeile 2. Ausser ganz am Anfang die Zeile 1 werden also nur die Zeilen 2, 3, 6 und 7 ausgeführt. Die Probe würde in der Zeile 8 geschüttelt werden, die aber nie ausgeführt wird. Das heisst, dass das Programm die Probe letztlich nie schütteln wird. Da auch die Zeile 9 nie ausgeführt wird, beendet die Maschine das Programm auch nie.

Dies ist Informatik!

Das Programm benutzt als Kontrollstruktur den Befehl „`GEHE ZU ZEILE`“ (engl. “`GO TO`”), um zu anderen Programmteilen zu springen. Diese Kontrollstruktur ist sehr hardwarenah und wurde in frühen Programmiersprachen (bis in die 1980er-Jahre) häufig geschrieben, um in Abhängigkeit von Benutzereingaben oder anderen Bedingungen zu reagieren. Es ist jedoch manchmal nicht einfach, als Mensch einen solchen Programmcode zu lesen und zu verstehen, häufig passierten so Fehler. Moderne Programmiersprachen, wie sie ab den 1950er-Jahren entwickelt wurden und sich langsam durchsetzten, benutzen deshalb diese Kontrollstruktur nicht mehr. Anstelle dessen benutzt man Schleifen (wie das `WIEDERHOLE` im Beispiel), Verzweigungen mit Programmblöcken oder Unterprogramme.

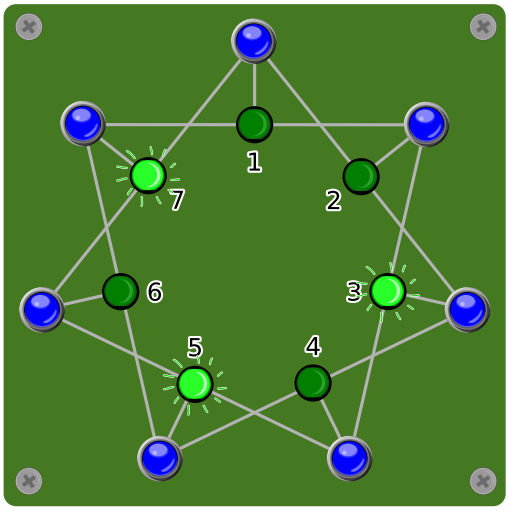
Stichwörter und Webseiten

Kontrollstruktur, Programmanalyse, `GOTO`

- `https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf`
- `https://de.wikipedia.org/wiki/Sprunganweisung`

