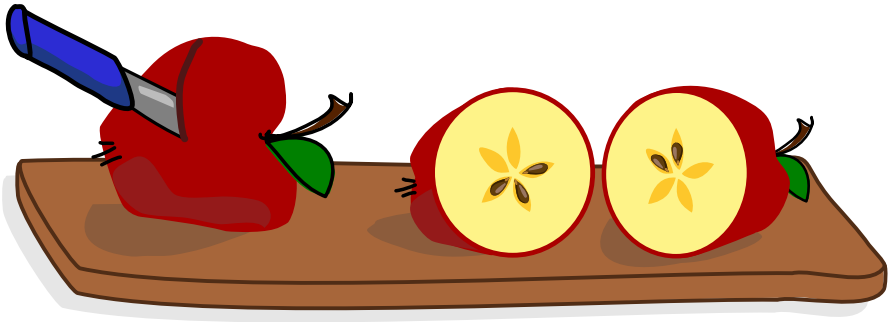
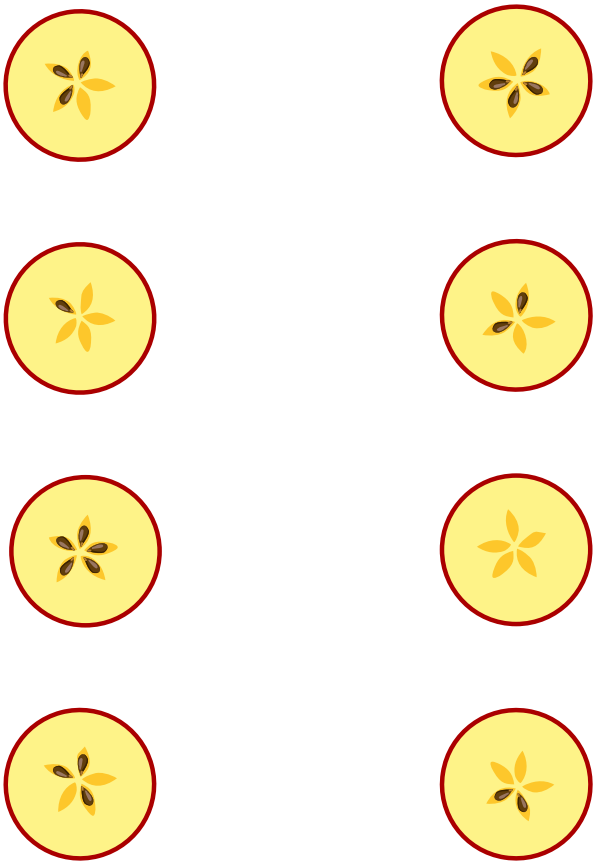
Äpfel halbieren

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:

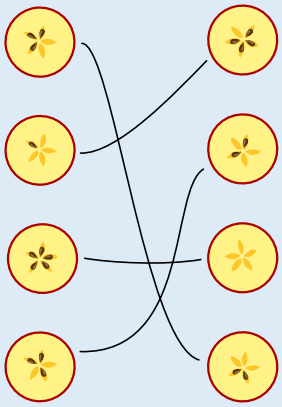


Gala halbiert vier Äpfel. Sie legt die oberen Hälften links und die unteren Hälften rechts untereinander.

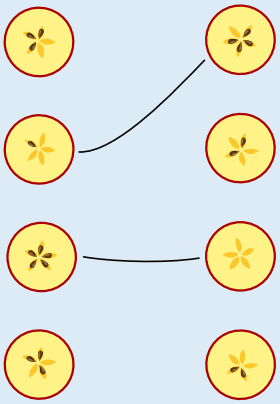
Welche Apfelhälften passen zusammen? Ordne die Apfelhälften einander zu.

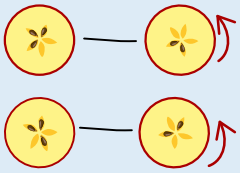


So ist es richtig:



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 im Bild: Oben 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 unten passen die beiden Hälften zusammen: Das Muster der oberen Hälfte lautet (wenn man oben beginnt und im Uhrzeigersinn weitermacht) K-L-K-L-K, das Muster der unteren Hälfte (wenn man unten beginnt) L-K-L-K-L.

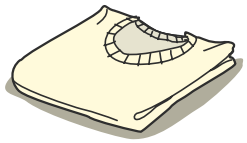
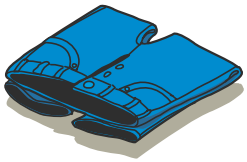
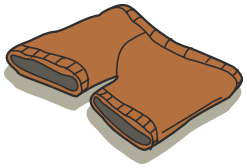
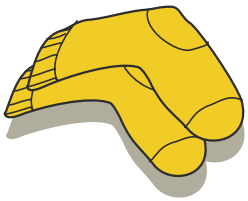
Das ist Informatik!

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 viele 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.

Anzieh-Stapel

Das sind Brunos Anziehsachen:

Hemd	Hose	Unterhose	Socken	Schuhe
				

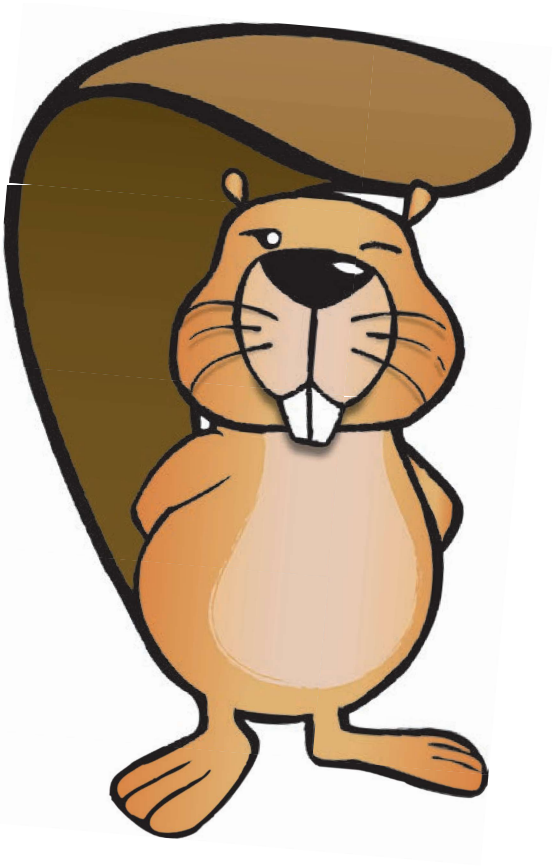
Abends legt Bruno seine Sachen auf einen Stapel.
Dann geht es morgens schnell:
Er zieht zuerst die oberste Sache an, dann die nächste – schön der Reihe nach.

Aber manchmal ist der Stapel falsch:
Dann sind am Ende die Socken über den Schuhen oder die Unterhose über der Hose.
Oh je!

Diese Stapel sind fast alle falsch. Nur einer ist richtig.
Welcher?

A) **B)** **C)** **D)**

Auch mit Biberheften kannst du Stapel bilden.
Es gibt sie schon seit dem Jahr 2007.
Alle Biberhefte kannst du hier finden:
bwinf.de/biber/downloads



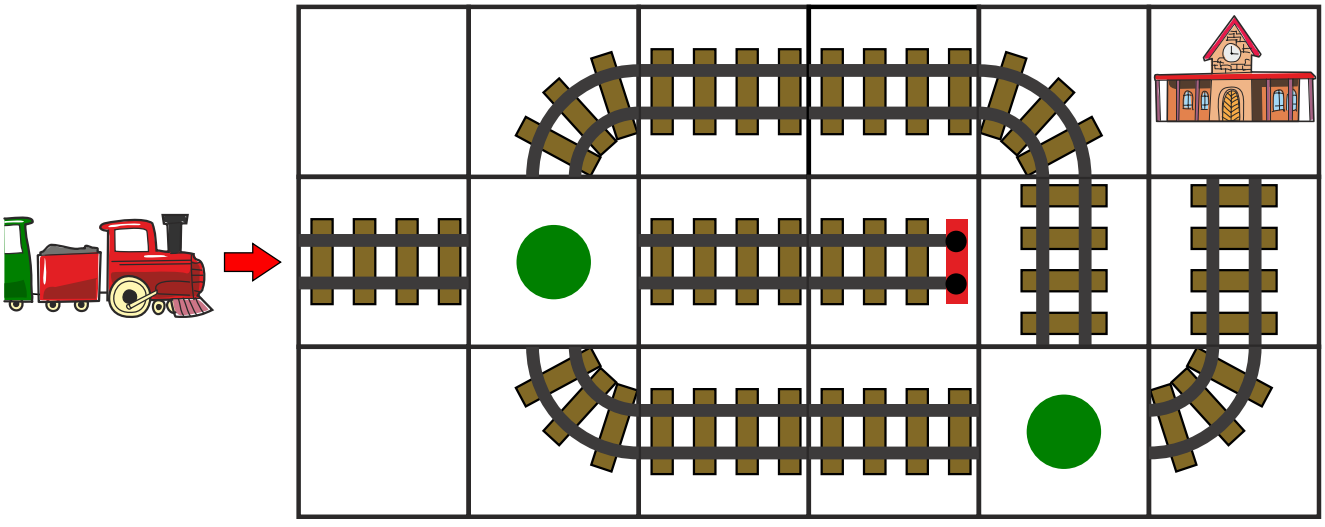
Antwort D ist richtig:
Bei Stapel A zieht Bruno die Socken über die Schuhe an.
Bei den Stapeln B und C zieht Bruno die Unterhose über die Hose an.
Bei Stapel D sind zwar merkwürdigerweise die Socken zuerst an der Reihe, aber am Ende ist alles richtig: Die Socken sind in den Schuhen, und die Unterhose ist unter der Hose.

