



**INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA**

Aufgaben und Lösungen 2023

Schuljahre 3/4

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

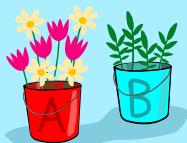
Herausgeber:

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

010100110101011001001001
01000010010110101010011
010100110100100101000101
001011010101001101010011
01001001010010010010001

SV!A

www.svia-ssie-ssii.ch
schweizerischerverein für informatik in
der ausbildung // société suisse pour l'infor
matique dans l'enseignement // società sviz
zera 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,
Ausbildungs- und Beratungszentrum für Informatikunterricht

Tobias Berner: Pädagogische Hochschule Zürich

Christian Datzko: Wirtschaftsgymnasium 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 Staatsministerium 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,
Ausbildungs- 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 Nöpfer, Kai Zürcher

Für die Übersetzung der Finalaufgaben ins Französische:

Jan Schönbacher: 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.



Die Aufgaben sind lizenziert unter einer Creative Commons Namensnennung – Nicht-kommerziell – Weitergabe unter gleichen Bedingungen 4.0 International Lizenz. Die Autoren sind auf S. 50 genannt.



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.bebas.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/>



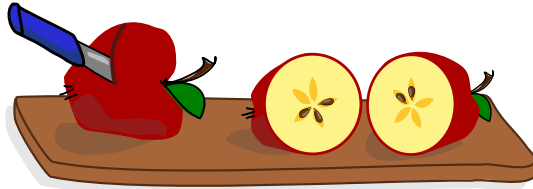
Inhaltsverzeichnis

Mitarbeit Informatik-Biber 2023	i
Vorwort	iii
Inhaltsverzeichnis	v
1. Äpfel halbieren	1
2. Wasser – Land	5
3. Neue Hüte	9
4. Spass im Zoo	13
5. Annas Regenschirm	17
6. Blumenstrauss	21
7. Besonderer Baum	25
8. Karlas Traumhaus	29
9. Rüepli pflanzen	31
10. Riccas	35
11. Gemüsebeet	39
12. Brunnen	43
13. Ogham	47
A. Aufgabenautoren	50
B. Akademische Partner	51
C. Sponsoring	52
D. Weiterführende Angebote	53



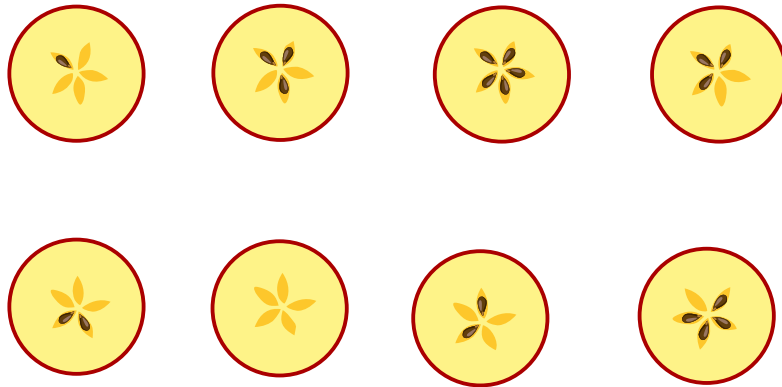
1. Äpfel halbieren

Äpfel kann man in eine obere und untere Hälfte teilen. Einige Apfelkerne bleiben in der oberen Hälfte, die anderen in der unteren Hälfte. An den Löchern und Kernen sieht man, dass die Hälften zusammen passen:



Das macht man in Tschechien zu Weihnachten. Gala halbiert vier Äpfel. Sie legt die oberen Hälften und die unteren Hälften in zwei Reihen.

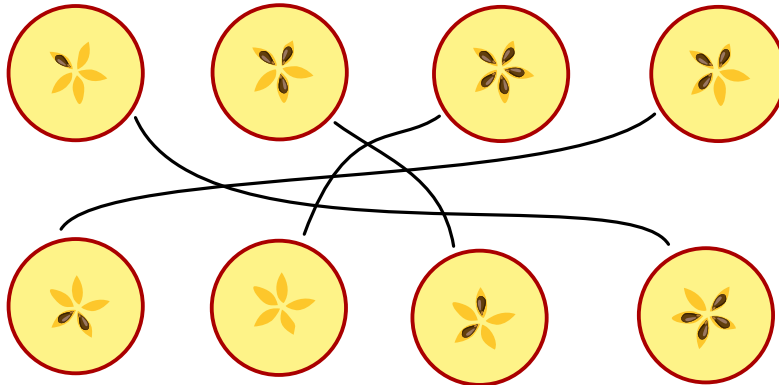
Welche Apfelhälften passen zusammen? Ordne die Apfelhälften einander zu.



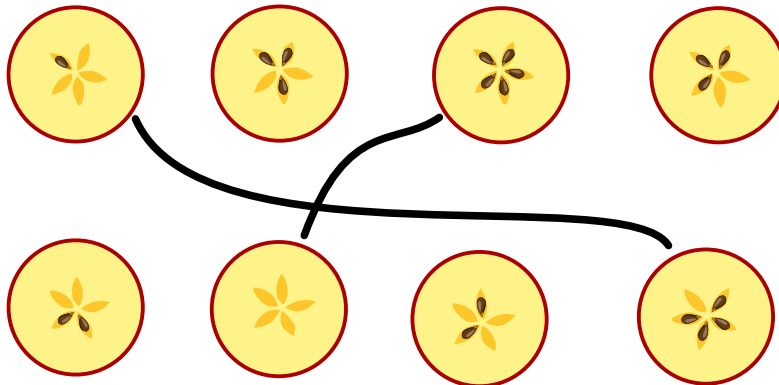


Lösung

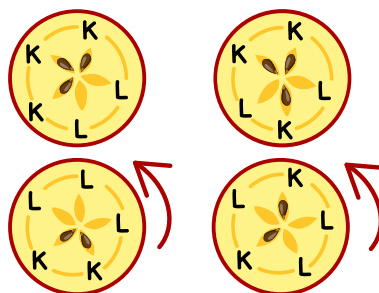
So ist es richtig:



Jeder Apfel hat 5 Apfelkerne. Zwei zueinander passende Apfelhälften müssen also insgesamt 5 Kerne haben. Diese Hälften können damit einfach zugeordnet werden:



Die übrigen vier Hälften sind nicht so einfach zuzuordnen. Die beiden oberen Hälften haben jeweils 3 Kerne, die beiden unteren Hälften haben jeweils 2 Kerne. Deshalb schauen wir uns die Muster der Apfelkerne genauer an. Denn wenn zwei Hälften zusammen passen, passen auch die Muster zusammen. Um das zu sehen, kann es aber nötig sein, die Hälften zu drehen. Dann lassen sich die Hälften so zuordnen wie unten im Bild: Links hat die obere Hälfte drei Kerne direkt hintereinander und danach zwei Löcher (K-K-K-L-L), die untere Hälfte hat drei Löcher direkt hintereinander und danach zwei Kerne (L-L-L-K-K): die Hälften passen zusammen. Auch die rechten Hälften passen zusammen: Das Muster der oberen Hälfte lautet K-L-K-L-K, das Muster der unteren Hälfte L-K-L-K-L.





