Bebras Australia Computational Thinking Challenge

2021 Solutions Guide
Round 2

Secondary School
Grades 7–12

bebras.edu.au

Bebras Australia Computational Thinking Challenge

Bebras is an international initiative aiming to promote Computational Thinking skills among students.

Started in 2004 by Professor Valentina Dagiene from the University of Vilnius, 'Bebras' is Lithuanian for beaver. This refers to their collaborative nature and strong work ethic.

The International Bebras Committee meets annually to assess potential questions and share resources. Questions are submitted by member countries and undergo a vetting process.

The Bebras international community has now grown to 60 countries with over 2.9 million students participating worldwide!

Bebras Australia began in 2014 and is now administered through CSIRO Digital Careers.

In Australia, the Bebras Challenge takes place in March and August–September each year. As of 2020, two separate challenges are offered for each round.

To find out more and register for the next challenge, visit bebras.edu.au

Engaging young minds for Australia's digital future

523
Australian schools participated in Bebras R1 2021

32,311
Australian students participated in Bebras R1 2021

2.9 million
students participate worldwide

CSIRO Digital Careers supports teachers and encourages students' understanding of digital technologies and the foundational skills they require in an ever-changing workforce. Growing demand for digital skills isn't just limited to the ICT sector. All jobs of the future will require them, from marketing and multimedia through to agriculture, finance and health. Digital Careers prepares students with the knowledge and skills they need to thrive in the workforce of tomorrow.

digitalcareers.csiro.au

CSIRO digital careers

What is a Solutions Guide?

Computational Thinking skills underpin the careers of the future. Creating opportunities for students to engage in activities that utilise their critical and creative thinking along with problem solving skills is essential to further learning. The Bebras Challenge is an engaging way for students to learn and practice these skills.

Within this Solutions Guide you will find all of the questions and tasks from Round 1 of the Bebras Australia Computational Thinking Challenge 2021. On each page above the question you will find the age group, level of difficulty, country of origin and key Computational Thinking skills.

After each question you will find the answer, an explanation, the Computational Thinking skills most commonly used, and the Australian Digital Technologies curriculum key concepts featured.

Contents

What is Computational Thinking?

Computational Thinking is a set of skills that underpin learning within the Digital Technologies classroom. These skills allow students to engage with processes, techniques and digital systems to create improved solutions to address specific problems, opportunities or needs. Computational Thinking uses a number of skills, including:

DECOMPOSITION
Breaking down problems into smaller, easier parts.

PATTERN RECOGNITION
Using patterns in information to solve problems.

ABSTRACTION
Finding information that is useful and taking away any information that is unhelpful.

MODELLING AND SIMULATION
Trying out different solutions or tracing the path of information to solve problems.

ALGORITHMS
Creating a set of instructions for solving a problem or completing a task.

EVALUATION
Assessing a solution to a problem and using that information again on new problems.

More Computational Thinking resources

Visit digitalcareers.csiro.au/CTIA to download the Computational Thinking in Action worksheets. These can be used as discussion prompts, extension activities or a framework to build a class project.

Each resource was designed to develop teamwork; critical and creative thinking; problem solving; and Computational Thinking skills.

Computational Thinking skills alignment

2021 Round 2 Questions	Grade level	Decomposition	Pattern Recognition	Abstraction	Modelling & Simulation	Algorithms	Evaluation
Years 7+8							
Colourful Flags	Easy	■	■		■	■	■
Fake News	Easy	■	■	■	■	■	
Arranging Cubes	Easy	■	■		■	■	
Bus Schedule	Easy	■	■		■	■	■
Party Messages	Easy	■	■			■	■
At the Castle	Medium	■		■	■	■	■
Flower Box	Medium		■	■	■	■	■
Bakery on Wheels	Medium	■		■	■	■	■
Snakes and Ladders	Medium	■		■	■	■	■
Decide-uous Trees	Medium				■	■	■
Cakes and Neighbours	Hard	■		■		■	■
Household Appliances A	Hard	■		■	■	■	■
Beaverburg Delivery	Hard	■			■	■	■
Wizard Beaver's Alchemy	Hard	■	■	■		■	■
Car Factory	Hard	■	■	■	■		
Years 9+10							
Recover my Robot	Easy	■			■	■	■
Conveyor Belt	Easy	■		■		■	■
Permutations	Easy	■			■	■	■
Party Message A	Easy	■	■		■	■	
Glowing Panels	Easy	■	■		■		■
Echo Cypher	Medium		■	■		■	■
Robot Parking	Medium	■			■	■	■
Beaver Meetings	Medium	■			■	■	■
Traffic Lights	Medium	■			■	■	■
Squares	Medium	■		■	■	■	■
Robot Tree Planter	Hard	■	■	■	■	■	■
Secret Siblings	Hard	■	■			■	■
Lost Car	Hard	■		■	■	■	■
Key Points	Hard	■			■	■	■
Train Trip	Hard	■		■		■	■

Computational Thinking skills alignment

2021 Round 2 Questions	Grade level	Decomposition	Pattern Recognition	Abstraction	Modelling & Simulation	Algorithms	Evaluation
Years 11+12							
DNA Sequence	Easy	■				■	■
Dinner Table	Easy			■	■	■	■
Passwords B	Easy		■	■	■	■	■
Flag Semaphore	Easy	■	■	■		■	■
Interpret Programs	Easy	■	■			■	■
Mixed Results	Medium	■	■			■	■
Production Units	Medium	■			■	■	■
Three Workers	Medium	■			■	■	■
Decryption Map	Medium	■			■	■	■
Coin Collector	Medium	■			■	■	■
MathMachine B	Hard	■		■	■	■	■
Optimal Processing Flow	Hard	■		■	■	■	■
Squirrel Prison	Hard	■	■	■		■	■
Telegraph Networks	Hard	■			■	■	■
Flipping Cards B	Hard	■	■		■	■	■

Australian Digital Technologies curriculum key concepts

Abstraction

Hiding details of an idea, problem or solution that are not relevant, to focus on a manageable number of aspects.

Data Collection

Numerical, categorical, or structured values collected or calculated to create information, e.g. the Census.

Data Representation

How data is represented and structured symbolically for storage and communication, by people and in digital systems.

Data Interpretation

The process of extracting meaning from data. Methods include modelling, statistical analysis, and visualisation.

Specification

Defining a problem precisely and clearly, identifying the requirements, and breaking it down into manageable pieces.

Algorithms

The precise sequence of steps and decisions needed to solve a problem. They often involve iterative (repeated) processes.

Implementation

The automation of an algorithm, typically by writing a computer program (coding) or using appropriate software.

Digital Systems

A system that processes data in binary, made up of hardware, controlled by software, and connected to form networks.

Interactions

Human-Human Interactions: How users use digital systems to communicate and collaborate.
Human-Computer Interactions: How users experience and interface with digital systems.

Impact

Analysing and predicting how existing and created systems meet needs, affect people, and change society and the world.

For more information on the Digital Technologies curriculum, please visit the Australian Curriculum, Assessment and Reporting Authority (ACARA) website: australiancurriculum.edu.au/f-10-curriculum/technologies/digital-technologies

Digital Technologies key concepts alignment

2021 Round 2 Questions	Abstraction	Data Collection	Data Representation	Data Interpretation	Specification	Algorithms	Implementation	Digital Systems	Interactions	Impacts
Years 7+8										
Colourful Flags			■	■	■	■		■	■	
Fake News	■					■				
Arranging Cubes					■	■	■			
Bus Schedule				■	■	■		■		
Party Messages			■	■	■	■				
At the Castle	■			■	■	■				
Flower Box	■			■	■	■				
Bakery on Wheels	■			■	■	■		■	■	
Snakes and Ladders	■				■	■	■			
Decide-uous Trees				■	■	■	■			
Cakes and Neighbours	■				■	■				
Household Appliances A	■	■			■	■		■		
Beaverburg Delivery					■	■		■		
Wizard Beaver's Alchemy	■				■	■				
Car Factory	■		■	■	■	■		■		
Years 9+10										
Recover my Robot				■	■	■	■			
Conveyor Belt					■	■				
Permutations				■	■	■				
Party Message A		■	■	■	■	■		■		
Glowing Panels			■	■	■	■	■	■		
Echo Cypher	■		■	■	■	■		■		
Robot Parking					■	■				
Beaver Meetings				■	■	■		■	■	
Traffic Lights				■	■	■	■	■	■	
Squares	■			■	■	■				
Robot Tree Planter	■				■	■		■		
Secret Siblings					■	■		■		
Lost Car	■	■	■	■	■	■		■	■	■
Key Points			■	■	■	■	■	■	■	■
Train Trip	■			■	■	■				

Digital Technologies key concepts alignment

2021 Round 2 Questions	Abstrac-tion	Data Collection	Data Represen-tation	Data Interpre-tation	Specifica-tion	Algo-rithms	Imple-menta-tion	Digital Systems	Interac-tions	Impacts
Years 11+12										
DNA Sequence				■	■	■				
Dinner Table	■		■	■	■	■	■	■		
Passwords B	■	■	■	■	■	■		■		
Flag Semaphore	■			■	■	■				
Interpret Programs				■	■	■				
Mixed Results	■			■	■	■				
Production Units					■	■		■	■	
Three Workers					■	■				
Decryption Map				■	■	■		■		
Coin Collector					■	■	■	■	■	
MathMachine B	■			■	■	■	■	■		
Optimal Processing Flow	■		■	■	■	■	■	■	■	
Squirrel Prison	■		■	■	■	■		■		
Telegraph Networks			■	■	■	■	■	■	■	■
Flipping Cards B			■	■	■	■	■	■		

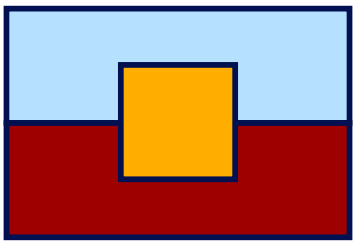
Bebras Challenge 2021 Round 2

Years 7+8

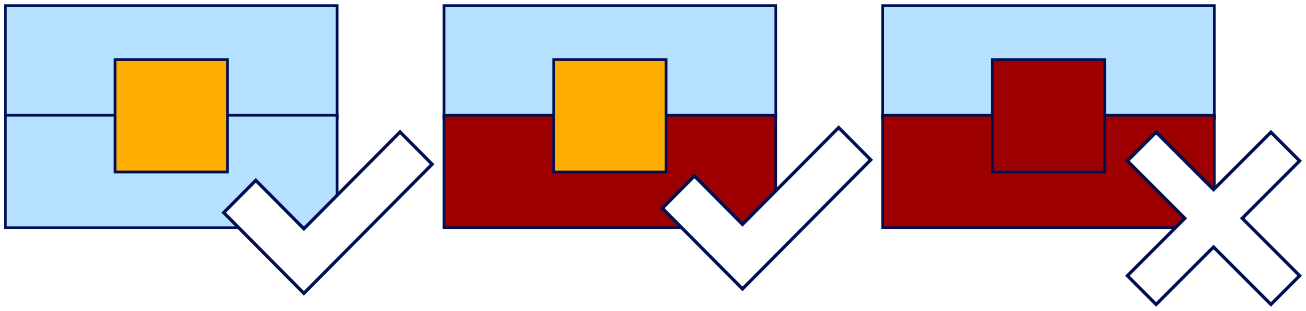
Colourful Flags

The Bebras shipyard builds excellent boats and every beaver would like to own a boat built by them. But there is a problem: how do you recognise your boat if all the boats look alike?