Das ist Informatik!
Das ist klar: Die Unterhose muss vor der Hose und die Socken müssen vor den Schuhen angezogen werden. Aber ob das Hemd vor den Socken angezogen wird oder umgekehrt, ist egal. Nur gleichzeitig geht es nicht, und deshalb wird eine Reihenfolge benötigt, bei der eine Sache nach der anderen kommt. Eine Reihenfolge, bei der alle „vor-Sachen" (z. B. die Socken) auch wirklich vor ihren „danach-Sachen" (z. B. die Schuhe) und alle anderen Sachen irgendwo stehen, wird in der Mathematik als topologische Sortierung bezeichnet. Die Informatik kennt Verfahren, mit denen man eine topologische Sortierung berechnen kann – oder herausfinden kann, dass es keine gibt: Wenn Brunos Eltern auf den merkwürdigen Gedanken kämen, dass er das Hemd vor der Unterhose, die Unterhose vor der Hose und die Hose vor dem Hemd anziehen müsste, könnte Bruno niemals einen richtigen Stapel legen.

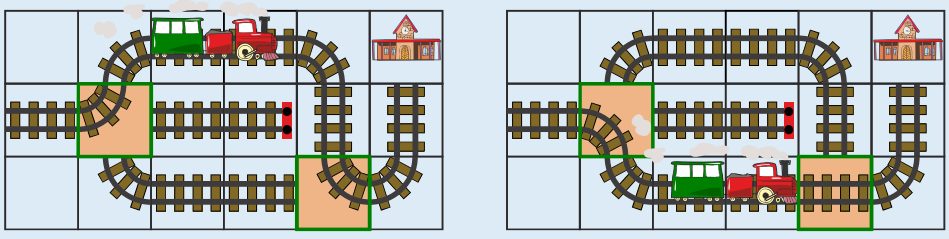
Zum Bahnhof

Ziehe Schienen auf die grünen Punkte,

so dass der Zug  zum Bahnhof  fahren kann.



So ist es richtig:
Es gibt zwei Möglichkeiten, Schienenstücke so auf die freien Stellen zu legen, dass der Zug zum Bahnhof fahren kann.

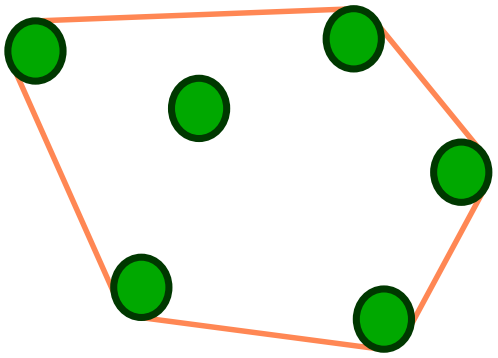
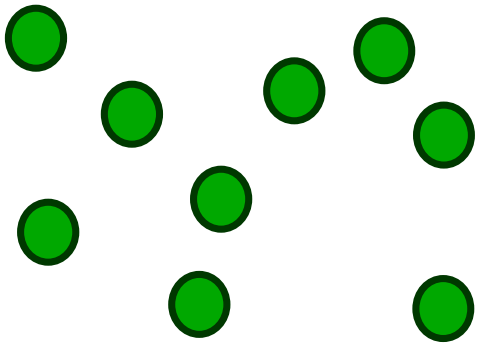


Bei allen anderen Möglichkeiten entgleist der Zug oder fährt gegen den Prellbock.

Das ist Informatik!
Wie der Zug in dieser Biberaufgabe genau den Schienen entlang fährt, führt ein Computer genau die Anweisungen eines Programms aus. Der Computer kann nicht erkennen, ob ein Programm einen Fehler enthält – so wie ein Zug nicht erkennen kann, ob Schienen falsch verlegt wurden. Informatikerinnen und Informatiker versuchen deshalb, in Programme „Notbremsen" einzubauen, mit denen verhindert werden kann, dass Programmfehler Schäden verursachen. Wenn das nicht gelingt, kann der Computer „abstürzen" und unbenutzbar werden, so wie ein Zug entgleisen kann. Wer ein Programm schreibt, muss also ähnlich sorgfältig sein wie Menschen, die Schienen für Züge verlegen.

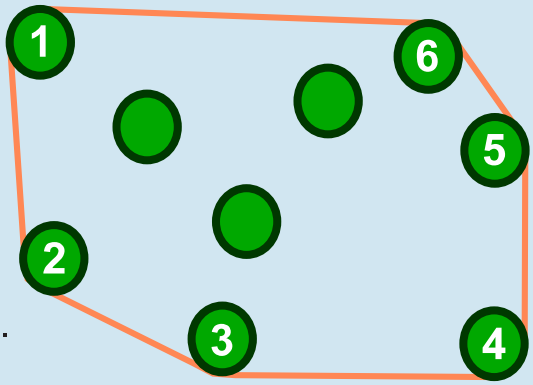
Bäume-Band

Die Biber spannen immer ein langes Band um Bäume, die sie fällen wollen.

Gestern wollten sie sechs Bäume fällen. Das Band hat aber nur fünf Bäume berührt. Aus der Luft sah das so aus:	Heute wollen die Biber diese Bäume fällen:
	

Wie viele Bäume berührt das gespannte Band diesmal?
A) 3 Bäume **B)** 5 Bäume **C)** 6 Bäume **D)** 9 Bäume

Antwort C ist richtig:
Die Biber spannen das Band so um die Bäume:



Das Band berührt die sechs nummerierten Bäume.

Das ist Informatik!
Wenn die Bäume im Lösungsbild winzige Punkte wären, hätte das Biber-Bäume-Band die Form eines Sechsecks. Dieses Sechseck ist das kleinste Vieleck, auf dem alle zu fällenden Bäume stehen.
Ein solches kleinstes Vieleck, das alle Punkte aus einer vorgegebenen Menge enthält, heißt in der Mathematik auch „konvexe Hülle“ dieser Punktmenge. Dabei bedeutet „konvex“, dass sich etwas nach außen dehnt, wie z. B. die konvexe Linse einer Lupe. Und eine „Hülle“ ist etwas, das etwas anderes umschließt, so wie die Haut den Körper, dabei aber nicht größer als nötig ist. Das Biber-Bäume-Band ist also wie eine solche konvexe Hülle: Es umschließt alle Bäume und geht zwischen zwei Bäumen nicht nach innen – und „nicht nach innen“ ist in der Mathematik schon so viel wie „nach außen“.
In der Informatik ist es häufig wichtig, die konvexe Hülle einer Menge von Punkten zu berechnen:

- Mustererkennung: Ist in einem Bild ein Gesicht zu sehen?
- Handschrifterkennung: Ist ein handgeschriebenes Zeichen der Buchstabe B?
- Geographische Informationssysteme: Wie groß ist ein Überschwemmungsgebiet oder ein Flusssystem?
- Verpackungen: Was ist die kleinste Menge an Material, die genügt, einen bestimmten Gegenstand zu verpacken?

Die Informatik kennt Verfahren, welche die konvexe Hülle einer Punktmenge effizient berechnen können. Sie funktionieren also auch dann noch gut, wenn die Punktmenge sehr groß ist.