Dies ist Informatik!

Bei der Erklärung der richtigen Antwort haben wir gesehen: Wenn zwei Hälften zusammen passen, passen nicht nur die Anzahlen der Kerne zusammen, sondern auch die Reihenfolgen der Kerne und Löcher (also der leeren Kernfächer). Für die richtige Zuordnung der Hälften muss man also auch diese Reihenfolgen betrachten. Es genügt nicht zu wissen, wie viel Kerne in jeder Hälfte sind.

Bei Problemen, die mit Hilfe von Computerprogrammen gelöst werden sollen, stellen sich ähnliche Fragen. Informatikerinnen und Informatiker müssen sich Gedanken machen, wie die Informationen, die das Programm berücksichtigen soll, als Daten beschrieben werden. Dabei wird oft versucht, es so einfach wie möglich zu machen. Einfache Programme sind nämlich weniger anfällig für Fehler. Beim Apfelhälften-Problem in dieser Biberaufgabe schien es zunächst ausreichend zu sein, die Hälften allein durch die Anzahl der Kerne zu beschreiben. Doch dann wurde klar, dass das nicht in allen Fällen genügt. Zur Beschreibung der Apfelhälften in einem Computerprogramm muss es also möglich sein, eine Reihenfolge zu beschreiben. Das geht zum Beispiel mit Hilfe der Datenstruktur *Liste*, die in den meisten Programmiersprachen zur Verfügung steht.

Stichwörter und Webseiten

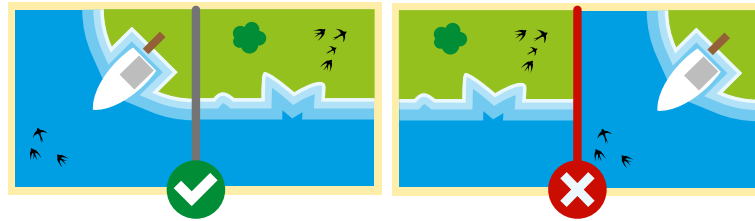
- Reihenfolge
- Liste: [https://de.wikipedia.org/wiki/Liste_\(Datenstruktur\)](https://de.wikipedia.org/wiki/Liste_(Datenstruktur))





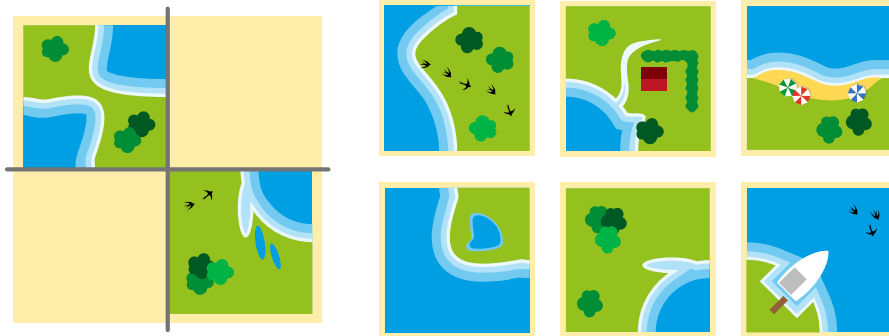
2. Wasser – Land

Edu hat ein neues Spiel. Es besteht aus Kärtchen mit Wasser- und Landflächen. Mit den Kärtchen kann Edu Landschaften legen. Die Kärtchen müssen *zusammenpassen*: Land an Land. Wasser an Wasser.



Edu legt zwei Kärtchen und lässt zwei Lücken.

Welche Kärtchen passen in die Lücken?





Lösung

Das ist die richtige Antwort:



Die beiden Kärtchen passen in die Lücken: Überall liegt Wasser an Wasser und Land an Land. Von den sechs möglichen Kärtchen passen nur diese beiden in die Lücken.

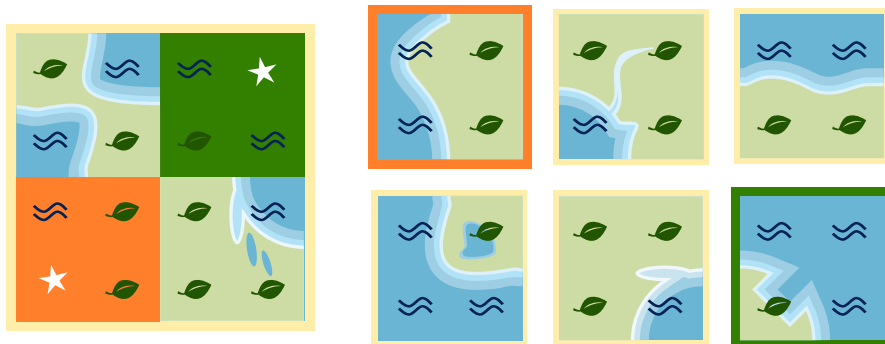
Nur wenn man die Kärtchen drehen dürfte, würden noch weitere Kärtchen in die Lücken passen.

Dies ist Informatik!

Schauen wir uns Edus Kärtchen genauer an. Alle Kärtchen können in vier Bereiche geteilt werden. Die äusseren Ränder dieser Bereiche zeigen entweder Land oder Wasser.














Es gibt also nur zwei verschiedene Bereicharten, denn die äusseren Ränder zeigen entweder Wasser (≡) oder Land (🌿).



Zwei Kärtchen passen nur zusammen, wenn ihre benachbarten Bereicharten gleich sind. Daher können wir für drei Bereiche der Lücken die erforderliche Art eintragen. Der vierte Bereich kann Wasser oder Land sein, deswegen tragen wir ★ ein.



Auf diese Weise erstellen wir für jede Lücke ein Muster. Die Kärtchen, die die Lücken füllen sollen, müssen in diese Muster passen: Bei  und  muss der Bereich des Kärtchens auch  beziehungsweise  haben. Bei  kann der Bereich  oder  haben.

Wir haben eine Eigenschaft der Kärtchen entdeckt. Diese Entdeckung haben wir genutzt, um sie durch eine Anordnung der Zeichen  und  zu ersetzen. Durch diesen Schritt haben wir, die in den Bildern enthaltenen Informationen, erheblich reduziert. Wir konzentrieren uns auf die Information, die zur Lösung dieser Aufgabe erforderlich sind. Informatiker würden sich auf die Anordnung der Zeichen in den Bildern beziehen. Durch die Reduktion der Bilder auf die Bereicharten  und  entsteht ein Modell für die fehlenden Kärtchen. *Modellierung* bedeutet *Abstraktion* (oder Vereinfachung), und Abstraktion reduziert die Information. Computer müssen mit Modellen von der Realität arbeiten. Bei der Erstellung solcher Modelle muss darauf geachtet werden, dass wichtige Eigenschaften der Realität nicht verloren gehen.