To address this, the beavers decide to put flags on all the boats. The flags have the following design:



The beavers agree on three possible colour options for the three areas of the flag design: red, yellow, and blue. While the two stripes may be of the same colour, the square in the centre must be a different colour to the two stripes.

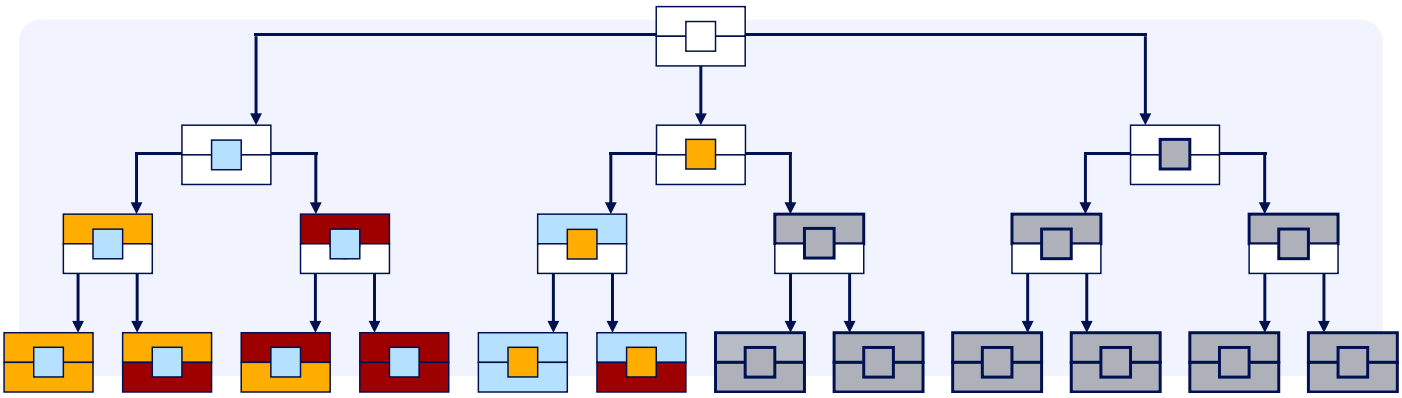


The beavers started to create a diagram which they use to show all possible colour combinations for the flags. Unfortunately they did not finish and some of the flags are not completely coloured in yet.

Question

Help the beavers complete the diagram by clicking on the missing fields (in grey) to assign a colour to them.

Note: there are multiple solutions, you only need to find one.



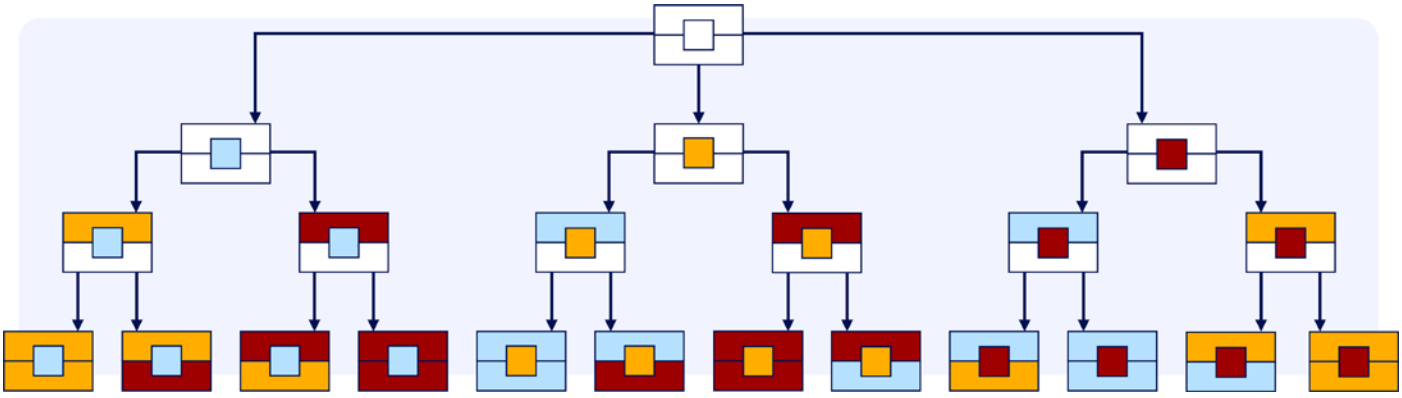
Continued on next page

Colourful Flags – continued

Answer

The answer is: there are multiple solutions, you only need to find any one solution.

Here is one example of one solution:



It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Data Representation, Data Interpretation, Specification, Algorithms, Digital Systems, Interactions

We all face different tasks in our lives: some are easy to solve, others require more time. One of the ways to solve difficult tasks is to list all possible options and find the best solution. That's why it is crucial for computer scientists to know how to effectively write out all the options.

This enumeration must be systematic (so that no object or option gets lost or repeated) and clearly laid out (so that it's easily understood by other people where each option must be placed). This is necessary for checking if all objects or options are found, and it also enables the process to be automated by programming.

To perform such operations you need to use appropriate data structures with a specific layout. If we have some features (parts of the flag) that can have different values (colours), we numerate these features in such a way that the values of the features are limited just by those with a smaller number (stripes depend on the colour of the square, so the square is first, then we numerate stripes with numbers 2 and 3). If we can do this, we usually use branching to represent all options. Trees have a structure that can represent nodes as we move from the trunk to the end of each branch. In each node we move in different paths depending on what is the value of the following feature. This procedure helps us to list all possible values in a structured way.

Fake News

Beaverland has a national broadcasting company that produces evening news bulletins. Most of the news items are true, but some are fake.

Each day, four beavers – Ahmed, Bert, Lotte and DaHye – watch the evening news together.

- Ahmed is perfect at recognising whether news items are real or fake.
- Bert thinks that all news items are fake.
- Lotte always thinks that all the true news items are fake, and the fake items are true.
- DaHye thinks that every news item is true.

They have agreed that in order to decide whether a news item is true, at least three of them must think (or know) that it is true.

Question

When will the beavers agree that a news item is true?

Select one of the following options.

When the news item is actually true

When the news item is actually fake

Always

Never

Continued on next page

Fake News – continued

Answer

The answer is: D) Never

This can be established by drawing a table like this one:

	Ahmed says	Bert says	Lotte says	DaHye says
If the news item is true:	True	Fake	Fake	True
If the news item is fake:	Fake	Fake	True	True

In any case, only two of the beavers will state that a news item is true. Since you need at least three beavers to say that it is 'True', that means they will never agree.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Abstraction, Modelling & Simulation, Algorithms

Concepts: Abstraction, Algorithms

Computers are good at doing exactly what is asked of them. They prefer a clear yes/no input in order to make correct decisions. However, there are situations where the information to be processed is not guaranteed to be correct, for example if a light sensor is dirty and did not register the sun had risen. Still we don't want all the lights in our house to be on all day.

For many applications, a computer program must be able to handle contradictory inputs – if we have three light sensors and two of them say it is light outside, while one says that it is still dark, what do we want the computer to do?

One way to solve this problem is to trust what the majority of the inputs say, or as per the task, to only trust the inputs if a large enough number of them agree.

Interestingly, on the Space Shuttle there were four identical computers running exactly the same programs to make decisions. More than six times every second it was checked to see whether the four computers still agreed and were doing exactly the same thing. If one of them did something different from the others (three times in a row) an alarm was raised and the computer was taken out of service and replaced by a backup.

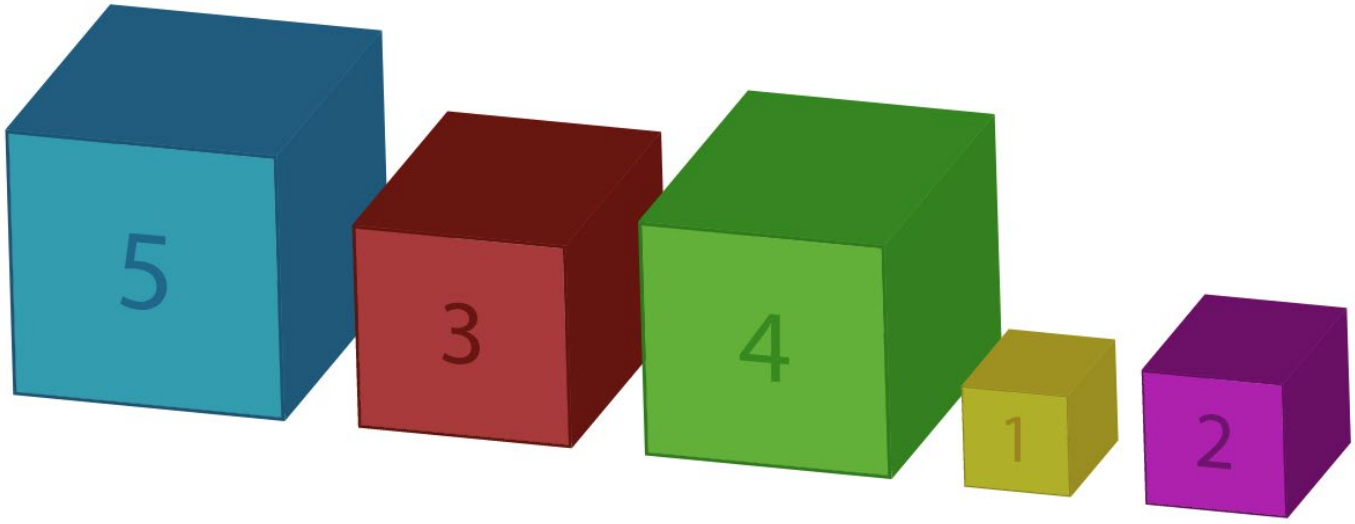
Arranging Cubes

Rebecca has several cubes of different heights that she wants to place in ascending order to form a staircase.