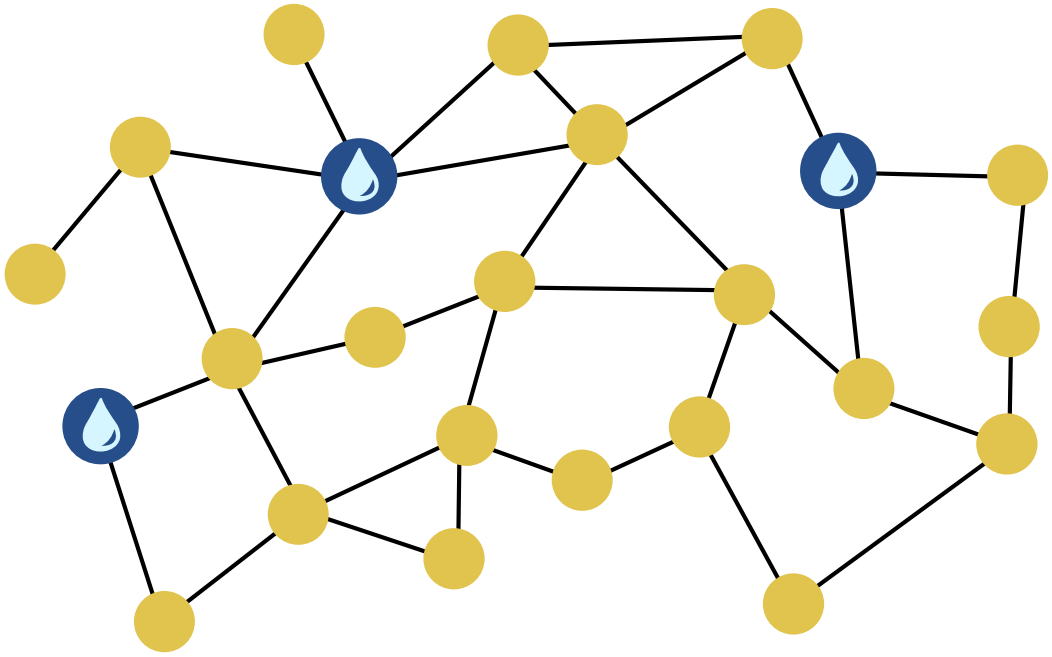
Brunnen

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

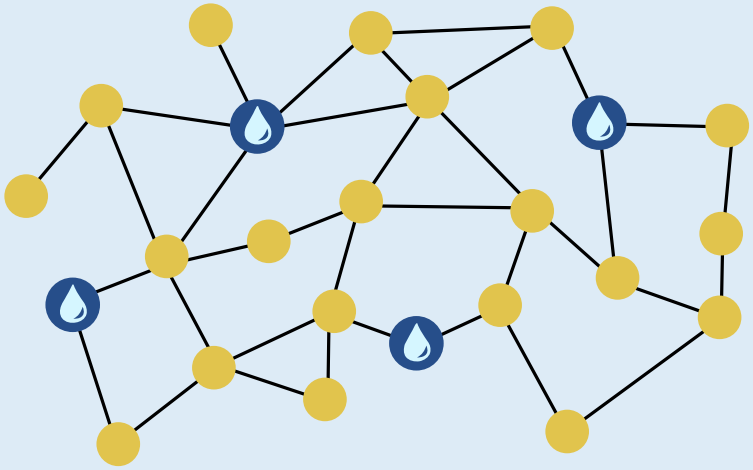
Die Brunnen sollen so stehen, dass man von jeder Straßenecke aus
höchstens zwei Straßenabschnitte gehen muss, um einen Brunnen zu erreichen.
Dann ist die Bürgermeisterin zufrieden.

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.



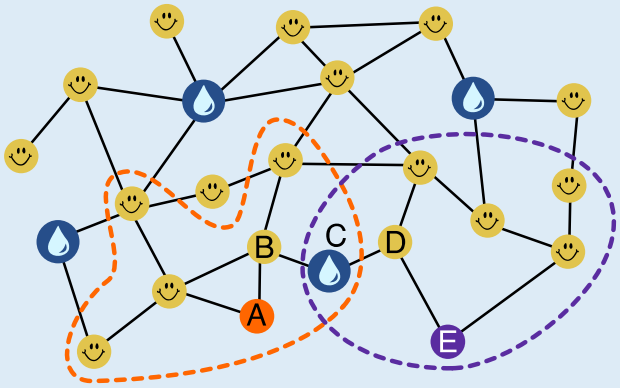
So ist es richtig:



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

Wie können wir herausfinden, an welcher Straßenecke ein weiterer Brunnen aufgestellt werden soll?

Im Stadtplan markieren wir alle Straßenecken mit einem  , die höchstens zwei Straßenabschnitte von einem der Brunnen entfernt sind, die bereits aufgestellt sind. In Bezug auf diese Ecken kann die Bürgermeisterin bereits zufrieden sein.



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

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

Das 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.

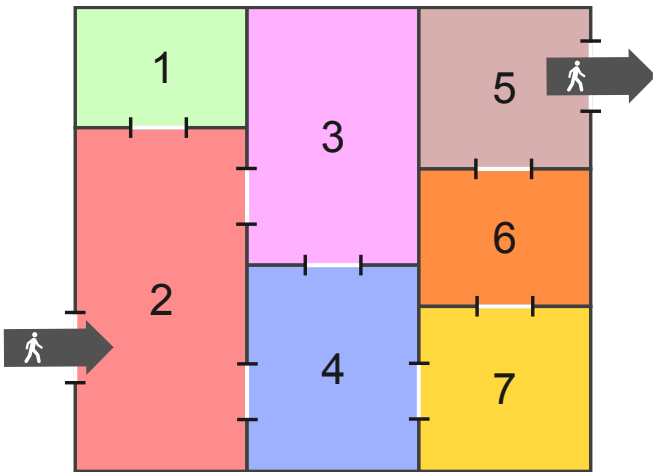
Rundgang

Ein neues Museum wird geplant. Die Besucher sollen darin einen Rundgang machen.
Bei einem Rundgang geht man durch alle Räume und betritt jeden Raum nur einmal.

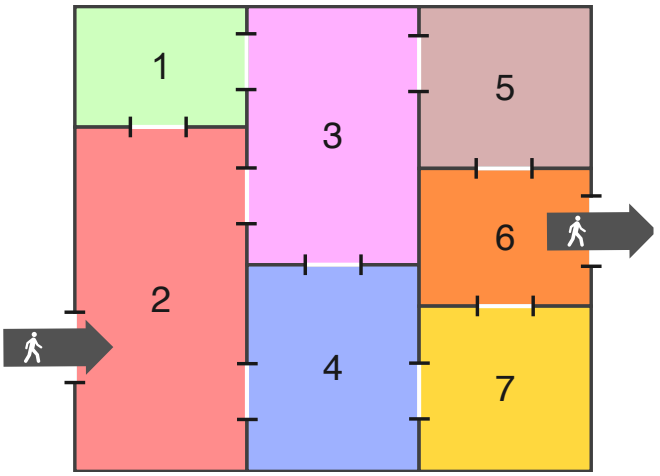
Es werden vier Pläne gemacht. Alle haben sieben Räume (1 bis 7).
Die Pfeile zeigen, wo die Besucher in das Museum hineingehen und wo sie wieder hinausgehen sollen.

Nur bei einem Plan kann man einen Rundgang machen. Bei welchem?

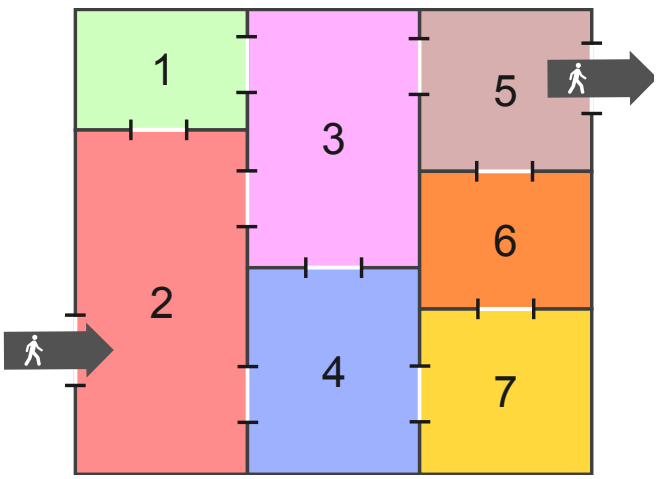
A)