14. Licht an!

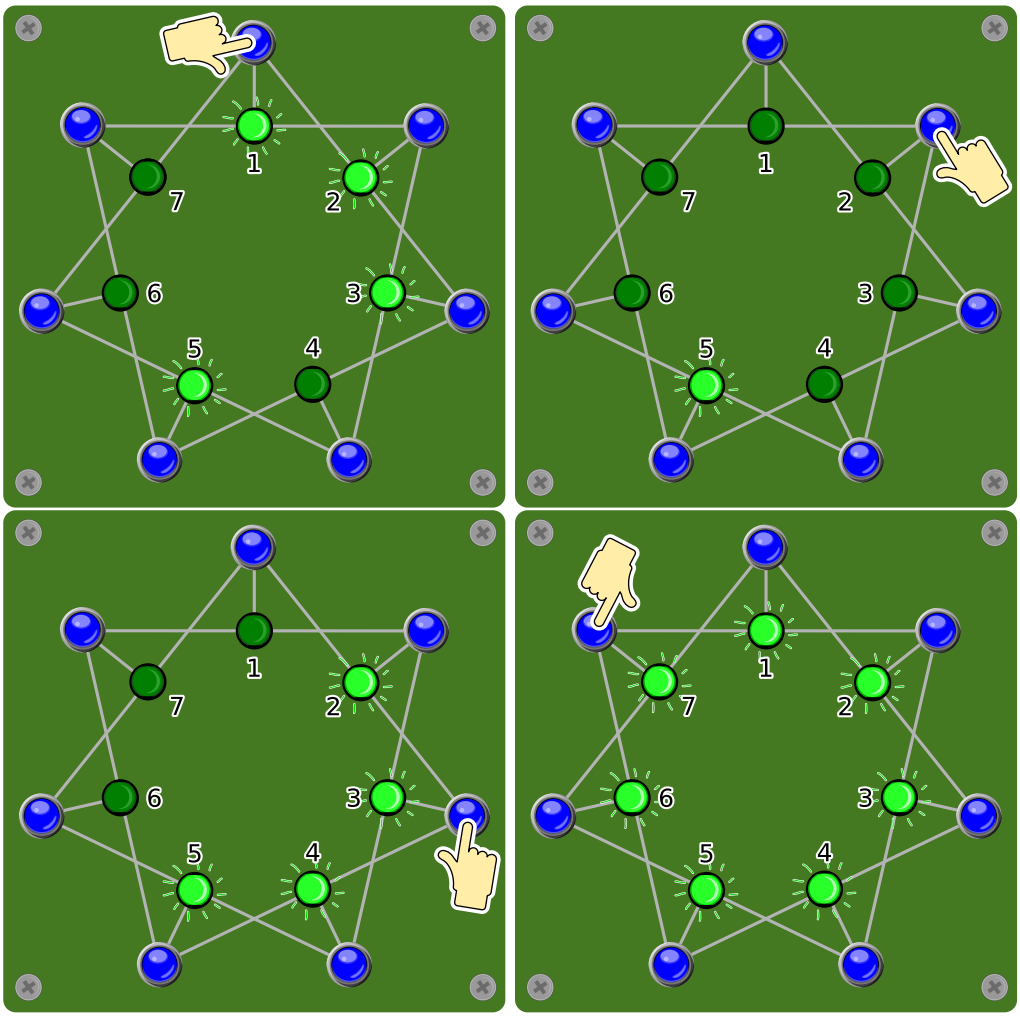
Sieben Schalter sind mit sieben Lampen verbunden und zwar so, dass ein Schalter immer drei Lampen kontrolliert. Wird ein Schalter gedrückt, werden eine von ihm kontrollierte Lampe, die eingeschaltet war, ausgeschaltet und eine von ihm kontrollierte Lampe, die ausgeschaltet war, eingeschaltet.



Welche Schalter muss man drücken, dass am Ende alle Lampen leuchten?

Lösung

Wenn man den Schalter bei der Lampe 1 (oder bei der Lampe 2) drückt, werden die beiden Lampen 1 und 2 eingeschaltet und die danebenliegende Lampe 7 (oder 3) ausgeschaltet. Damit sind genau drei nebeneinanderliegende Lampen eingeschaltet. Indem man auf den Schalter bei der mittleren dieser Lampen drückt (Schalter 2 oder 1, je nachdem was man vorher gedrückt hat), werden diese drei ausgeschaltet. Damit sind alle Lampen ausser der Lampe 5 ausgeschaltet. Da dies sechs Lampen sind, können diese mit zwei Schaltern jeweils in der Mitte eines der beiden Dreierblöcke (Schalter 3 und 7) einschalten. Damit sind alle Lampen eingeschaltet.



Jede mögliche Reihenfolge dieser Schalter (1, 2, 3, 7) führt zum richtigen Resultat. Das zweimalige Drücken desselben Schalters macht das Resultat des ersten Drückens rückgängig. Damit stellt sich letztlich nur die Frage, welcher Schalter gedrückt werden muss und welcher nicht. Bei sieben Schaltern gibt es immerhin noch $2^7 - 1 = 127$ verschiedene Möglichkeiten, die Schalter zu drücken oder nicht (keinen Schalter zu drücken ist offensichtlich falsch).

Dies ist Informatik!

Diese Aufgabe besteht darin, ein System in einem bestimmten bekannten Zustand (Lampen 3, 5 und 7 sind eingeschalten) in einen anderen bekannten Zustand (Lampen 1, 2, 3, 4, 5, 6 und 7 sind

eingeschaltet) zu überführen. Dabei müssen bestimmte Regeln beachtet werden. Das Finden des Weges steht im Vordergrund.
In der Informatik stellen sich solche Fragen öfters. Naiv könnte man natürlich anfangen, einfach alle Kombinationen durchzuprobieren (was zu den oben erwähnten 127 verschiedenen Möglichkeiten führt). Will man schneller auf das Ergebnis kommen, lohnt es sich, das Problem von zwei Seiten her anzugehen: zum einen aus Richtung des Startzustandes und parallel dazu aus Richtung des Endzustandes. Je grösser das System ist, desto mehr Zeit kann man mit dieser Methode der bidirektionalen Suche sparen.