Stichwörter und Webseiten

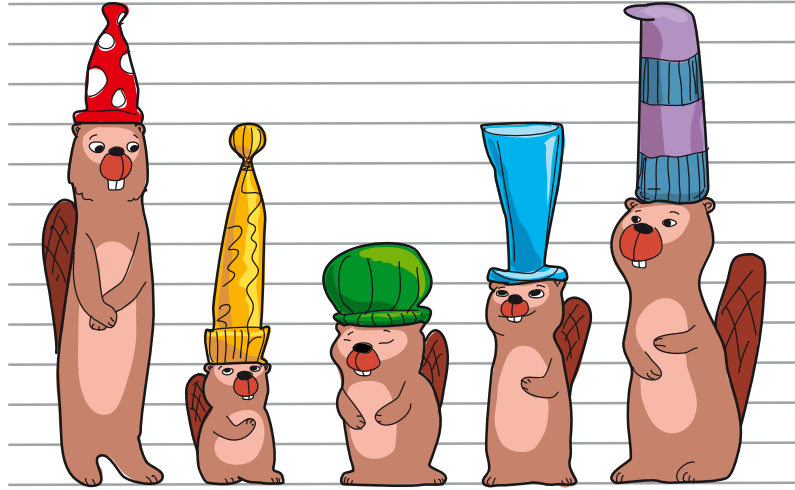
- Modellierung, Codierung: <https://de.wikipedia.org/wiki/Datenmodellierung>





3. Neue Hüte

Die Biber haben neue Hüte.

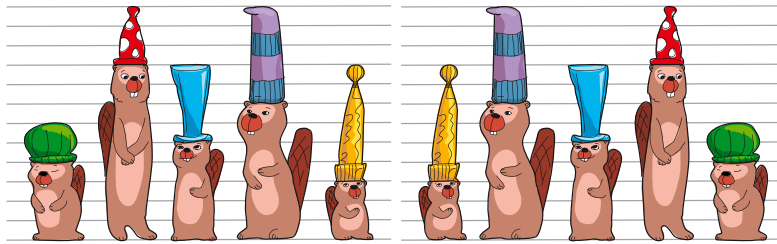


Sortiere die Hüte der Grösse nach.



Lösung

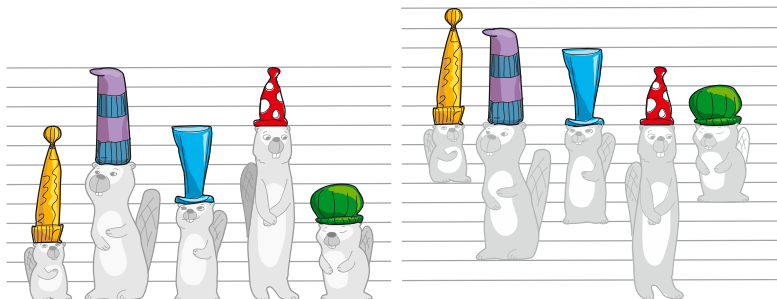
So sind die Hüte richtig sortiert:



Es gibt zwei richtige Lösungen, die Hüte werden von links nach rechts

- immer grösser oder
- immer kleiner.




Beim Sortieren der Biber achten wir nur auf die Hüte. Dann ist es viel einfacher, sie der Grösse nach zu sortieren.

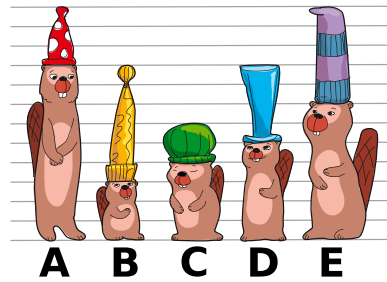


Dies ist Informatik!





Viele Dinge in unserer Umgebung sind sortiert, um einzelne Dinge besser herausuchen zu können: Wenn Werkzeuge nach Grösse sortiert sind, lässt sich das passende Werkzeug leichter finden. Weil die Einträge in einem Wörterbuch alphabetisch sortiert sind, kann man die Seite mit dem gesuchten Wort schneller finden.

In dieser Aufgabe sollst du die Biber sortieren, und zwar der Grösse der Hüte nach. Die Schwierigkeit ist aber, dass die *Eigenschaft* «Grösse der Hüte» nicht leicht erkennbar ist. Wir könnten nach mindestens drei unterschiedlichen Grössen sortieren:

- Grösse der Biber ()
- Grösse der Hüte ()
- gesamte Grösse ()



Die Sortierung der Biber ist für jede der drei Grössen-Eigenschaften unterschiedlich.

Biber			 + 
A	3	9	12
B	6	3	9
C	2	4	6
D	4	5	9
E	5	7	12

Zum Sortieren ist es also erstens wichtig, die Eigenschaft genau festzulegen, nach der sortiert werden soll. Zweitens müssen die Werte dieser Eigenschaft sortierbar sein. Nach Eigenschaften, die in Zahlen ausgedrückt werden (wie z.B. Grösse, Länge, Gewicht, ...) kann man sortieren: Für zwei verschiedene Zahlen können wir sagen, welche Zahl die kleinere ist. Wörter kann man sortieren, weil die Reihenfolge der Buchstaben im Alphabet festgelegt ist und deshalb für zwei verschiedene Wörter klar ist, welches weiter vorn im Wörterbuch stehen muss. Allgemein kann man sagen: Eine Eigenschaft ist sortierbar, wenn man für ihre einzelnen Werte eine eindeutige «kleiner als»-Beziehung (eine *Ordnung*) angeben kann.

Mit Computern werden grosse Datenmengen verwaltet. Um darin einzelne Daten schnell finden zu können, müssen sie sortiert sein. Die Informatik kennt viele schnelle Sortierverfahren, und es ist gut untersucht, in welchen Fällen welche Verfahren angewendet werden sollen.

Stichwörter und Webseiten

- Sortierung: <https://de.wikipedia.org/wiki/Sortierung>
- Ordnungsrelation: <https://de.wikipedia.org/wiki/Ordnungsrelation>
- Suchalgorithmus: <https://de.wikipedia.org/wiki/Suchverfahren>