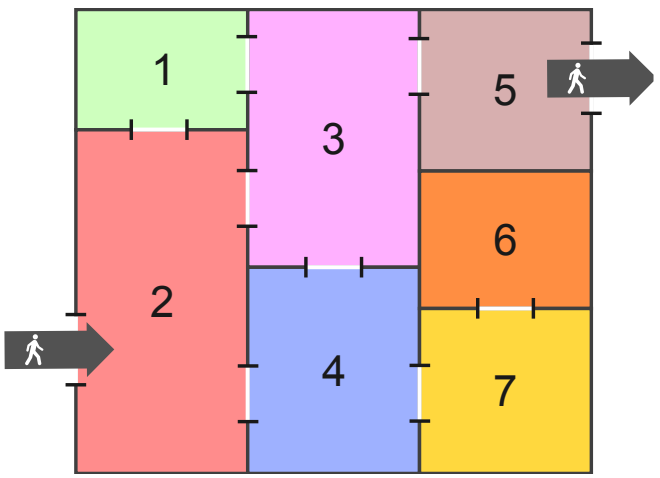
B)



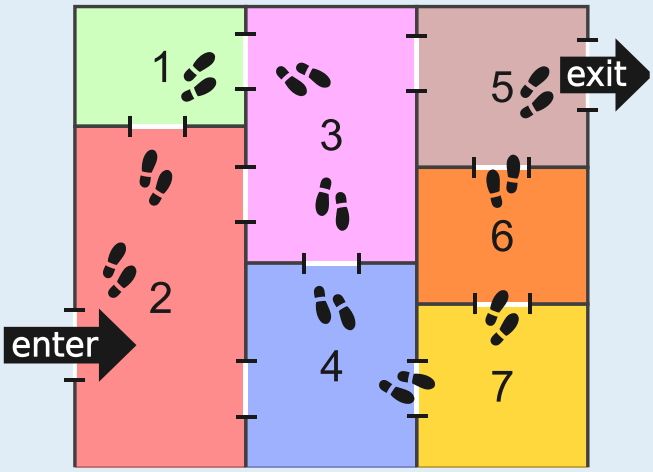
C)



D)

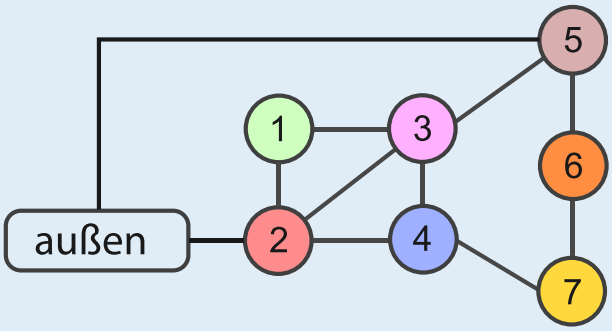


Nur der Plan C erlaubt den Besuchern, einen Rundgang zu machen, und zwar so:



Falls ein Raum nur einen Eingang hat, kann man keinen Rundgang nach den vorgegebenen Regeln machen: Wenn eine Besucherin diesen Raum betritt, kann sie ihn nur durch den einen Eingang auch wieder verlassen. So kommt sie zurück in den Raum, aus dem sie gekommen ist, und betritt diesen zum zweiten Mal. Damit ist die Rundgang-Regel verletzt.
Deshalb erlauben die Pläne A und D keinen Rundgang. In Plan A hat Raum 1 nur einen Eingang, und in Plan D hat Raum 6 nur einen Eingang.
In Plan B kann der Raum 6 von Raum 5 oder von Raum 7 erreicht werden. Falls die Besucherin von Raum 5 kommt, muss sie Raum 6 durchqueren, um den Raum 7 zu betreten (und umgekehrt). Dann kann sie den Ausgang nur noch erreichen, indem sie Raum 6 zum zweiten Mal betritt.

Das ist Informatik!



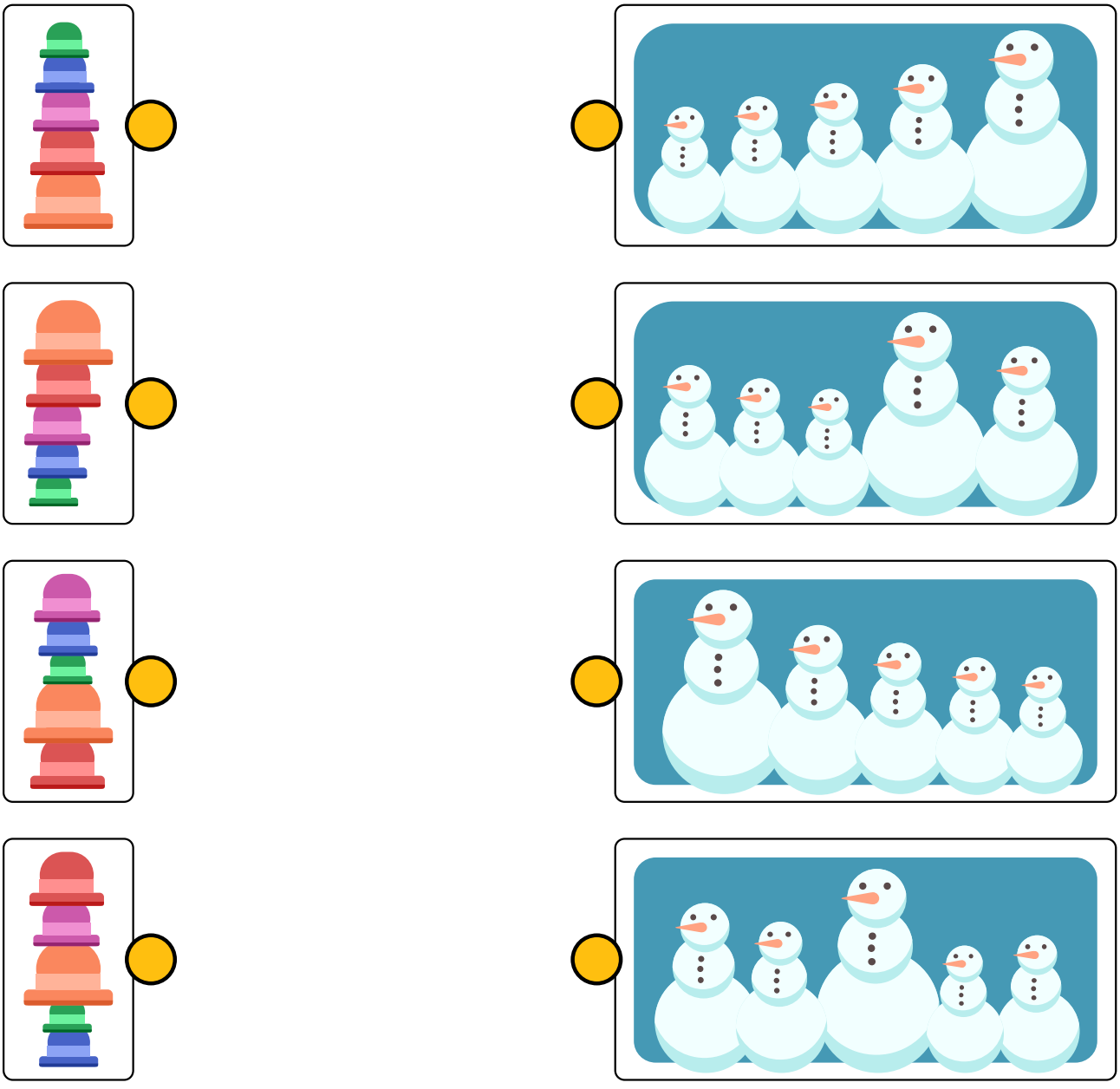
Auch wenn das Museum nicht sehr groß ist, muss man genau hinsehen, wo man lang geht, um die möglichen Wege zu prüfen. Dabei sind die Größe der Räume und deren genauen Lage für die Frage nach einem Rundgang unerheblich. Man muss nur wissen, von welchem Raum man in welche anderen Räume gelangen kann und wo die Verbindungen nach außen sind. Das können wir für Plan C so aufzeichnen:
Damit haben wir den Plan auf das Wesentliche reduziert – und ihn als Graphen modelliert (siehe auch die Aufgabe „Geschäfte“ in diesem Biberheft). Ein Graph ist eine mathematische Struktur aus Knoten und Kanten: Hier entsprechen die Räume (einschließlich des Raums „außen“) den Knoten und die Türen den Kanten. Ein Rundgang ist dann ein Weg über Kanten des Graphen, bei dem alle Knoten genau einmal besucht werden, bis man beim ersten Knoten („außen“) wieder ankommt.
In der Welt der Graphen nennt man einen solchen Weg einen Hamiltonkreis. Das Hamiltonkreis-Problem, also die Entscheidung darüber, ob es in einem Graph einen Hamiltonkreis gibt, ist eines der berühmten NP-vollständigen Probleme und damit im Allgemeinen eines der schwierigsten Probleme, die die Informatik kennt. Wenn der Graph so übersichtlich ist wie bei den Museumsplänen dieser Biberaufgabe, ist das Hamiltonkreis-Problem aber nicht so schwer zu lösen. Generell lässt sich die Informatik von einem schwierigen Problem nicht beeindrucken. Sie sucht dann unter anderem nach Spezialfällen, für die man das Problem doch gut lösen kann. Für das Hamiltonkreis-Problem sind einige bekannt.