On a face of each cube its height is written. When arranging the cubes, the smallest one must be on the left.

Rebecca starts to arrange the cubes from left to right. She looks at all the cubes, and if a smaller cube is placed to the right of a bigger cube, she switches their positions.

When she reaches the two last boxes (on the right-hand side), she starts over from the left.



Question

What is the number of switches she has to make?

Select one of the following options.

6 7 8 9

Continued on next page

This question comes from Romania

Arranging Cubes – continued

Answer

The answer is: C) 8

On the first pass she swaps cubes 5 and 3, then 5 and 4, then 5 and 1 and finally 5 and 2. So the result should be: 3-4-1-2-5.

Start	53412
Swap one	35412
Swap two	34512
Swap three	34152
Swap four	34125

On the second pass, she switches 4 and 1; 4 and 2, so the result should be: 3-1-2-4-5.

Start	34125
Swap five	31425
Swap six	31245

On the third pass, she switches 3 and 1 and then 3 and 2 and the final result is: 1-2-3-4-5.

Start	31245
Swap seven	13245
Swap eight	12345

It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Modelling & Simulation, Algorithms

Concepts: Specification, Algorithms, Implementation

This is an example of a Bubble sort. It is a simple sorting algorithm, which works by methodically switching adjacent elements of a list if they are in the wrong order.

This is one of the many possible sorting algorithms.

Bus Schedule

The following tables show when the orange bus, Route 1 and the blue bus, Route 2 will arrive at each stop.

	Route 1		
Bus stop A	10:00	11:00	12:00
Bus stop B	10:20	11:20	12:20
Bus stop C	10:40	11:40	12:40
Bus stop D	11:00	12:00	13:00
Bus stop E	11:20	12:20	13:20

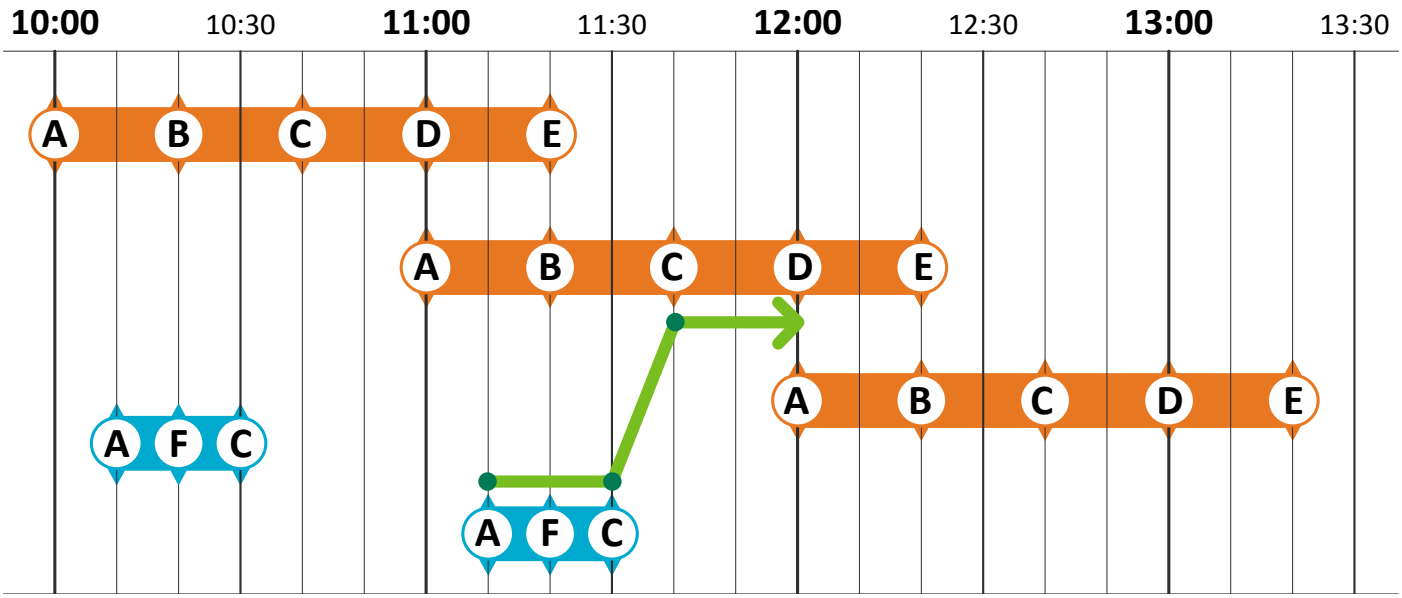
	Route 2	
Bus stop A	10:10	11:10
Bus stop F	10:20	11:20
Bus stop C	10:30	11:30

Question

If James is at Stop A at 11:05, what is the earliest time that he can reach Stop D?

Answer

The answer is: 12:00



Beaver James will:

1. Take the Route 2 blue bus from stop A at 11:05 to stop C at 11:30
2. Take the Route 1 orange bus from stop C at 11:40 to stop D at 12:00

If James had taken the Route 1 orange bus from stop A, he would not have arrived until 13:00.

Continued on next page

Bus Schedule – continued

It's Computational Thinking

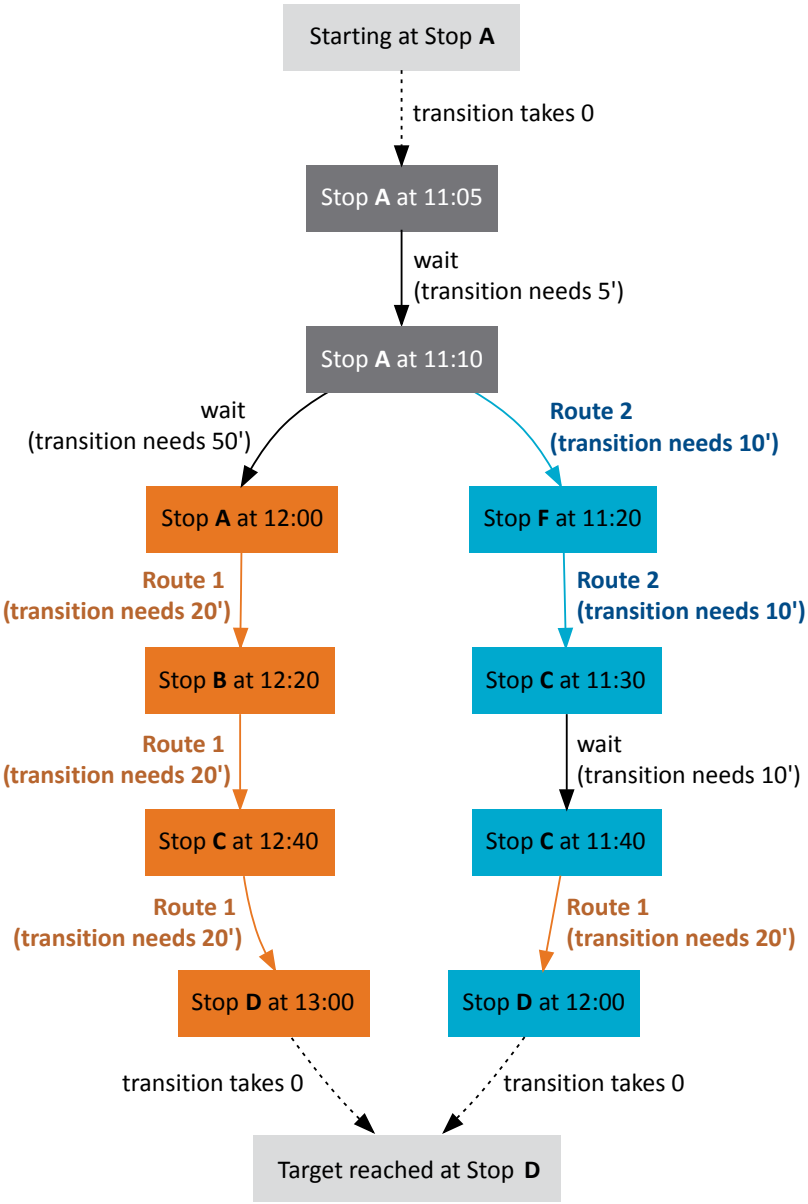
Computational Thinking Skills: Decomposition, Pattern Recognition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Data Interpretation, Specification, Algorithms, Digital Systems

A common approach in informatics is modelling a problem as a graph: a diagram made using boxes and arrows.

Here it is useful to use boxes to record the time of the day and the bus stop James is at; an arrow from one box to another can be used to record the travel time or transitions between the stops.

This is the resulting graph:



The problem now can be formulated as a *shortest path problem*, as it is known in graph theory. Finding a path between two vertices (the boxes) in a graph such that the sum of the waits (the transition times) of its constituent edges (the arrows) is minimised.

From the ellipse box at the top to the ellipse to the bottom one can find only 2 paths: the left path has a total wait time of 0+5+50+20+20+20+0=115 minutes whereas the right path is 0+5+10+10+10+20+0=55 minutes thus the shortest time is the right path.

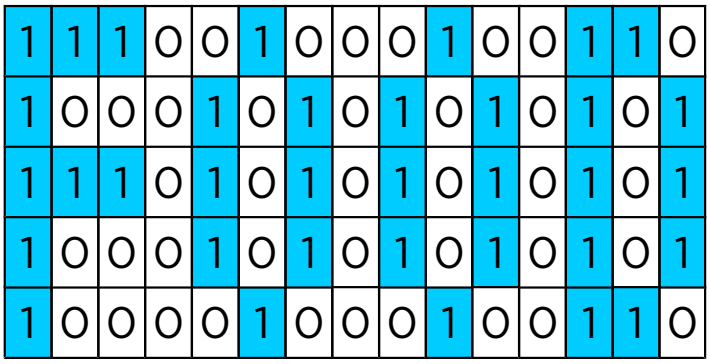
Party Messages

Three friends have a secret language for sending messages to each other. Each of them has written down what they will bring to the party on a piece of paper using their secret language.