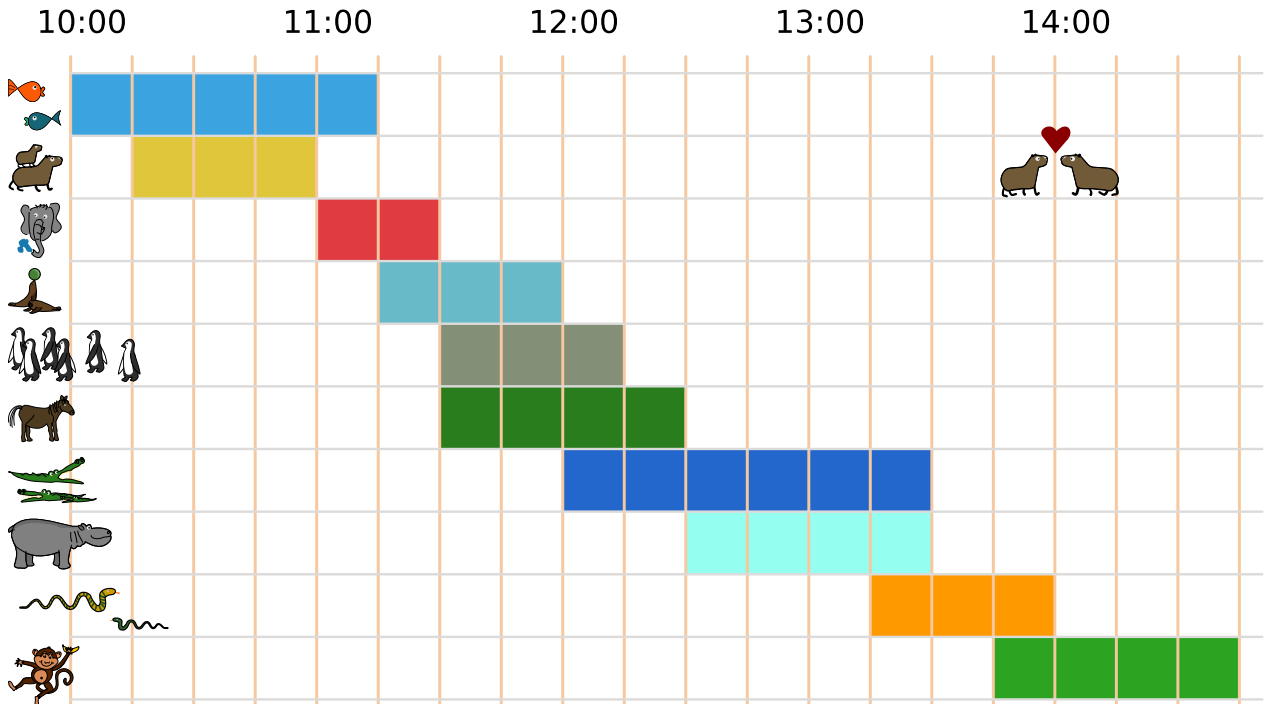




4. Spass im Zoo

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

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



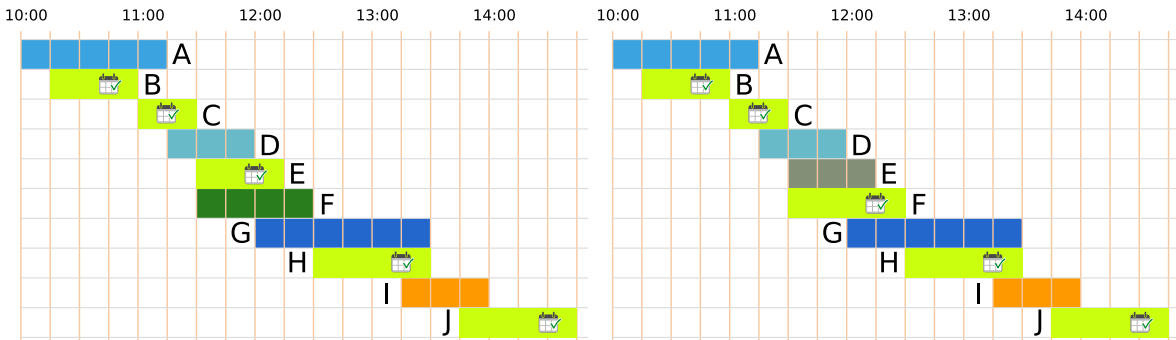
Anja besucht eine Vorführung immer ganz, von Anfang bis Ende. Kannst du Anja helfen?

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



Lösung

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



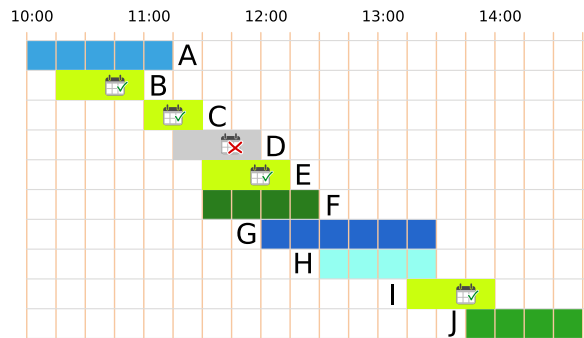
Es gibt unterschiedliche Wege, die richtigen Antworten zu finden.

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

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

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

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





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

Dies ist Informatik!

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

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

Stichwörter und Webseiten

- Scheduling-Problem: <https://de.wikipedia.org/wiki/Scheduling>
- Betriebssystem: <https://de.wikipedia.org/wiki/Betriebssystem>





5. Annas Regenschirm



Das ist Annas Regenschirm:

Eines der vier Bilder zeigt Annas Regenschirm. Welches?



A)



B)



C)

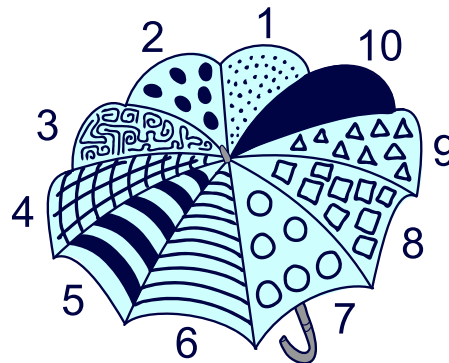


D)



Lösung

Jedes Muster auf Annas Regenschirm kommt genau einmal vor.



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

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

	A)	B)	C)	D)
Antwortbild				
Annas Regenschirm				

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

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

Dies ist Informatik!

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



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

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

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

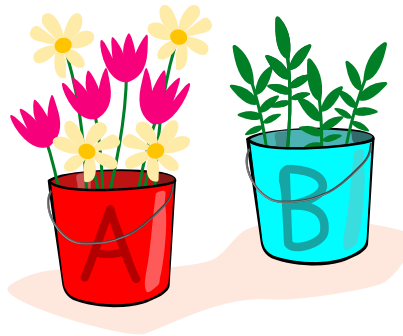
Stichwörter und Webseiten

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








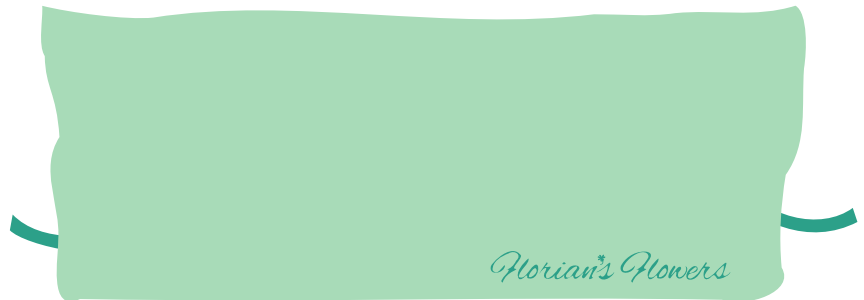
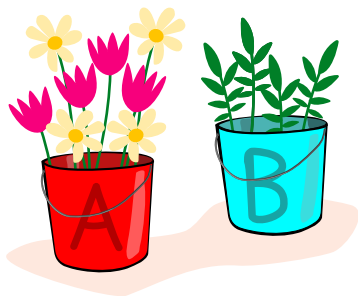
6. Blumenstrauß



Florian verkauft Blumensträuße. Jeden Blumenstrauß bindet Florian nach dieser Anleitung:

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

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



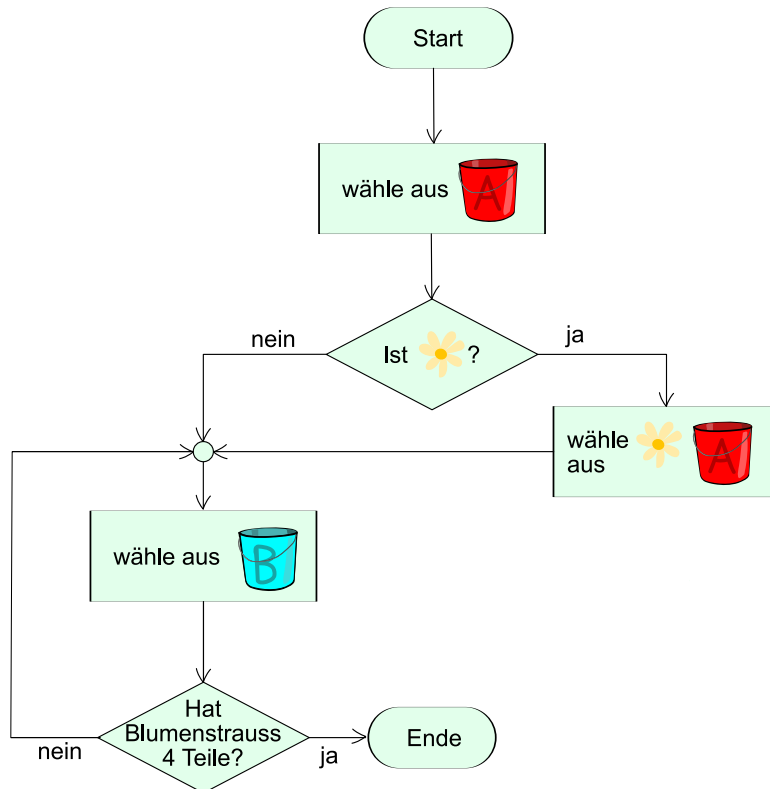


Lösung

Es gibt zwei richtige Lösungen:



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




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

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

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



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

Dies ist Informatik!

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

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

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

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

Stichwörter und Webseiten




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



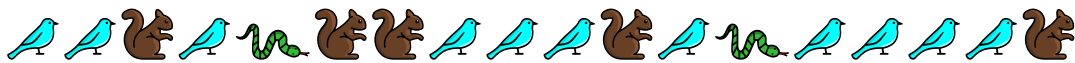


7. Besonderer Baum

Ben hat einen besonderen Apfelbaum im Garten:

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

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



Wie viele Äpfel hängen danach am Baum?

- A) 3 Äpfel
- B) 7 Äpfel
- C) 17 Äpfel
- D) 31 Äpfel



Lösung

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

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

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

Tier:	Übertrag								
Anweisung:	-	-	-	+2	+2	+2	-1	+2	reset
Anzahl Äpfel:	0	0	0	2	4	6	5	7	0

Tier:	Übertrag					
Anweisung:	-	+2	+2	+2	+2	-1
Anzahl Äpfel:	0	2	4	6	8	7

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

Dies ist Informatik!

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

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

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

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



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

Stichwörter und Webseiten

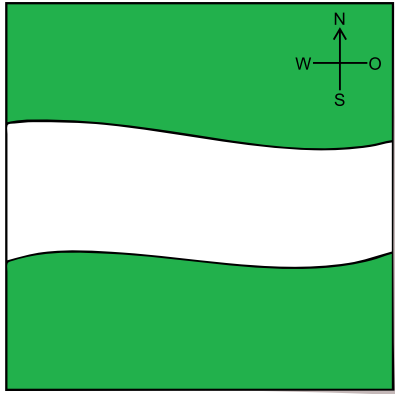
- Variablen: [https://de.wikipedia.org/wiki/Variable_\(Programmierung\)](https://de.wikipedia.org/wiki/Variable_(Programmierung))
- (Programm-)Zustand: <https://media.kswillisau.ch/in/zustand/zustand.html>



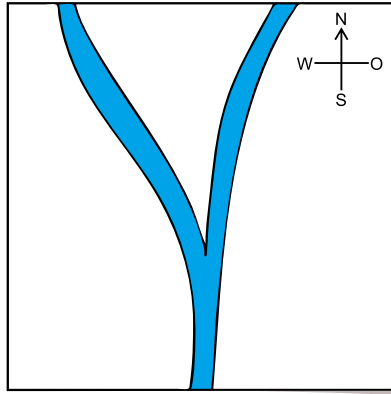


8. Karlas Traumhaus

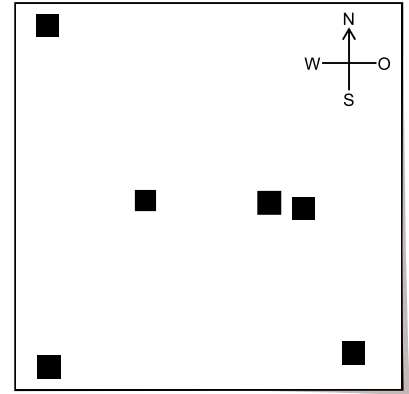
Karla hat drei Karten, die alle genau das gleiche Gebiet zeigen. Eine Karte zeigt die Wälder, eine die Flüsse und eine die Häuser in diesem Gebiet. Karlas Traumhaus liegt im Wald und in der Nähe eines Flusses.



Waldkarte



Flusskarte



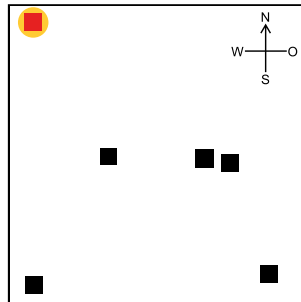
Hauskarte

Welches ist Karlas Traumhaus?

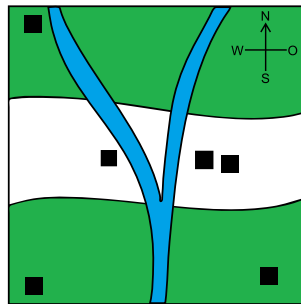


Lösung

Das Haus oben links auf der Hauskarte ist Karlas Traumhaus:



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



Dies ist Informatik!

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

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

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

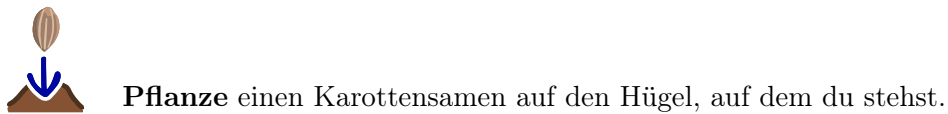
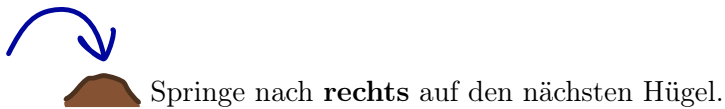
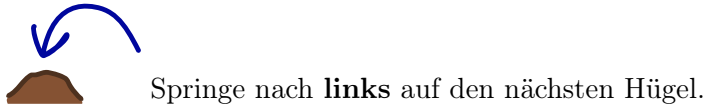
Stichwörter und Webseiten

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



9. Rüeblli pflanzen

Der Kaninchenroboter kann folgende Anweisungen ausführen:



Der Kaninchenroboter hat diese Folge von Anweisungen ausgeführt:



Dabei ist der Roboter auf vier Hügeln gewesen. Wir wissen aber nicht, auf welchem Hügel er angefangen hat.