https://de.m.wikipedia.org/wiki/Hamiltonkreisproblem

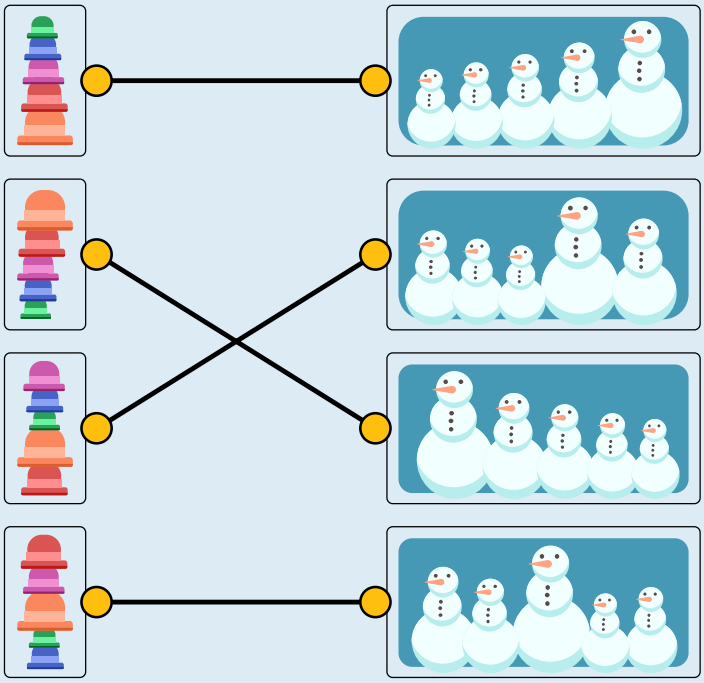
Schneemann-Hüte

Die Schneemänner bekommen Hüte. Sie stellen sich in vier Reihen an.
Jede Reihe sucht sich den passenden Hutstapel aus. Dann nimmt der erste Schneemann (links) den obersten Hut vom Stapel, der zweite den nächsten Hut – und so weiter.
Am Ende hat jeder den passenden Hut:
Der kleinste Schneemann hat den kleinsten Hut, der zweitkleinste Schneemann den zweitkleinsten Hut – und so weiter.

Verbinde jede Schneemann-Reihe mit dem passenden Hutstapel.



So ist es richtig:
Der oberste Hutstapel passt zur obersten Schneemannreihe. In der Reihe stehen die Schneemänner nach Größe geordnet, der kleinste zuerst.



Auf dem Stapel liegen die Hüte nach Größe geordnet, der kleinste zuoberst.
Der zweite Hutstapel passt zur dritten Schneemannreihe. In der Reihe stehen die Schneemänner nach Größe geordnet, der größte zuerst. Auf dem Stapel liegen die Hüte nach Größe geordnet, der größte zuoberst.
Der dritte Hutstapel passt zur zweiten Schneemannreihe.

Um das zu erkennen, ordnen wir Schneemännern und Hüten Nummern zu, der Größe nach: Der größte Schneemann bzw. der größte Hut bekommt die Nummer 5, der zweitgrößte Schneemann bzw. Hut bekommt Nummer 4, und so weiter.

Die Schneemänner in der zweiten Reihe haben dann von vorne die Nummern 3-2-1-5-4. Die Hüte auf dem dritten Stapel haben von oben die Nummern 3-2-1-5-4 – das passt.
Die Schneemänner in der vierten Reihe haben von vorne die Nummern 4-3-5-1-2. Dazu passt der vierte Hutstapel: Darin haben die Hüte von oben auch die Nummern 4-3-5-1-2.

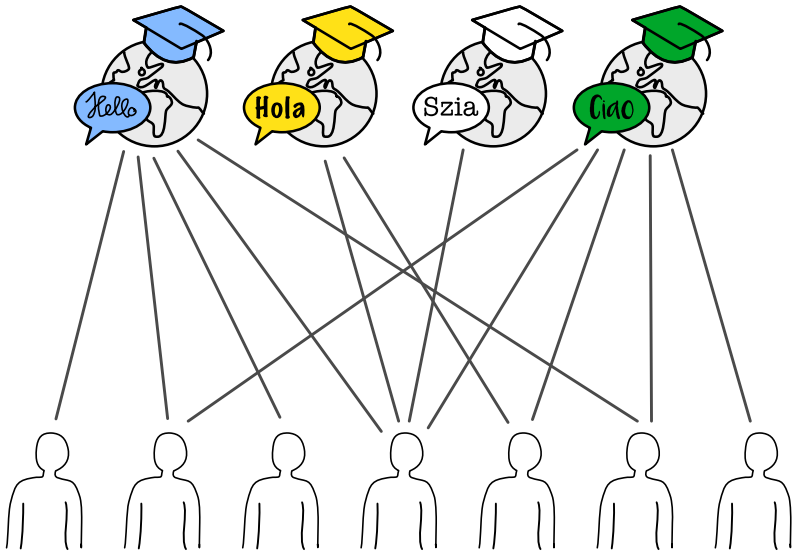
Das ist Informatik!

Kenner früherer Biberaufgaben vermuten vielleicht, was jetzt kommt. Allzu offensichtlich sind die Bezüge zu den Datenstrukturen Stapel und Reihe bzw. Schlange bei Hüten und Schneemännern. Doch die Erklärung der Lösung legt ein anderes, viel grundsätzlicheres Informatikthema nahe: die Abstraktion – die bei der Aufgabe „Schatzkarte“ schon kurz angesprochen wurde. Um bei den etwas komplizierteren Fällen erläutern zu können, welcher Hutstapel zu welcher Schneemann-Reihe passt, haben wir aufgehört, über Hüte und Schneemänner zu reden. Stattdessen haben wir Nummern verwendet, welche die Größe von sowohl Hüten als auch Schneemännern beschreiben. So konnten die Abfolgen der Größen jeweils aufgelistet und die Größen-Listen miteinander verglichen werden. Alle anderen Eigenschaften von Hüten (wie Farben oder Form) und Schneemännern (etwa die Anzahl der Knöpfe auf der mittleren Kugel) können in dieser Biberaufgabe ignoriert werden.

Die Darstellung von Hutstapeln und Schneemannreihen als Größenlisten ist eine Abstraktion oder Reduktion auf die für das gegebene Problem relevanten Informationen. Aber sollen Hüte und Schneemänner immer nur bezüglich ihrer Größe betrachtet werden? Vielleicht erfordert das nächste Hut-Schneemann-Problem ganz andere Informationen? Informatiksysteme arbeiten immer mit Abstraktion: mit einer Abbildung der Realität auf die Möglichkeiten binärer Repräsentation und algorithmischer Prozesse. Bei Abstraktion geht Information verloren. Das ist nicht grundsätzlich schlecht, aber jede Informatikerin und jeder Informatiker sollte sich dessen bewusst sein.

Sprachkurse

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

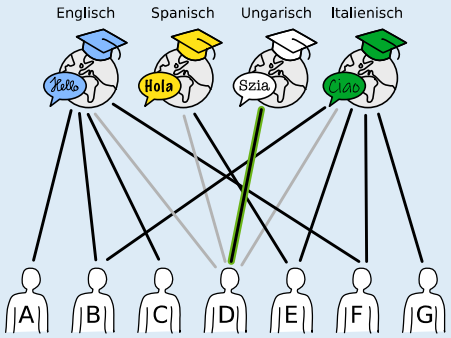


Eine Lehrperson kann nur einen Kurs halten. Trotzdem gibt es mehrere Möglichkeiten, jedem Kurs eine geeignete Lehrperson zuzuordnen.

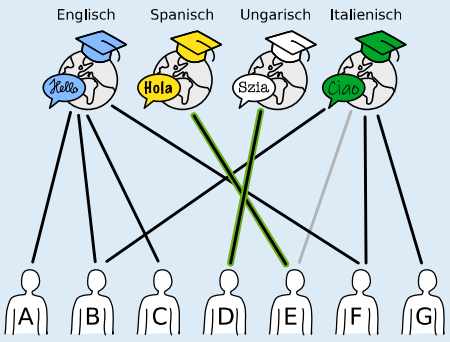
Ordne jedem Kurs eine geeignete Lehrperson zu.
Markiere dazu die Linie zwischen Person und Kurs.