Stichwörter und Webseiten

Bidirektionale Suche, Endlicher Automat

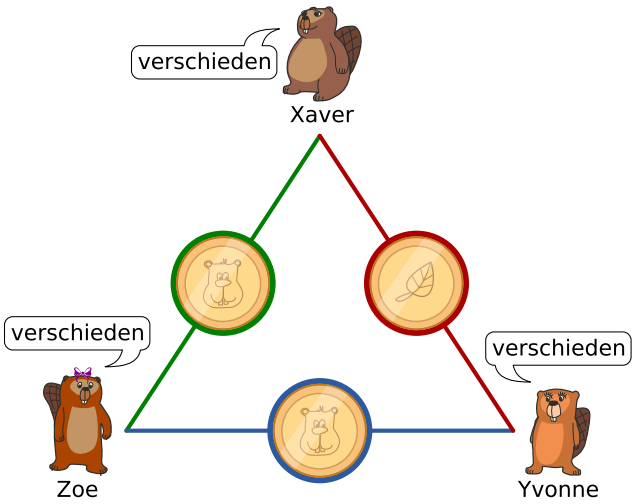
- `https://de.wikipedia.org/wiki/Bidirektionale_Suche`

15. Streng geheim

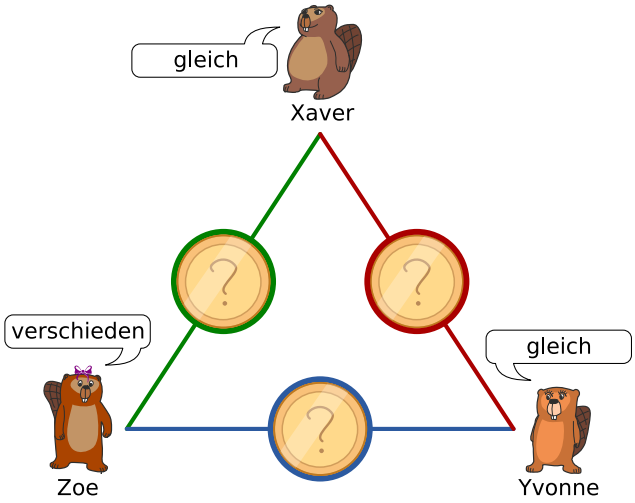
Xaver, Yvonne und Zoe spielen gelegentlich bei einer Tombola mit. Jetzt wurde bekannt, dass der einzige Haupttreffer von jemandem aus ihrer Stadt gewonnen wurde. Gerne würden sie wissen, ob einer von ihnen zufällig den Hauptgewinn gewonnen hat, aber andererseits sollte geheim bleiben, wer es ist. So gehen sie vor:

1. Xaver und Yvonne werfen eine Münze, nur sie kennen das Ergebnis.
2. Xaver und Zoe werfen eine Münze, nur sie kennen das Ergebnis.
3. Yvonne und Zoe werfen eine Münze, nur sie kennen das Ergebnis.
4. Jeder von ihnen gibt danach bekannt, ob ihre beiden jeweiligen Münzwürfe „gleich“ oder „verschieden“ ausgegangen sind.
   - Jemand, der nicht den Hauptgewinn gewonnen hat, soll wahrheitsgemäss antworten.
   - Jemand, der den Hauptgewinn gewonnen hat, soll den Wahrheitswert seiner Aussage umdrehen (also „gleich“ sagen, auch wenn seine beiden Ergebnisse verschieden waren und umgekehrt).

Unten ein Beispiel mit erfolgten Münzwürfen und mit der Annahme, dass Zoe den Hauptpreis gewonnen hat.



Betrachte folgende Situation, bei der die Münzwürfe erfolgen, dir aber nicht bekannt sind.



Welche der folgenden Aussagen ist wahr?

A) Keiner von den drei Freunden hat den Hauptgewinn gewonnen.

B) Einer von den drei Freunden hat den Hauptgewinn gewonnen, aber wir wissen nicht wer.

C) Einer von den drei Freunden hat den Hauptgewinn gewonnen, und wir wissen genau wer.

D) Wir wissen nicht, ob jemand den Hauptgewinn gewonnen hat.