Auf welche Hügel hat der Roboter die Rüeblisamen gepflanzt?

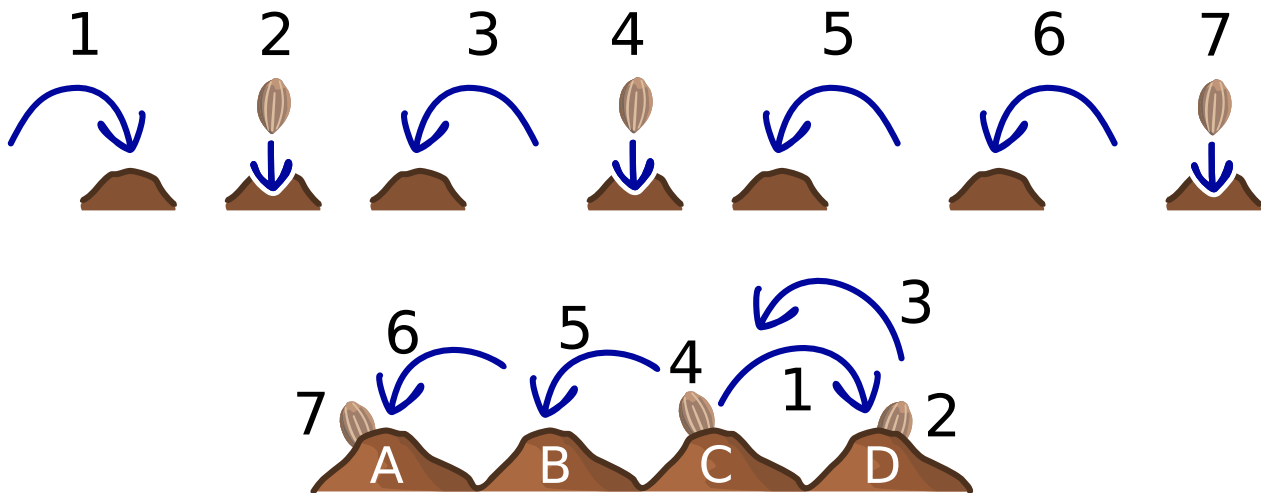




Lösung



Um die richtige Antwort besser erklären zu können, geben wir den Hügeln Buchstaben (siehe oben) und den Anweisungen Nummern:



Zuerst bestimmen wir den Startpunkt des Roboters: Da der Roboter dreimal hintereinander nach links springt (Anweisungen 3, 5, 6), muss er vorher auf Hügel D stehen. Bevor er dreimal nach links springt, springt er einmal nach rechts (Anweisung 1). Der Roboter hat also auf Hügel C angefangen. Folglich werden die Rübliisamen – den Anweisungen 2, 4 und 7 entsprechend – zuerst auf Hügel D, dann auf Hügel C und zuletzt auf Hügel A gepflanzt.

Dies ist Informatik!

Echte Roboter haben eingebaute Computer, und die werden so ähnlich *programmiert* wie der Kaninchenroboter. Ein Computerprogramm besteht aus vielen einzelnen *Anweisungen*.

In unserem Fall wird die Abfolge der Anweisungen für den Roboter-Computer mit Hilfe von Bildblöcken angegeben. Das Ergebnis (*Output*) des Programms hängt nicht nur von der Startposition (*Input*), sondern auch von der Folge und Reihenfolge der Anweisungen ab.

Diese Biberaufgabe zeigt ein Beispiel für den Einsatz von Robotern in der Landwirtschaft. Roboter können nicht nur pflanzen, sondern auch bewässern, bestäuben oder Pflanzenschutzmittel gezielt verteilen.

Stichwörter und Webseiten

- Algorithmus: <https://de.wikipedia.org/wiki/Algorithmus>
- Anweisungen: [https://de.wikipedia.org/wiki/Anweisung_\(Programmierung\)](https://de.wikipedia.org/wiki/Anweisung_(Programmierung))



- Smart Farming: https://de.wikipedia.org/wiki/Smart_Farming,
<https://www.agroscope.admin.ch/agroscope/de/home/themen/wirtschaftstechnik/smart-farming.html>
- Roboter in der Landwirtschaft: <https://cordis.europa.eu/article/id/441912-robots-help-farmers-say-goodbye-to-repetitive-tasks/de>





10. Riccas

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



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



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

Welcher dieser Sätze über Riccas ist nun sicher falsch?

- A) Alle Riccas haben Zähne.
- B) Einige Riccas haben Flügel.
- C) Riccas haben entweder Hörner oder drei Augen, aber nie Hörner *und* drei Augen.
- D) Wenn Riccas genau zwei Arme haben, dann haben sie auch genau zwei Beine.



Lösung

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

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

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

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

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

Dies ist Informatik!

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

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

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



Anhand dieser Festlegungen

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

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

Stichwörter und Webseiten

Modellbildung: <https://de.wikipedia.org/wiki/Modell#Modellbildung>,

https://de.wikipedia.org/wiki/Objektorientierte_Analyse_und_Design Maschinelles Lernen:

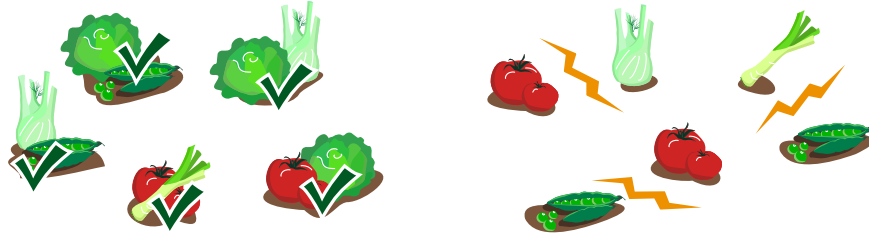
https://de.wikipedia.org/wiki/Maschinelles_Lernen






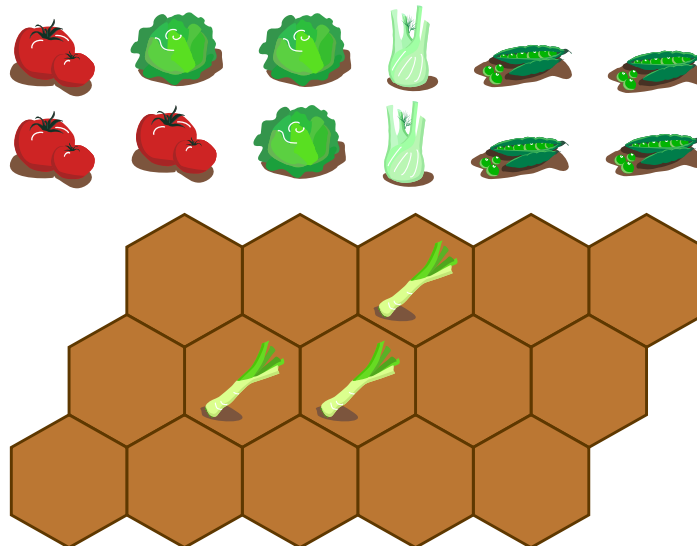
11. Gemüsebeet

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



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

In drei Bereiche hat Lisa schon Lauch  gepflanzt.