Um die Antwort leichter erklären zu können, markieren wir die Lehrpersonen mit den Buchstaben A bis G. Die Sprachen, die in den Kursen unterrichtet werden, sind (von links nach rechts) Englisch, Spanisch, Ungarisch und Italienisch.

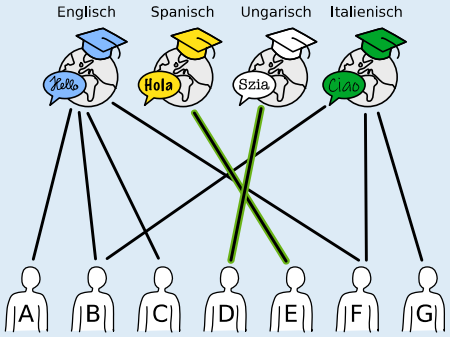
D ist die einzige Lehrperson, die für den Ungarischkurs geeignet ist. Sie muss diesem Kurs zugeordnet werden und kann keine weiteren Kurse übernehmen.



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



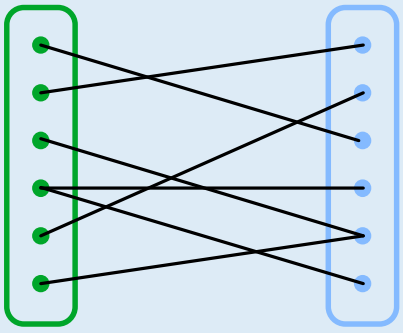
Bei den beiden verbleibenden Kursen (Englisch und Italienisch) kann man recht frei wählen. B und F dürfen aber nur einem Kurs zugeordnet werden, auch wenn sie für beide geeignet sind



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

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

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



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

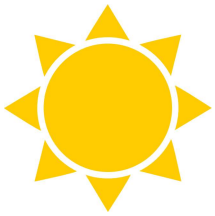
Zum Beispiel wird auch das sogenannte Heiratsproblem mithilfe eines bipartiten Graphen dargestellt. Dabei steht eine Menge von heiratswilligen Männern einer Menge von heiratswilligen Frauen gegenüber. Ziel des Verfahrens ist, unter Berücksichtigung der jeweiligen Wünsche, alle Männer beziehungsweise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

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

Stempelbild

Mika hat vier verschiedene Stempel.
Er nimmt jeden Stempel einmal in die Hand und stempelt damit zweimal.
So entsteht dieses Bild:

Welchen Stempel hat Mika zuerst genommen?

A)	B)	C)	D)
			

Antwort C ist richtig:
Im Bild kann man sehen, dass manche gestempelten Figuren über anderen Figuren sind. Eine Figur, die über einer anderen liegt, kann Mika nicht mit dem ersten Stempel gemacht haben.
Nun schauen wir uns das Bild genau an:


Ganz rechts ist das Haus über der Blume;
das Haus kann also nicht der erste Stempel sein.


Links daneben ist das Blatt über der Sonne;
auch das Blatt kann nicht der erste Stempel sein.


Weiter links ist die Blume über dem Blatt;
auch die Blume kann nicht der erste Stempel sein.

Nur die Sonne ist über keiner anderen Figur: Mika hat also die Sonne zuerst genommen.

Weil Mika jeden Stempel nur einmal nimmt, wissen wir auch die Reihenfolge der anderen Stempel: Nach der Sonne hat Mika das Blatt genommen, danach die Blume und zuletzt das Haus.

Stickmuster

Lana besitzt eine programmierbare Stickmaschine.

Die Maschine kann diese zwei Zeichen sticken: + oder X .
Außerdem kann sie aus diesen Zeichen ein drittes Zeichen zusammensetzen: ✳

Dazu muss der Stoff zwischen X und + (oder umgekehrt)
um ein Zeichen zurückgeschoben werden.
Lana programmiert die Stickmaschine mit diesen drei Tasten:

⊕	Die Stickmaschine stickt + .
⊗	Die Stickmaschine stickt X .
⊖	Der Stoff wird um ein Zeichen zurückgeschoben.

Die Stickmaschine führt ein Programm so oft hintereinander aus, wie Lana will.
So hat Lana dieses Muster

+✳X+✳X+✳X+✳X+✳X+✳X+✳X

mit diesem Programm erstellt:

⊕ ⊕ ⊖ ⊗ ⊗

Mit welchem Programm hat Lana nun dieses Muster erstellt?

XX✳X✳XX✳X✳XX✳X✳XX✳X

A ⊗ ⊗ ⊗ ⊖ ⊕ ⊗ ⊕ ⊖ ⊗ ⊗ ⊗

B ⊗ ⊗ ⊗ ⊖ ⊕ ⊗ ⊗

C ⊗ ⊗ ⊗ ⊖ ⊕ ⊗ ⊕ ⊖ ⊗

D ⊕ ⊖ ⊗ ⊕ ⊖ ⊗ ⊗ ⊕ ⊖ ⊗ ⊗

Antwort C ist richtig:

Das Muster wird durch eine (unbekannte) Anzahl von Ausführungen des gesuchten Programms erstellt. Es enthält also mehrfach hintereinander ein immer gleiches Teil-Muster, das durch einmalige Ausführung des Programms entsteht. Zunächst müssen wir also dieses sich wiederholende Teil-Muster erkennen; das ist dieses hier:

.

Welches Programm erzeugt nun dieses Muster bei einmaliger Ausführung? Wir können zum einen versuchen, aus dem Muster auf das Programm zu schließen. Wir wissen, dass das Zeichen X durch die Taste ⊗ gestickt wird. Das Zeichen ✳ wiederum kann durch zwei verschiedene Tastenfolgen gestickt werden, nämlich ⊗ ← ⊕ oder ⊕ ← ⊗. Das gesuchte Programm hat also die Form ⊗ ⊗ ? ⊗ ?, wobei an der Stelle der Fragezeichen je eine der beiden Tastenfolgen für ✳ stehen kann. Nur das Programm in Antwort C hat diese Form.

Zum anderen können wir für jedes Programm feststellen, welches Muster es bei einmaliger Ausführung erzeugt. Hier ist eine Tabelle; das obige Teil-Muster entsteht durch Programm C.

A	⊗ ⊗ ⊗ ← ⊕ ⊗ ⊕ ← ⊗ ⊗ ⊗	X X ✳ X ✳ X X
B	⊗ ⊗ ⊗ ← ⊕ ⊗ ⊗	X X ✳ X X
C	⊗ ⊗ ⊗ ← ⊕ ⊗ ⊕ ← ⊗	X X ✳ X ✳
D	⊕ ← ⊗ ⊕ ← ⊗ ⊗ ⊕ ← ⊗ ⊗	✳ ✳ X ✳ X

Das ist Informatik!

Die Stickmaschine kennt drei verschiedene Anweisungen. Sie kann eine beliebige, aber endliche Folge dieser Anweisungen ausführen und so ihre Muster erzeugen. Außerdem kann sie – auf Anstoß des Benutzers oder der Benutzerin – die Ausführung der Anweisungsfolge wiederholen. Die Anweisungsfolge wird als *Programm* bezeichnet.

Damit funktioniert die Stickmaschine ganz ähnlich wie ein Computer. Auch Computer kennen eine Reihe von Anweisungen, meist aber deutlich mehr als drei. Auch Computerprogramme bestehen aus Folgen dieser Anweisungen. Allerdings ist es für Menschen recht schwierig, mit Hilfe der Anweisungen, die ein Computer direkt ausführen kann (in der Informatik auch *Maschinenanweisungen* genannt), ein Programm zu schreiben. Deshalb gibt es Programmiersprachen mit anderen Anweisungen, die Menschen besser verstehen können. Damit ein Computer ein Programm ausführen kann, das in einer solchen höheren Programmiersprache geschrieben ist, müssen die Anweisungen der Programmiersprache in Maschinenanweisungen übersetzt werden.