Emily and Olivia's messages have already been revealed as shown in the picture below:

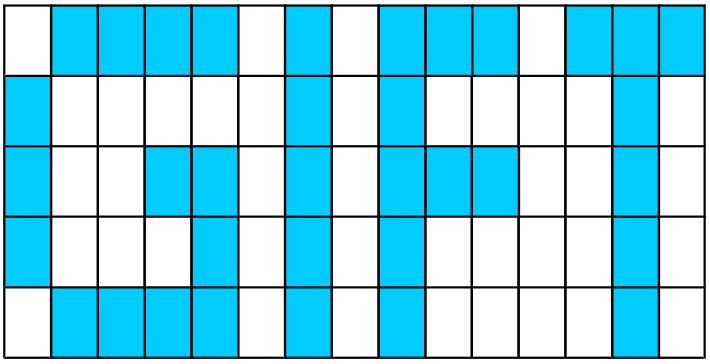
Emily's message

1:32131221
1:1311111111111
1:3111111111111
1:1311111111111
1:14131221



Olivia's message

0:14111313
1:15111411
1:1221113211
1:1311111411
0:141111411



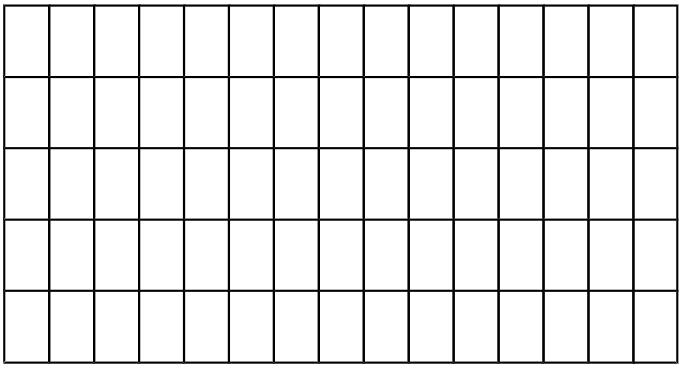
Question

Jack's secret message is shown below. What will Jack bring to the party?

Jack's message

0:122123121
1:1311111111111
1:133122111
1:1311111111111
0:1211111111121

?



Select one of the following options.

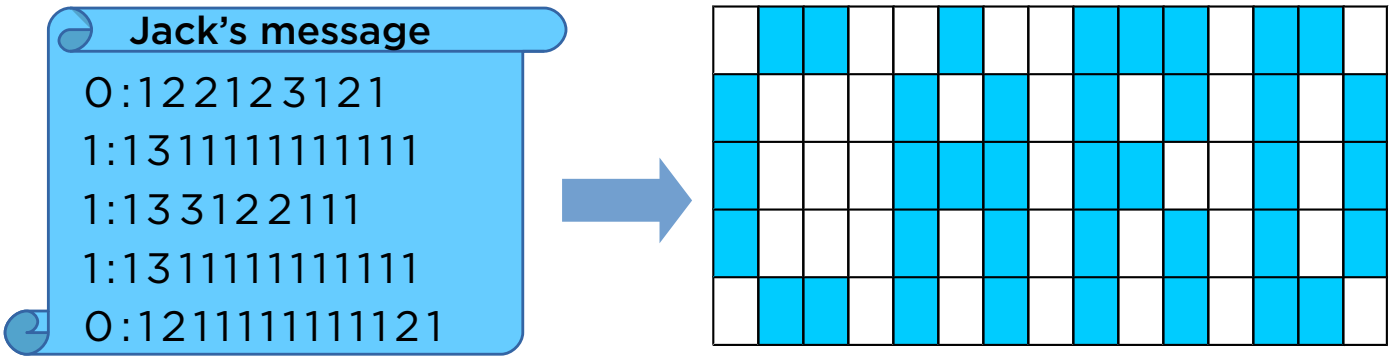
Game **Cola** **Card** **Cake**

Continued on next page

Party Messages – continued

Answer

The answer is: c) CARD



The first digit on each line of the code tells us the colour of the squares that line starts with.

0 means 'white' and 1 means 'blue'.

The numbers on each line of the code after the colon (:) tell us how many squares of each alternating colour will follow.

The first line of Jack's message begins with a white square (0). The first digit after the colon shows the number of squares for the starting color (which is one for 'white'). Then we have two blue squares that are followed by two white, one blue and so on.

The first row of the code makes following decoded colour pattern:

0 : 1 2 2 1 2 3 1 2 1
first square white : 1w 2b 2w 1b 2w 3b 1w 2b 1w

The second row of the code makes following decoded colour pattern:

1 : 1 3 1 1 1 1 1 1 1 1 1 1 1
first square blue : 1b 3w 1b 1w 1b 1w 1b 1w 1b 1w 1b 1w 1b

This task can be solved if these steps are repeated for the remaining three lines.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Algorithms, Evaluation

Concepts: Data Representation, Data Interpretation, Specification, Algorithms

This task shows an example of run-length encoding which is a form of lossless data compression. The task also includes concepts such as algorithms, data representation, digital images, picture elements (pixels), and cryptography.

The participant has to come up with an algorithm to represent the lines of code in picture format by looking at the lines of example code. Participants then apply this algorithm to solve the remaining lines of code themselves.

This task also deals with data representation by introducing a way of representing a binary image. A digital image that has been changed into a sequence of numbers that computers can understand. A pixel (short for picture element) is a tiny square of colour. Lots of these pixels together can form a digital image.

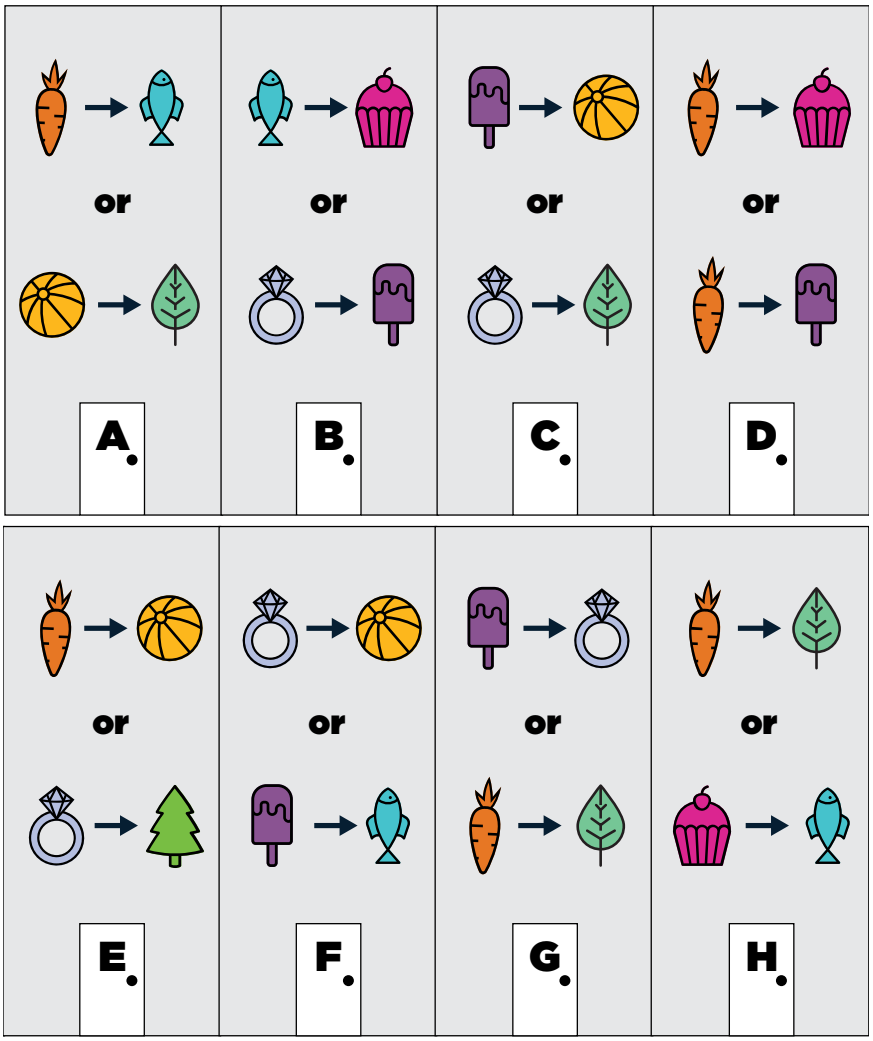
At the Castle

A smart beaver needs a fir tree  to build a dam on a river. But he only has one carrot  .

At the Castle there is a trade today and exchange is possible.

The beaver goes there with his carrot  and hopes to exchange it for a fir tree  .

In each room two exchanges are allowed according to the following picture:



Question

What is the sequence of rooms the beaver has to go in, to ensure that he will eventually get a fir tree ?

Select one of the following options.