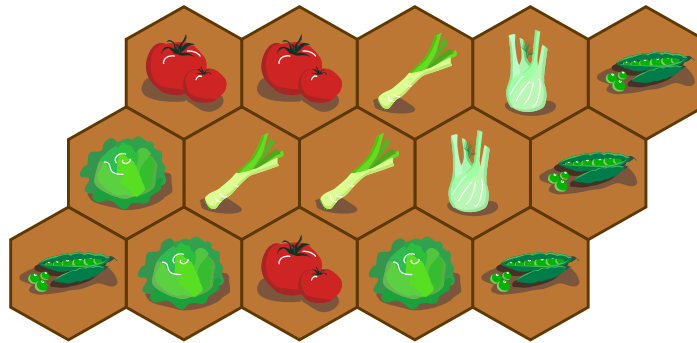
Beim Pflanzen beachtet Lisa folgende Regel: Gemüse, die sich nicht vertragen, dürfen nicht in Bereiche gepflanzt werden, die sich berühren.

Bepflanze alle noch freien Bereiche und beachte Lisas Regel!

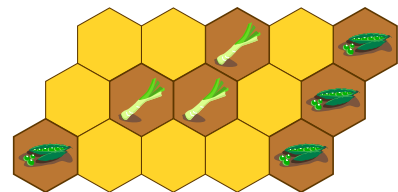


Lösung

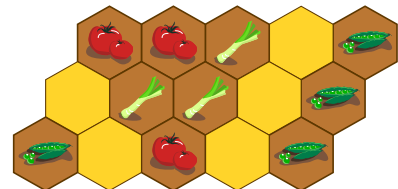
So ist es richtig:



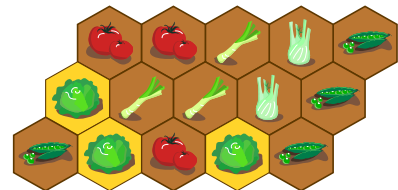
Weil Erbsen sich mit Lauch nicht vertragen, pflanzt Lisa keine Erbsen in die gelben Bereiche. Für die Erbsen bleiben nur die übrigen Bereiche.



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



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



Dies ist Informatik!

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

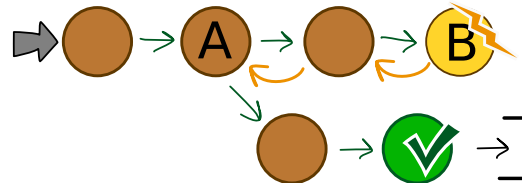
Um eine Bepflanzung von Lisas Beet zu finden, die alle Verträglichkeitsbedingungen beachtet, könnte man so vorgehen: Man probiert systematisch alle Kombinationen aus, die Gemüse auf dem Beet zu platzieren. Erst wenn das Beet voll ist, wird geprüft, ob diese Kombination alle Bedingungen erfüllt und eine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

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



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

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



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

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

Stichwörter und Webseiten

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




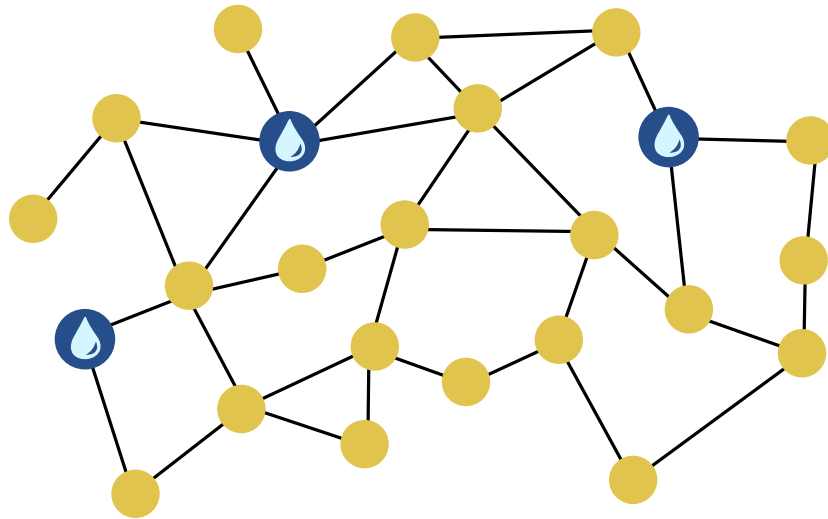


12. Brunnen

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

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

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

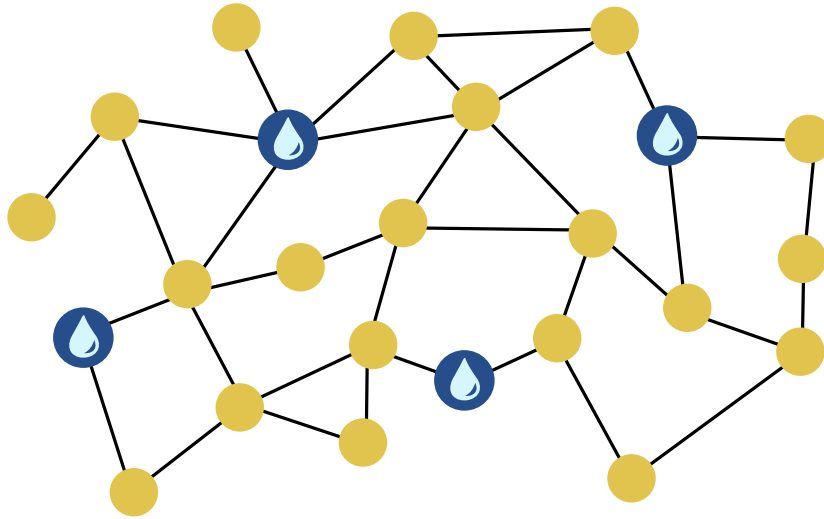


Stelle einen weiteren Brunnen so auf, dass die Bürgermeisterin zufrieden ist.



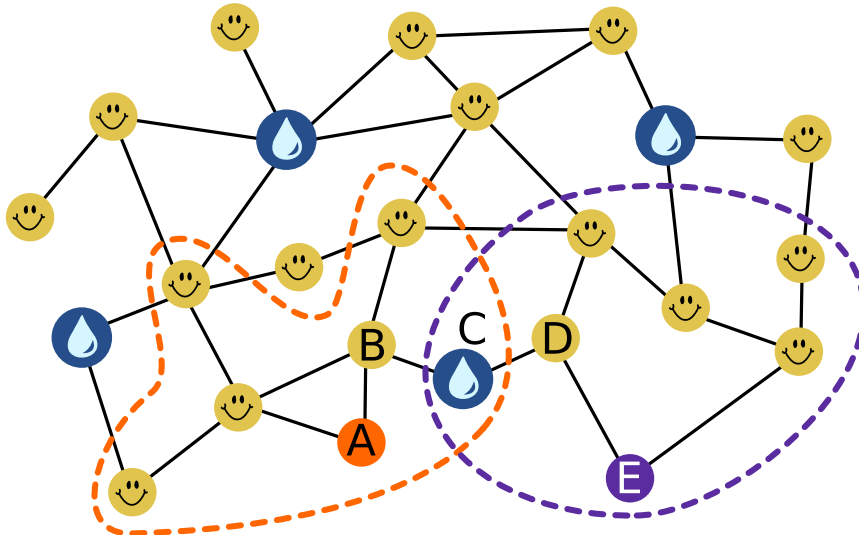
Lösung

So ist es richtig:



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

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



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



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

Dies ist Informatik!