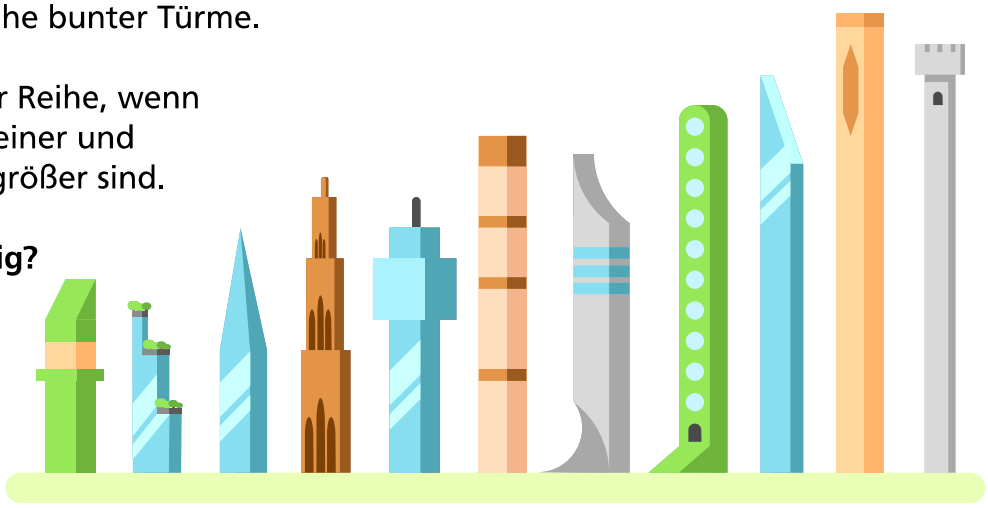
Im Gegensatz zu den Programmen der Stickmaschine gibt es in allen höheren Programmiersprachen für Computer nicht nur einfache Anweisungen und Anweisungsfolgen, sondern insbesondere *bedingte Anweisungen* und *Wiederholungsanweisungen.* Sie sorgen dafür, dass Teile des Programms abhängig von Bedingungen ausgeführt bzw. wiederholt ausgeführt werden. Erst diese besonderen Anweisungen verhelfen Programmiersprachen und damit auch Computern zu ihrer besonderen Leistungsfähigkeit.

Türme

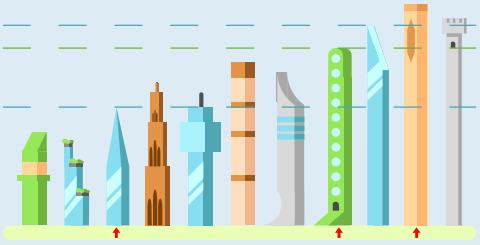
In Colortown steht eine Reihe bunter Türme.

Ein Turm steht richtig in der Reihe, wenn alle Türme links von ihm kleiner und alle Türme rechts von ihm größer sind.

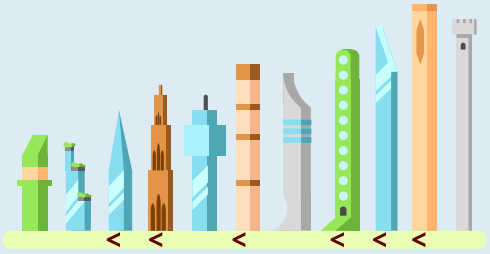
Welche Türme stehen richtig?

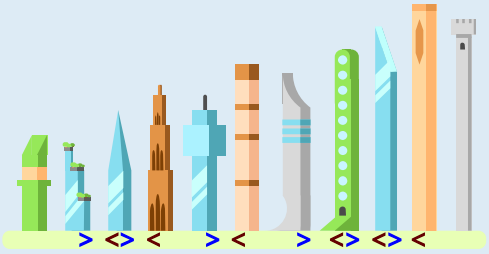


So ist es richtig:



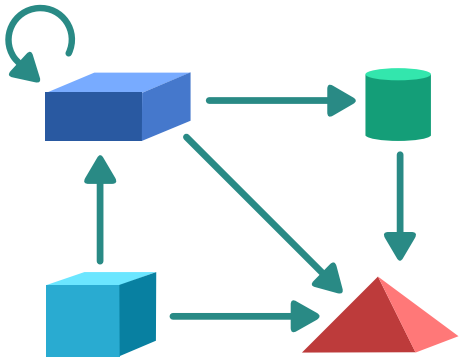
Drei Türme stehen richtig, weil jeweils alle Türme links von ihnen kleiner und alle Türme rechts von ihnen größer sind. Das kann man an den gestrichelten waagerechten Linien erkennen, die genau auf der Höhe der Türme verlaufen.
Ein Verfahren, mit dem man die „richtigen" Türme bestimmen kann, geht so: Zuerst geht man die Türme der Reihe nach durch und markiert einen Turm mit <, wenn alle Türme links von ihm kleiner sind. Dann geht man die Türme noch einmal durch und markiert einen Turm mit >, wenn alle Türme rechts von ihm größer sind. Alle Türme, die danach mit < > markiert sind, erfüllen beide Bedingungen und stehen richtig.





Das ist Informatik!
In dieser Biberaufgabe werden die Türme nach und nach anhand ihrer Höhe verglichen, um festzustellen, ob sie „richtig" stehen. Ganz ähnlich kann man vorgehen, wenn man dafür sorgen will, dass alle Türme richtig stehen. Dazu geht man ebenfalls die Türme durch und vergleicht die Höhen benachbarter Türme miteinander. Findet man dabei einen Turm, der nicht richtig steht, weil der Turm direkt links von ihm größer ist, vertauscht man die beiden Türme. Das wiederholt man so lange, bis alle Türme richtig stehen. Dann ist die Turmreihe der Höhe nach sortiert.
Dieses Sortierverfahren kennt die Informatik unter dem Namen „Bubble-Sort". Es funktioniert zuverlässig, aber nicht besonders schnell. Das macht sich besonders dann bemerkbar, wenn es um das Sortieren großer Datenmengen geht. Weil das Sortieren eine der wichtigsten Beschäftigungen von Computern ist, haben sich Informatikerinnen und Informatiker viele verschiedene und insbesondere schnelle Sortierverfahren überlegt. Sie werden unter anderem dadurch charakterisiert, wie viele Vergleiche beim Sortieren benötigt werden. Während Bubble Sort für 1.000 zu sortierende Elemente 1.000.000 Vergleichen benötigen kann, kommen schnellere Verfahren mit 10.000 Vergleichen aus. Bei 1.000.000. Elementen sind es schon 1.000.000.000.000 zu 20.000.000 Vergleichen. Auch wenn diese Zahlen nur grobe Annäherungen sind, ist der Unterschied jedenfalls groß.

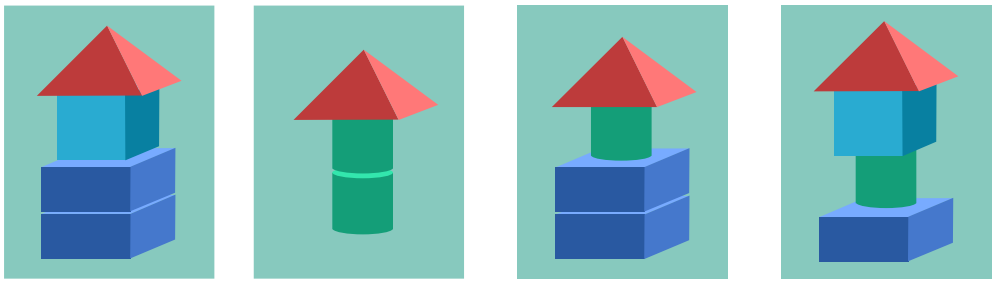
Julias Turm



In ihrem neuen Baukasten findet Julia vier Sorten Bauklötze. Sie erfindet Regeln, wie sie mit den Klötzen Türme bauen möchte. Das Bild zeigt Julias Regeln:
Julia setzt nur dann einen Klotz auf einen anderen, wenn ein Pfeil vom unteren zum oberen Klotz zeigt.

Ein Beispiel: Auf einen blauen Quader darf sie einen grünen Zylinder setzen, eine rote Pyramide, viele andere blaue Quader, aber keinen türkisen Würfel.
Wenn Julia einen Turm baut, fängt sie mit irgendeinem Klotz an und hält sich dann genau an ihre Regeln. Sie hört auf, wenn sie keine Lust mehr hat – oder wenn es nach den Regeln nicht mehr weitergeht.

Welchen Turm hat Julia gebaut?



A) **B)** **C)** **D)**

Antwort C ist richtig:
So baut Julia Turm C: Sie fängt mit einem blauen Quader an, setzt einen weiteren blauen Quader darauf, folgt dann dem Pfeil zum grünen Zylinder und abschließend dem Pfeil zur roten Pyramide. Damit ist Schluss, denn von der roten Pyramide zeigt kein Pfeil zu einem nächsten Klotz.
Die anderen Türme hat Julia nicht gebaut, denn sie verletzen die Regeln.
Turm A: Ein türkiser Würfel sitzt auf einem blauen Quader, obwohl es in den Regeln keinen Pfeil vom Quader zum Würfel gibt – nur umgekehrt.
Turm B: Ein grüner Zylinder sitzt auf einem anderen, obwohl es in den Regeln keinen "Kreispfeil" von Zylinder zu Zylinder gibt.
Turm D: Ein türkiser Würfel sitzt auf einem grünen Zylinder, obwohl es in den Regeln keinen Pfeil vom Zylinder zum Würfel gibt.