DGE | GGE | AGE | DBC

Continued on next page

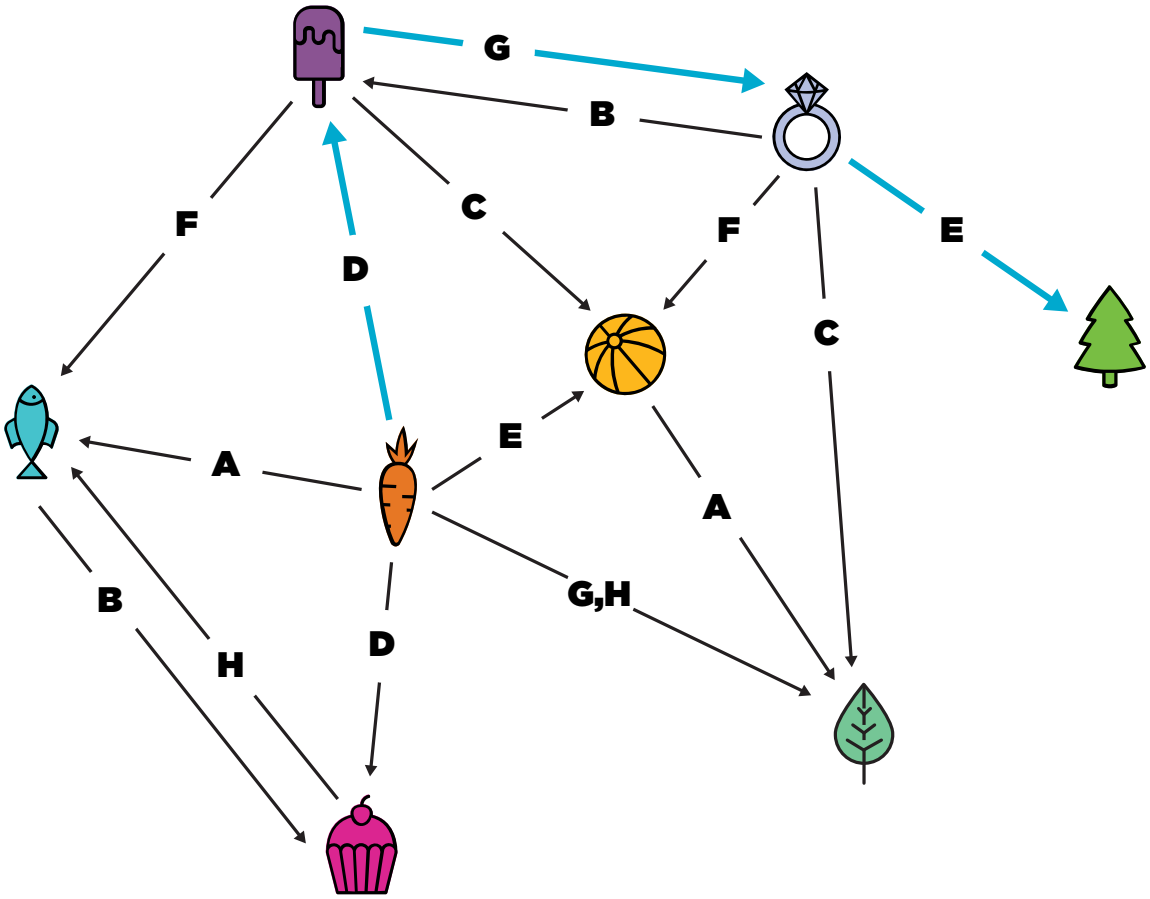
At the Castle – continued

Answer

The answer is: a) DGE

In Room D, the beaver exchanges his carrot for an ice block. After that he moves to Room G to exchange the ice block for a ring. Finally, the beaver goes to Room E to exchange the ring for the fir tree.

You can visualise the trading with the following directional graph. Each vertice (item) is connected with an edge (a line) marked with the room where the item can be traded in the direction of the arrow.



It's Computational Thinking

Computational Thinking Skills: Decomposition, Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Data Interpretation, Specification, Algorithms

The known states of such a problem can be represented with a *graph*. In Computer Science, a directed graph is used to visualise known task conditions, to demonstrate finite state machines.

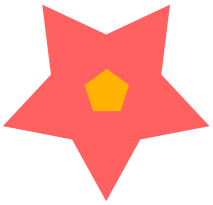
Posing such a problem as a graph-based problem is advantageous as a whole series of existing, known, graph-based strategies can be used.

Searching through the graph to find the solution can be done systematically by a sequence of steps, an algorithm. One algorithm that could be used here is called *depth-first search*. This algorithm would start from the carrot and explore as far as possible along each vertex (representing an object) until it finds the fir tree. Using the depth-first search algorithm, be careful not to fall into an endless cycle.

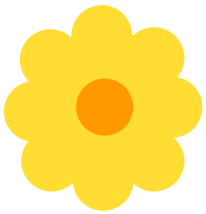
This question comes from Germany

Flowerbox

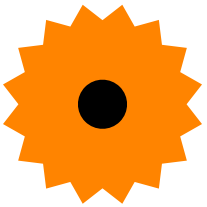
Ibbi has a new flowerbox, which can hold an arrangement of 3x3 flowers. He has three types of flowers:



Pink flowers



Yellow flowers



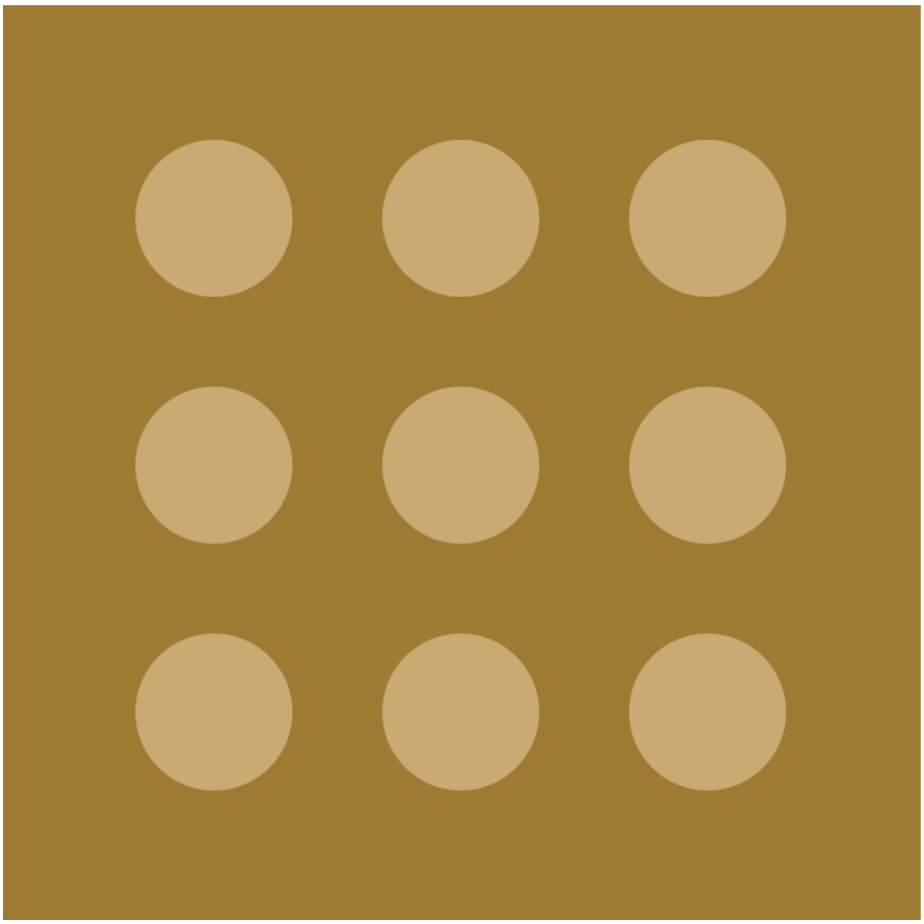
Orange flowers

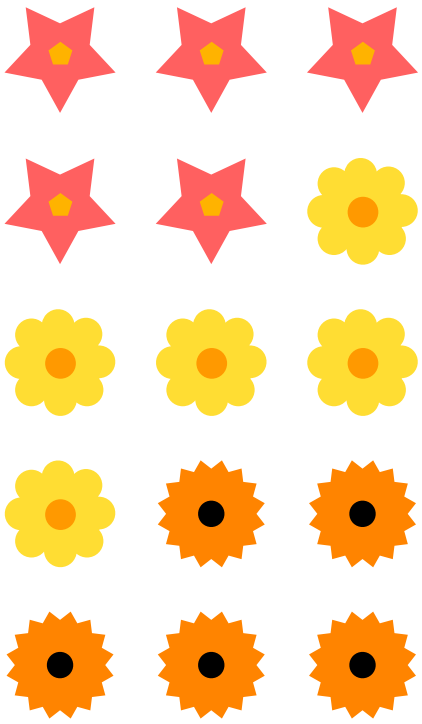
Ibbi wants the perfect arrangement for his flowers. He scores flower arrangements using this system:

- If a pink flower is next to a yellow flower, Ibbi adds 3 points to the score.
- If a yellow flower is next to an orange flower, Ibbi adds 1 point to the score.
- In all other cases, Ibbi does not add points to the score.
- All three types of flowers must appear in the arrangement.

Question

Drag the flowers into the spaces below to create a flowerbox arrangement with the highest possible score.





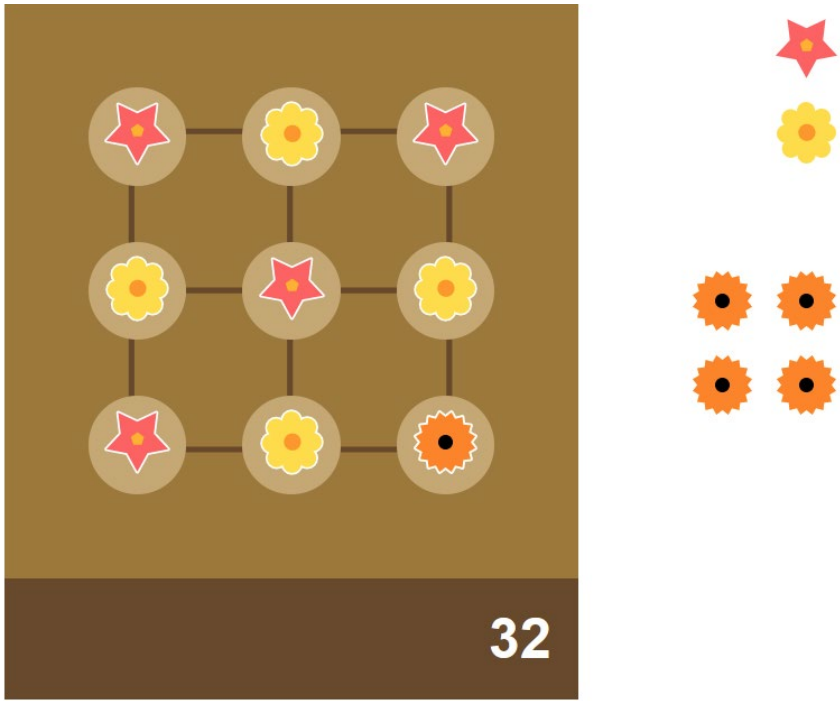
Continued on next page

Flowerbox – continued

Answer

The answer is: 32

An arrangement that scores 32 is the highest possible score. Ibbi assigns points to pairs of flowers that are next to each other in the flowerbox arrangement. In the image, these points are shown as the lines connecting the 'next-to-pairs'. The flowerbox arrangement with the highest possible score should have as many next-to-pairs of pink and yellow flowers as possible. However, at least one orange flower must appear. In order to help with the score, this orange flower must be part of next-to-pairs with yellow flowers – but as few as possible.



Therefore, the only orange flower must be put into a corner, where it will only be part of two next-to-pairs. It does not matter which corner to begin with; so there are four correct solutions. In all other positions, the orange flower would be next to more than two other flowers. In order to achieve at least a few points, yellow flowers must be put next to the orange flower. From then on, put pink and yellow flowers next to each other in order to achieve the highest score.