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

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

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

Stichwörter und Webseiten

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



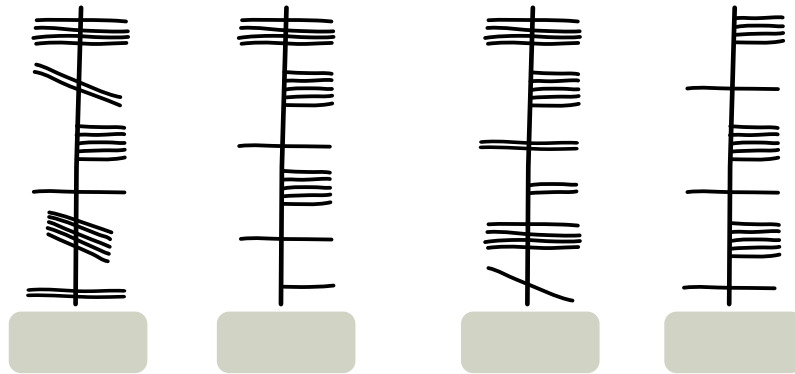


13. Ogham

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

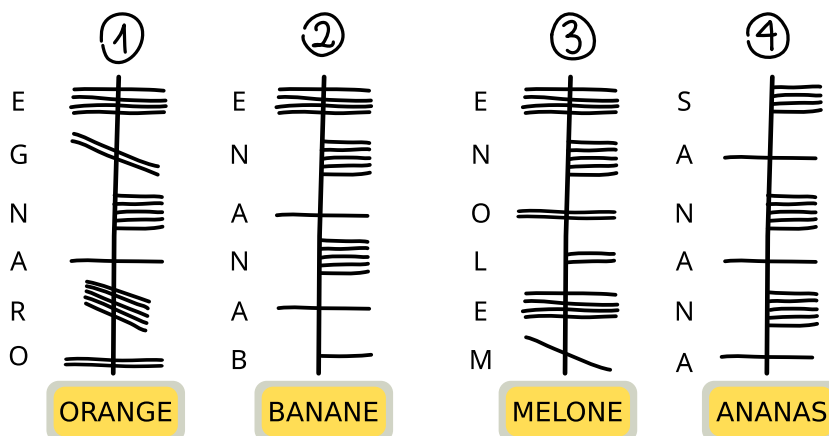
MELONE

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 unbekanntem 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 jahrhundertlang



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>



A. Aufgabenautoren

- | | |
|---|--|
|  Somayah Albaradei |  Vaidotas Kinčius |
|  Esraa Almajhad |  Mhairi King |
|  Aldrich Ellis Catapang Asuncion |  V́ictor Koleszar |
|  Masiar Babazadeh |  Taina Lehtimäki |
|  Leonardo Barichello |  Angélica Herrera Loyo |
|  Liam Baumann |  Carlos Luna |
|  Wilfried Baumann |  Yong Mao |
|  Javier Bilbao |  Yoshiaki Matsuzawa |
|  Špela Cerar |  Natalia Natalia |
|  Sarah Chan |  Marika Parviainen |
|  Marios Omar Choudary |  Jean-Philippe Pellet |
|  Gunnar Collier |  Zsuzsa Pluhár |
|  Eimear Colreavy |  Wolfgang Pohl |
|  Valentina Dagiene |  Estela Ramić |
|  Darija Dasović |  Kirsten Schlüter |
|  Christian Datzko |  Giovanni Serafini |
|  Nora A. Escherle |  Alieke Stijf |
|  Gerald Futschek |  Gabrielė Stupurienė |
|  Bence Gaál |  Marianne Thut |
|  Emily Gates |  Monika Tomcsányiová |
|  Štefan Gura |  Svetlana Unković |
|  Josefine Hiebler |  Jiří Vaníček |
|  Mathias Hiron |  Florentina Voboril |
|  Hyun-seok Jeon |  Michael Weigend |
|  David Khachatryan |  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

Scuola universitaria professionale
della Svizzera italiana



<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.



Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich



<http://www.ubs.com/>

Wealth Management IT and UBS Switzerland IT



<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.



<http://senarclens.com/>

Senarclens Leu & Partner



D. Weiterführende Angebote



Das Lehrmittel zum Informatik-Biber

Module

Verkehr – Optimieren

Musik – Komprimieren

Geheime Botschaften – Verschlüsseln

Internet – Routing

Apps

Auszeichnungssprachen

IT Feuer: <https://it-feuer.ch/>

In der Schweiz engagieren sich zahlreiche Organisationen für die Nachwuchsförderung in Informatik. Die Initiative «IT-Feuer» möchte diese vorhandenen Kräfte bündeln und einen Beitrag leisten, das Thema in der Öffentlichkeit schweizweit bekannter zu machen. Das IT-Feuer präsentiert eine grosse Palette an Angeboten für Lehrpersonen sowie Schüler*innen und Schulklassen.

<http://informatik-biber.ch/einleitung/>

Das Lehrmittel zum Biber-Wettbewerb ist ein vom SVIA, dem schweizerischen Verein für Informatik in der Ausbildung, initiiertes Projekt und hat die Förderung der Informatik in der Sekundarstufe I zum Ziel.

Das Lehrmittel bringt Jugendlichen auf niederschwellige Weise Konzepte der Informatik näher und zeigt dadurch auf, dass die Informatikbranche vielseitige und spannende Berufsperspektiven bietet.

Lehrpersonen der Sekundarstufe I und weiteren interessierten Lehrkräften steht das Lehrmittel als Ressource zur Vor- und Nachbereitung des Wettbewerbs kostenlos zur Verfügung.

Die sechs Unterrichtseinheiten des Lehrmittels wurden seit Juni 2012 von der LerNetz AG in Zusammenarbeit mit dem Fachdidaktiker und Dozenten Dr. Martin Guggisberg der PH FHNW entwickelt. Das Angebot wurde zweisprachig (Deutsch und Französisch) entwickelt.



CoetryLab: <https://www.coetry-lab.org/>

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



Roteco: <https://www.roteco.ch/de/>

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



I learn it: <http://ilearnit.ch/>

In thematischen Modulen können Kinder und Jugendliche auf dieser Website einen Aspekt der Informatik auf deutsch und französisch selbständig entdecken und damit experimentieren. Derzeit sind sechs Module verfügbar.

010100110101011001001001
010000010010110101010011
010100110100100101000101
001011010101001101010011
010010010100100100100001

SV!A

www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikind
er Ausbildung//sociétésuissepourl'infor
matique dans l'enseignement//societàsviz
zeraperl'informaticanell'insegnamento

Werden Sie SVIA Mitglied – <http://svia-ssie-ssii.ch/svia/mitgliedschaft> und unterstützen 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.