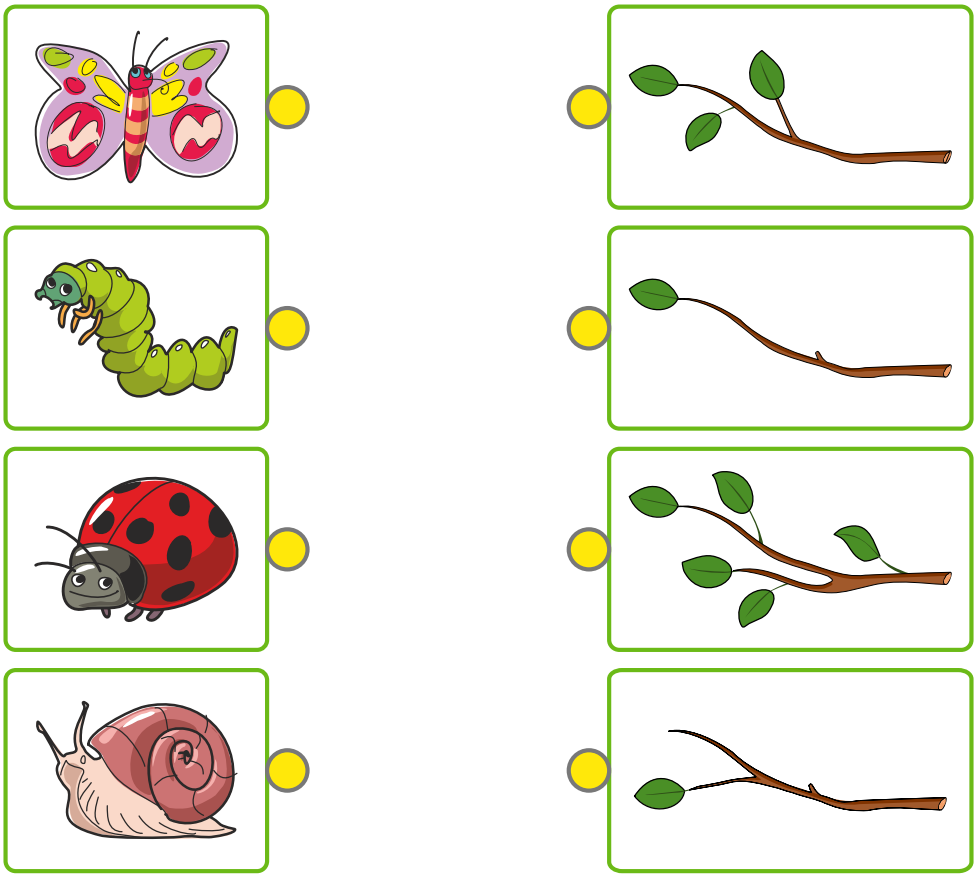
Das ist Informatik!
Julias Turmbau-Regeln orientieren sich daran, welcher Klotz gerade oben auf dem Turm sitzt. Der oberste Klotz ist also entscheidend für den aktuellen Zustand des Turms. Die Regeln legen dann die möglichen nächsten Zustände des Turms fest. Das Regelbild in dieser Biberaufgabe ist also ein Zustandsübergangsdiagramm für Bauklotz-Türme.
Zustandsübergangsdiagramme werden in der Informatik häufig verwendet. Sie eignen sich gut zur Beschreibung ganz unterschiedlicher Dinge: das Verhalten von Softwaremodulen oder auch ganzer Programme, einfache sprachliche Strukturen, das Zusammenspiel von Hardware-Bauteilen und vieles mehr. Mit Hilfe einer solchen formalen Beschreibung kann dann auch getestet werden: ob die Software sich wie gewünscht verhält – oder ob der Turm richtig gebaut ist.

Zweige

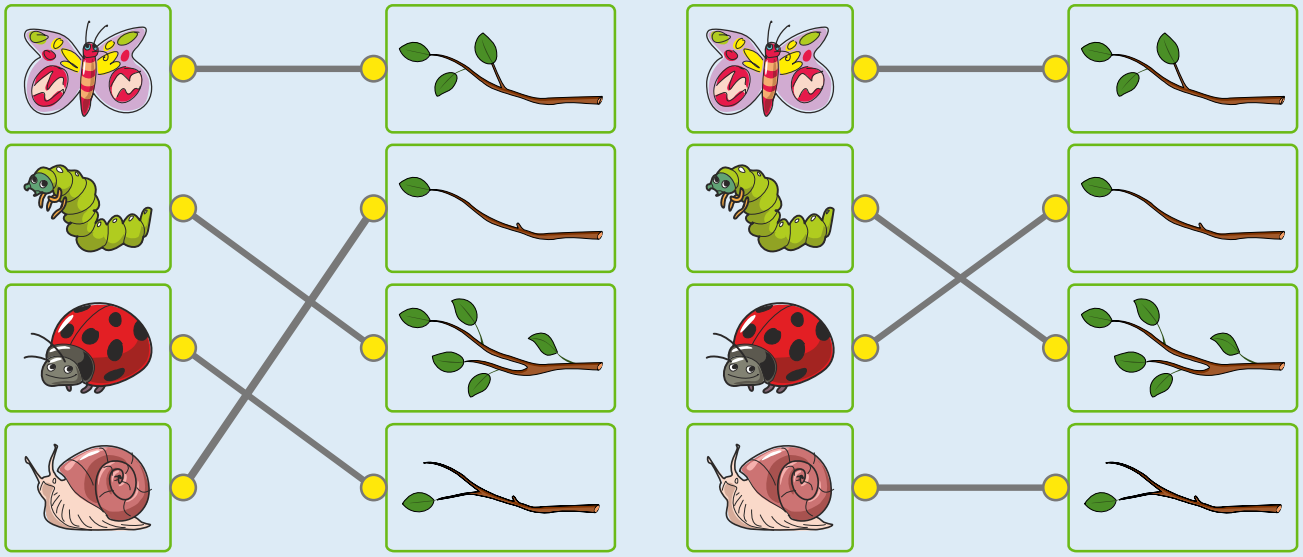
Emma und Felix entdecken in einem Busch vier Tiere.
Die Tiere sitzen auf vier verschiedenen Zweigen:

- Der Zweig mit der Raupe  hat die meisten Blätter.
- Der Zweig mit dem Schmetterling  hat mehr Blätter als der Zweig mit der Schnecke  .
- Der Zweig mit dem Marienkäfer  hat genau ein Blatt.

Verbinde jedes Tier mit seinem Zweig.



So ist es richtig:
Es gibt zwei richtige Antworten:



Der dritte Zweig von oben hat die meisten Blätter, nämlich fünf. Darauf sitzt die Raupe.
Der Marienkäfer sitzt auf einem Zweig mit einem Blatt. Das kann der zweite Zweig von oben sein, aber auch der untere Zweig. Es gibt also zwei richtige Lösungen.

Schmetterling und Schnecke sitzen auf den beiden anderen Zweigen. Der eine hat drei Blätter, und der andere hat ein Blatt. Die Schnecke sitzt auf dem zweiten Zweig mit einem Blatt. Dann sitzt der Schmetterling auf dem Zweig mit drei Blättern, also mit mehr Blättern als der Zweig mit der Schnecke.

Das ist Informatik!
Die richtige Zuordnung der Tiere zu den Zweigen hängt von der Anzahl der Blätter an den Zweigen ab. Eine besondere Rolle spielen Vergleiche zwischen diesen Blätteranzahlen, nämlich „mehr Blätter als“ und „die meisten Blätter“ (was so viel heißt wie „mehr Blätter als alle anderen“). Es wäre wohl einfacher gewesen, diese Biberaufgabe richtig zu beantworten, wenn die Zweige sortiert gewesen wären, nach der Anzahl der Blätter, von oben nach unten. Dann hätte man nämlich die Vergleichsergebnisse sehen können: Der Zweig mit den meisten Blättern wäre oben gewesen, und für einen Zweig oberhalb eines anderen wäre klar gewesen, dass er mehr Blätter hat als der andere.

Beim Umgang mit Daten, insbesondere mit vielen Daten, kann es sehr hilfreich sein, wenn die Daten sortiert sind. In der Kontaktliste auf dem Handy zum Beispiel sind die Daten alphabetisch sortiert; so ist es viel leichter, einen Kontakt zu finden, als wenn die Liste beliebig durcheinander wäre. Computer sind ständig mit dem Sortieren von Daten beschäftigt. Viele Informatikerinnen und Informatiker beschäftigen sich deshalb intensiv damit, wie Computer möglichst schnell auch große Datenmengen sortieren können.