See one example of this process with the orange flower in the lower right corner:

It's Computational Thinking

Computational Thinking Skills: Pattern Recognition, Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Data Interpretation, Specification, Algorithms

It's human to find not just any solution to a problem, but the best. This is called optimisation, and it has always driven human behaviour. Optimisation is a part of everyday life. If you want to buy a specific computer, you are looking for the shop that demands the lowest price. If you must travel from A to B, you do want to take the least expensive route. But expensive in what terms: travel costs, distance, or time taken?

In optimisation, you not only need to know what a solution looks like, but also a precise function is required that can express the cost or the value of each solution. Then, the solution with a minimal or maximal value (that depends on the function) is the optimum choice. In this Bebras task, Ibbi's method of computing a score for flower arrangements is the optimisation function.

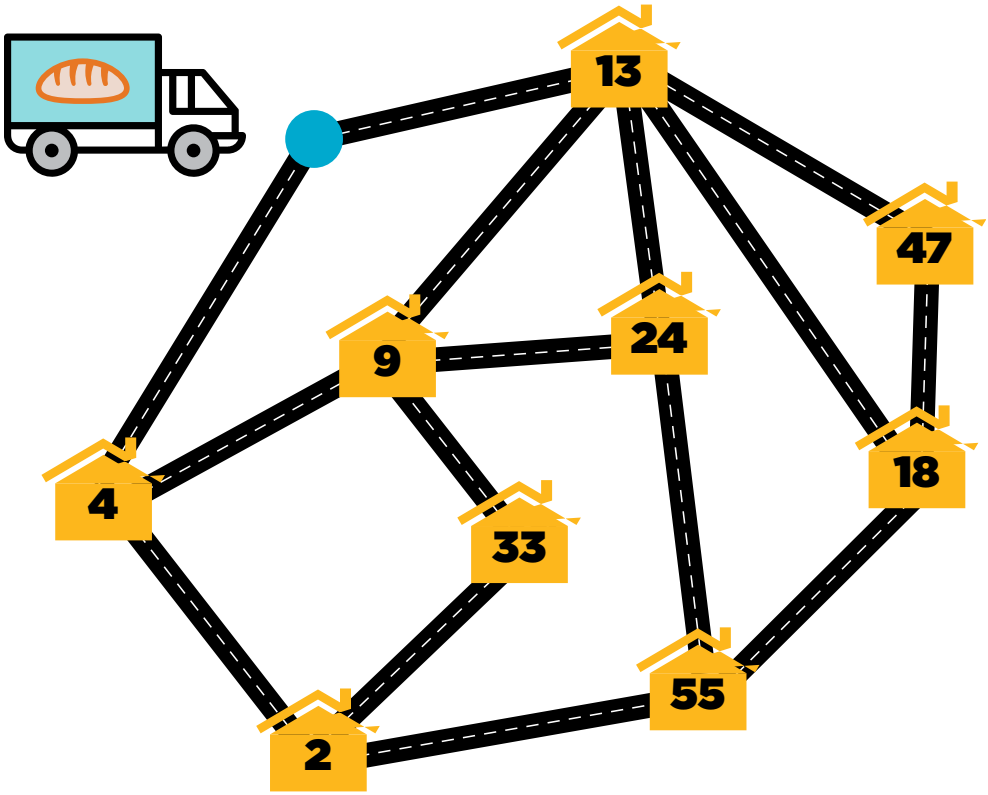
Mathematics has investigated optimisation for centuries. In informatics, much care has been taken with implementing mathematical algorithms, as computers can be used to optimise large scale problems where humans would not succeed. Computer scientists have discovered that many optimisation problems are hard to solve, and so for such problems they invented methods (heuristics) that will not always find the optimum, but solutions as close to the optimum as possible.

This question comes from Slovenia

Bakery on Wheels

Beaver Bartosz is an employee of the Bakery on Wheels. He is responsible for delivering bread to customers' houses.

He wants to travel as little as possible. Therefore, Bartosz decides to look at the map of the customers' houses and find a route that will take him to each house only once, and will not double back through the same street more than once. He must also start and end his journey at the blue dot on the map.



Question

In which order should Bartosz deliver the bread to the houses?

Select from one of the following options.

A) 13,47,18,55,24,9,33,2,4

B) 4,9,33,2,55,18,47,13,9,4

C) 13,9,24,47,18,55,2,33,4

D) 4,2,55,18,47,13,24,9,4

Answer

The answer is: A) 13, 47, 18, 55, 24, 9, 33, 2, 4

In B) 4, 9, 33, 2, 55, 18, 47, 13, 9, 4 there is no 24 and the 9 and 4 repeat twice.

In C) 13, 9, 24, 47, 18, 55, 2, 33, 4 there is no road between 24 and 47 or 33 and 4.

In D) 4, 2, 55, 18, 47, 13, 24, 9, 4 there is no 33 and 4 repeats twice.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Data Interpretation, Specification, Algorithms, Digital Systems, Interactions

In Computer Science we often meet problems related to graph theory, as graphs are useful for representing many different types of data. In this task, houses represent the vertices in the graph, and the streets represent the edges. Here, we are searching for a Hamiltonian graph, that is, to find a path visiting all the vertices only once, while starting and finishing in the same position.

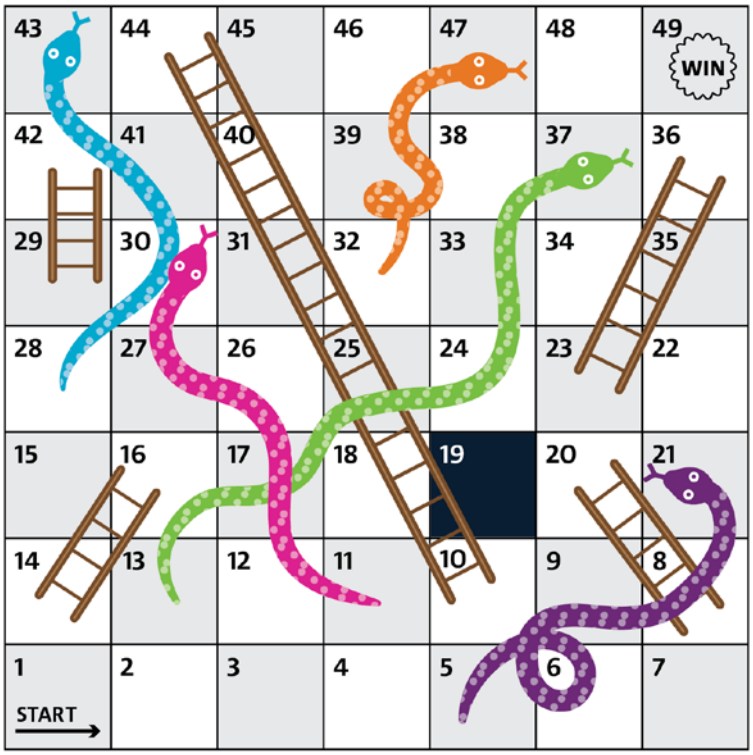
Snakes and Ladders

The game of Snakes and Ladders is played by rolling a single die (with values 1 to 6) and moving that many places from your current square.

Each player starts from square 1, and a player wins when they reach the last square, 49.

If you land on a square with a snake's head, you slide down to the snake's tail.

If you land on a square at the bottom of a ladder, you will climb up to the top of that ladder.



Example

If you land on square 21, the snake will take you back down to square 5.

If you land on square 23, the ladder will take you up to square 36.

Question

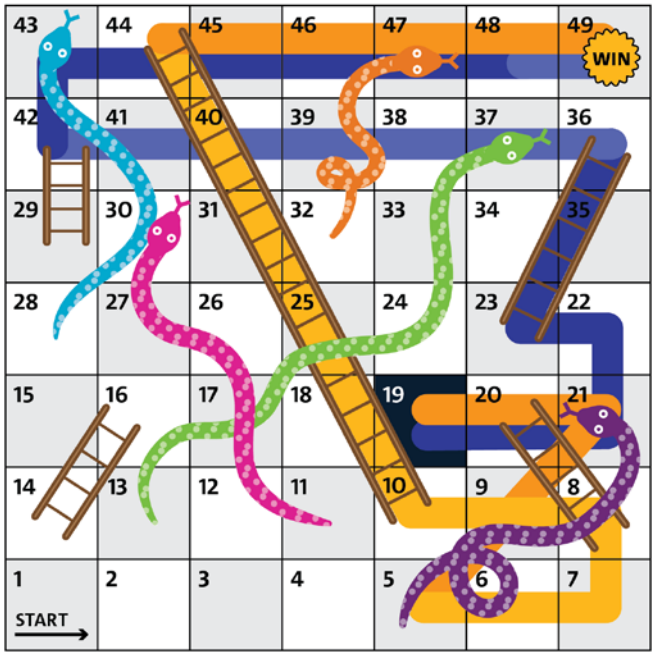
If you are on square 19, what is the minimum number of dice rolls you need to win?

Select from one of the following options.

2 **3** **4** **5**

Answer

The answer is: B) 3



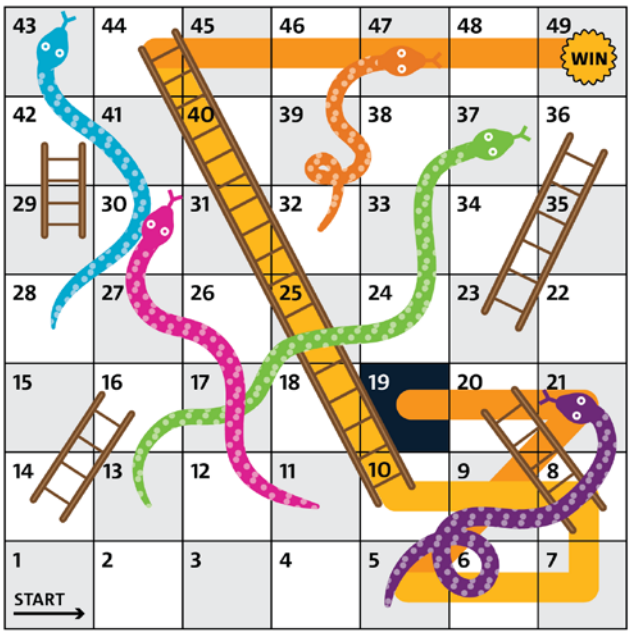
The shortest path only moving forward takes a minimum of 4 rolls of the dice.