Lösung

Die richtige Antwort ist B). Einer von den drei Freunden hat den Hauptgewinn gewonnen, aber wir wissen nicht wer.
Für diese Frage gibt es verschiedene Lösungsansätze. Einer ist: Bei drei Münzwürfen gibt es genau diese zwei Möglichkeiten:

- Alle drei Münzen zeigen das gleiche Ergebnis.
- Eine Münze zeigt ein anderes Ergebnis als die beiden anderen.

Unter der Annahme, dass niemand den Hauptgewinn gewonnen hat, gilt folgendes:

- Sind alle Münzen gleich, dann sagen alle drei Freunde „gleich“.
- Zeigt eine Münze ein anderes Ergebnis, dann sagt einer der Freunde „gleich“ und die beiden anderen sagen „verschieden“.

Da aber zwei Freunde „gleich“ sagen und einer „verschieden“, muss einer von ihnen lügen. Also hat einer den Hauptpreis gewonnen.
Wir können aber nicht sagen, wer. Denn wenn einer der beiden Freunde, die „gleich“ sagen, eigentlich „verschieden“ hätten sagen müssen, wenn sie die Wahrheit sagen würden, wissen wir nicht, welcher der beiden lügt. Und auch der dritte könnte lügen: dann würden alle eigentlich „gleich“ sagen müssen, was auch sein könnte.

Dies ist Informatik!

Die von Xaver, Yvonne und Zoe gewählte Vorgangsweise wurde als sogenanntes *Dining Cryptographers Problem* zuerst beschrieben.
Diese Methode ist für Informatiker deshalb besonders interessant, weil sowohl der Sender als auch der Empfänger der Nachricht anonym bleiben: Wenn das Verfahren richtig ausgeführt wird, wissen am Ende alle verlässlich, ob einer von ihnen der Gewinner ist. Ausser dem Gewinner selbst weiss aber keiner, von wem die Information kommt. Auch der Empfänger der Information bleibt anonym, weil alle Beteiligten die Information erhalten haben.

Stichwörter und Webseiten

Anonymität, Dining Cryptographers Problem

- `https://en.wikipedia.org/wiki/Dining_cryptographers_problem`

A. Aufgabenautoren

Andrea Adamoli
Jared Asuncion
Wilfried Baumann
Daphne Blokhuis
William Chan
Kessarapan Charoensueksa
Anton Chukhnov
Kris Coolsaet
Valentina Dagienė
Christian Datzko
Susanne Datzko
Marissa Engels
Hanspeter Erni
Veerle Fack
Gerald Futschek
Ionuț Gorgos
Shuchi Grover
Martin Guggisberg
Urs Hauser
Juraj Hromkovič
Svetlana Jakšić
Zhang Jinbao
Emil Kelevedjiev
Dong Yoon Kim
Vaidotas Kinčius
Iryna Kirynovich
Regula Lacher
Judith Lin
Violetta Lonati
Nils Mak
Dimitris Mavrovouniotis
Mattia Monga
Anna Morpurgo
Tom Naughton
Henry Ong
Wolfgang Pohl
Stavroula Prantsoudi
Nol Premasathian
J.P. Pretti
Doris Reck
Chris Roffey
Kirsten Schlüter
Andrea Maria Schmid
Victor Schmidt
Andrea Schrijvers
Eljakim Schrijvers
Vipul Shah
Jacqueline Staub
Allira Storey
Ahto Truu
Willem van der Vegt
Jiří Vaníček
Troy Vasiga
Rechilda Villame
Eslam Wageed
Pieter Waker
Michael Weigend

B. Sponsoring: Wettbewerb 2018

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

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

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

http://www.verkehrshaus.ch/

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

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

UBS

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

bbv Software Services

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

PRESENTEX
Das Geschenk – die gute Werbung

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

Bits with Bite
ZUBLER & PARTNER AG
Informatik

`http://www.zubler.ch/`
Zubler & Partner AG Informatik
Umfassendes Angebot an Dienstleistungen.

OXOCARD

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

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

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

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

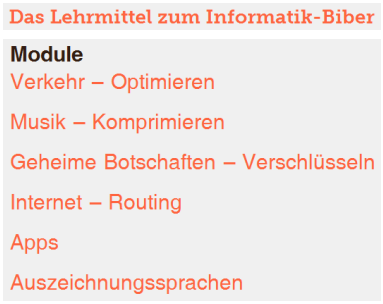
http://www.hepl.ch/
Haute école pédagogique du canton de Vaud

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

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

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

C. Weiterführende Angebote



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

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

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

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