From cell 19 you roll a four and move to cell 23, where the ladder takes you up to cell 36.

From here there are no remaining ladders and 13 cells remaining, so 3 rolls (6,6,1) are required for a total of 4 rolls.

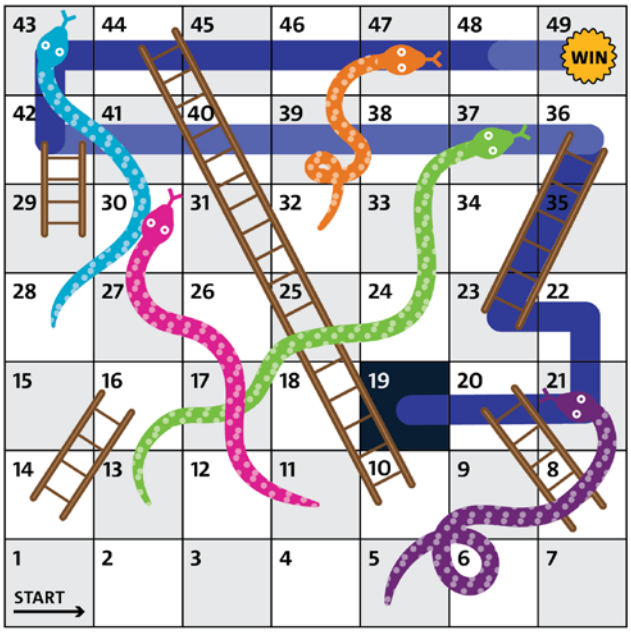
Continued on next page

Snakes and Ladders – continued



However, there is a shorter path if you move backwards and use the longest ladder from cell 10 to cell 44.

From cell 19, if you roll a two you would move to cell 21 and the snake would take you back to cell 5. From cell 5, if you roll a five, you will move to cell 10 where the ladder will take you up to cell 44. Finally, a roll of five will take you to the winning cell, for a total of 3 rolls.



It is impossible to win in 2 rolls, as from 1 roll, the cells you would land on could be 20,5,22,36,24, or 25 (from a roll of 1 to 6, respectively).

From any of these cells it is impossible to land on the final cell, 49, within 1 roll of the dice.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Specification, Algorithms, Implementation

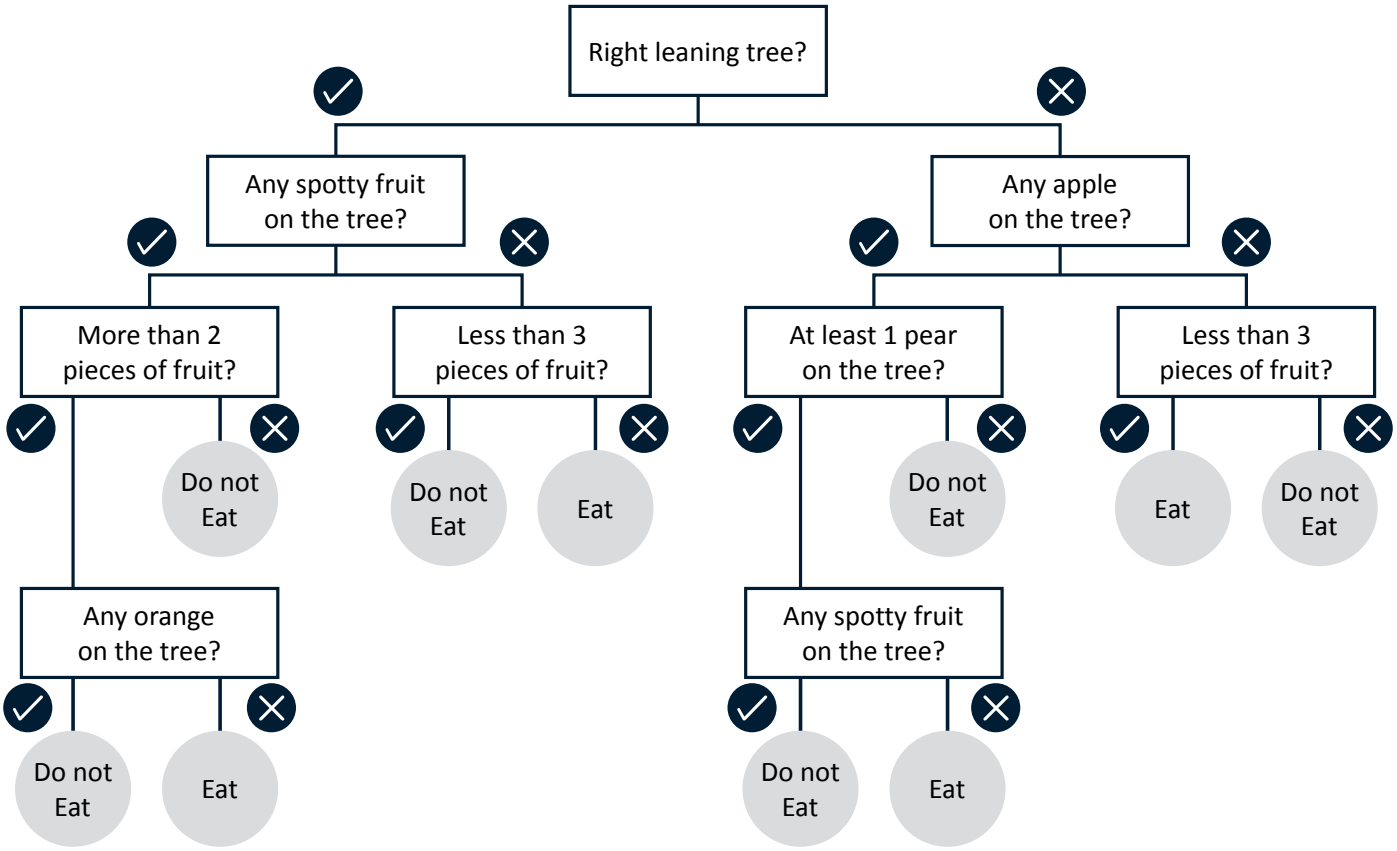
This task is a shortest path problem. A shortest path problem can be represented by graphs. Graphs are an abstract data-structure where nodes (cells in the game) are connected by edges. The cells here are said to have an edge if they can be reached from one to the other within a single throw of the dice. The problem then becomes one of finding the path with least number of edges, which may lead to the destination.

Breadth-first search (BFS) is an algorithm for searching graph data structures. It starts at the tree root and explores all of the neighbour nodes at the present depth prior to moving on to the nodes at the next depth level. This approach would be starting at cell 19, and exploring each dice roll before moving on to the second roll.

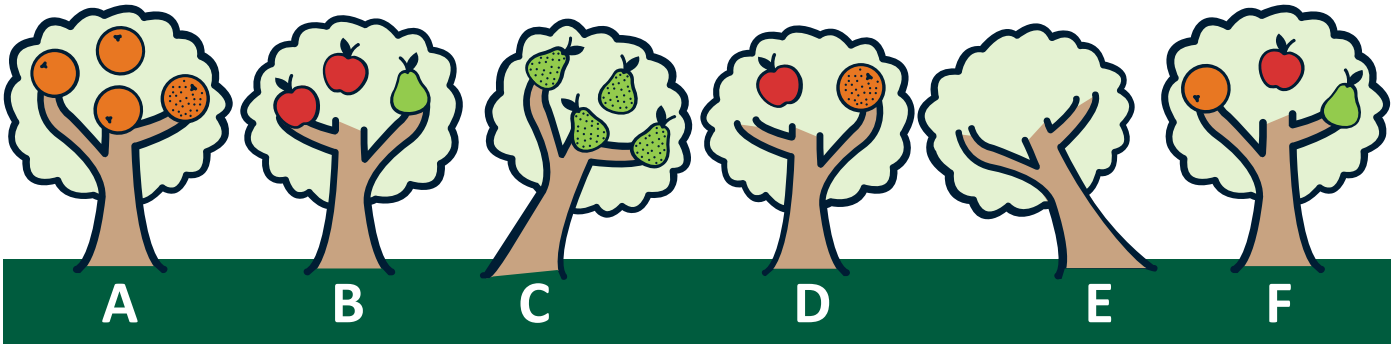
This question comes from Ireland

Decide-uous Trees

A hungry, but picky beaver wants to eat from some fruit trees. He will not eat from every tree. Instead, he uses the flowchart below to help him choose from which trees to eat.



These are the trees the beaver sees.



Question

From which trees will the beaver eat?

Select from one of the following options.

BCEF | ABCF | BCDE | BCDEF

Continued on next page

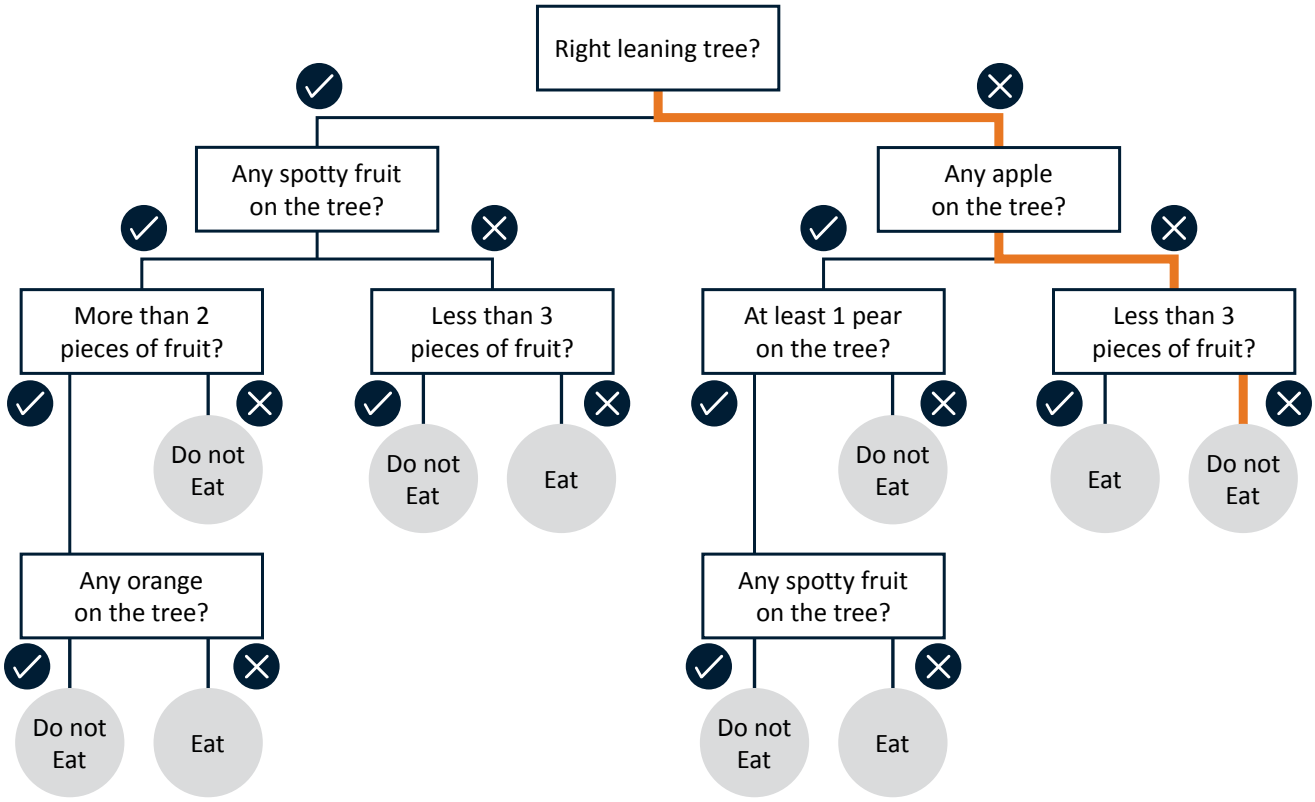
Decide-uous Trees – continued

Answer

The answer is: A) BCEF

The answer is reached by looking at each tree in turn and deciding whether the beaver should eat from it or not. The decision is reached by answering the questions in the rectangular text boxes (nodes) in the flowchart starting from the top until the final solution in a circle is reached. For each question, there are exactly two answer options, (1) *true* or (2) *false*. The chosen answer option leads either to another question in a rectangular text box (node) or to the final solution in a circle.

For example, tree A is not leaning to the right, doesn't have apples on it and has more than three fruits, so the beaver won't eat from it.



It's Computational Thinking

Computational Thinking Skills: Modelling & Simulation, Algorithms, Evaluation

Concepts: Data Interpretation, Specification, Algorithms, Implementation

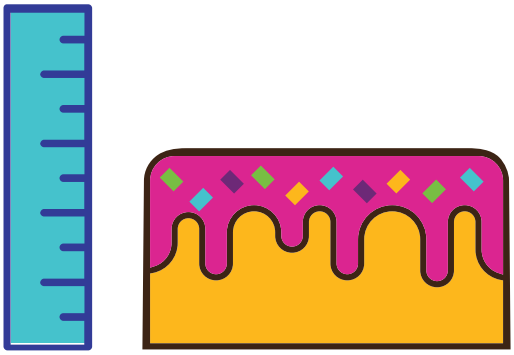
Flowcharts and decision trees are tools used for reaching decisions in cases of competing alternatives. They are modeled on the structure of trees in order to represent decisions and their possible outcomes. Decision trees usually have the following elements: root decision nodes (the uppermost, initial questions), branches (lines connecting all nodes), internal decision nodes (all questions following the initial question, all nodes with one incoming and two outgoing branches), and leaf nodes (final solutions, and terminal nodes with no outgoing branch).

The question in each decision node allows for exactly two possible answers: *true* or *false*. These are represented by two outgoing branches. Decision trees are a popular tool for classification and machine learning. They allow the creation of a decision (*true/false*) algorithm from an initial training set. Such algorithms can be trained on suitable data and then make predictions when presented with new data. These types of algorithms are used in many areas, for example, disease diagnosis and financial services. Decision trees have the useful property that, once generated by an algorithm, they can be easily understood.

Cakes and Neighbours

On Friday morning three neighbours, Niamh, Maryam and Clara, go to the baker to order a cake for a neighbourhood party on Saturday.

All three order the same type of cake – a cake that is 3cm tall.



However, on Friday afternoon, they all make (secret) phone calls to the baker to change their order. Each time the baker writes down the new order and throws away the old.

The cakes will be baked on Saturday in the early hours.

Niamh calls: Make my cake 1cm taller than what you have written down for me.
Later, Niamh calls again: Make my cake the same height as what you have written down for Maryam.
Maryam calls: Make my cake 2cm taller than what you have written down for me.
Later Maryam calls again: Make my cake 1cm less tall than what you have written down for me.
Clara calls: Make my cake 1cm taller than what you have written down for Niamh.
Later Clara calls again: Make my cake 1cm taller than what you have written down for me.

Important note:

We do not know at what time precisely these phone calls were made, and also not in what order – except that the second call each neighbour makes must comes after the first call by that same person.

Question

Only one of the following statements is certain to be correct on Saturday, whatever the order the phone calls were made in. Which one?

Select one of the following options.

Niamh's and Maryam's cakes are the same height.

Maryam's cake is at least 1cm shorter than Clara's cake

Clara's cake is exactly 2cm taller than Niamh's cake.

All three cakes are at least 4cm tall.

Continued on next page

Cakes and Neighbours – continued

Answer

The answer is: B) Maryam's cake is at least 1cm shorter than Clara's cake.

- Answer A) *Niamh's and Maryam's cakes are the same height* is not correct.
- If Maryam happens to make all her calls after Niamh, then Niamh will end up with a cake of 3cm, and Maryam with one of 4cm.
- This also proves that answer D) *All three cakes are at least 4cm* tall is not correct.
- Answer C) *Clara's cake is exactly 2cm taller than Niamh's cak*e is not correct.

If Clara makes all her calls first, and then Maryam and then Niamh, Clara's cake will be 5cm tall, Maryam's cake 4cm and Niamh's also 4cm.

If you look at Maryam's instructions to the baker, you see that at some point in the afternoon the baker will have written down a height of 3cm, a height of 5cm and a height of 4cm for Maryam's cake.

As a consequence, for Niamh's cake the only heights the baker could have written down are also 3cm, 4cm or 5cm.

This means that on Saturday, Clara's cake will be 5cm, 6cm or 7cm tall. Maryam's cake will always be 4cm tall. So the answer B) *Maryam's cake is at least 1cm shorter than Clara's cake* is correct.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Abstraction, Algorithms, Evaluation

Concepts: Abstraction, Specification, Algorithms

A computer can do many different things at the same time, even within the same application. For example, in a word processor the spellchecker works while you are typing.

As the story shows, when different tasks are executed at the same time, it is not always easy to predict what the eventual outcome will be. This is because it is often difficult to predict when exactly each part of each task will be executed.

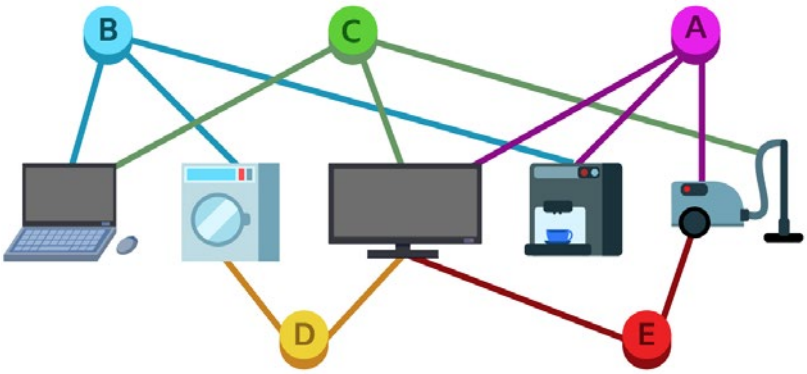
Writing programs that consist of tasks that are run at the same time is called *concurrent programming*. Programmers need to be aware of the problems involved in this kind of programming and write their programs in such a way that they are sure that certain actions will always be executed in the same order. For example, programmers who create software for a network printer have to ensure that two documents that are sent to the printer at (approximately) the same time, still are printed one after the other. The software must ensure that pages are not haphazardly intermingled or worse that some lines from the first document are printed then some lines from the second are printed on the same page until both documents are finished.

Nowadays, computers are specifically built to be able to do many things at the same time, concurrent programming, and the related techniques to avoid doing it incorrectly, has gained a lot of importance recently as part of the Computer Science curriculum.

This question comes from Japan

Household Appliances

In B-taro's house, there are five household appliances (a laptop, washing machine, TV, coffee machine, and vacuum cleaner), and five power outlets (A, B, C, D, and E) to control these appliances.



You can change the on/off state of the appliances by hitting the switches on the power outlets. However, the switches are designed to be inconvenient. As the outlets are connected to multiple appliances, each switch changes the on/off state of multiple appliances at the same time.

- Switch A is connected to TV, coffee machine, and vacuum cleaner.
- Switch B is connected to PC, washing machine, and coffee machine.
- Switch C is connected to PC, TV, and vacuum cleaner.
- Switch D is connected to washing machine and TV.
- Switch E is connected to TV and vacuum cleaner.

Question

Which is the correct sequence of switches to use to turn on only the TV and coffee machine?

Select one of the following options.

E,C,B,A **C,B,A,D** **D,A,E,C** **B,D,C,E**

Answer

The answer is: D) B, D, C, E

When you press Switch B, the laptop, washing machine, and coffee machine are turned on.

Next, when you press Switch D, the washing machine is turned off, and the TV is turned on.

Then, when you press Switch C, the laptop and TV are turned off, and the vacuum cleaner is turned on.

And lastly, when you press Switch E, the TV is turned on and the vacuum cleaner is turned off.

Continued on next page

Household Appliances – continued

At this point, the TV and coffee machine are on, whereas all the other appliances are now off.

A simple way to see that this is the correct answer while the others are not is to look at each switch and appliance independently. Then we see that if a power outlet's switch is pressed an odd number of times, the connected appliances are left in the "on" state, while an even number of presses leaves the appliances in the "off" state. The same holds true for a single appliance: if the number of switches pressed and connected to it is even, then the appliance will be turned off, but if it is odd, the appliance will be turned on.

This gives the following analysis:

- For the first sequence A) E, C, B, A we see for example that the vacuum cleaner is connected to three of the pressed switches (E, C, A). Hence, the vacuum cleaner will remain "on", which was not desired.
- For the second sequence B) C, B, A, D we see that the coffee machine is only connected to two of the pressed switches (B, A), which means that it will remain "off", again not desired.
- Finally, for the third sequence C) D, A, E, C we have for example the washing machine being connected to only one of the switches (D), which means that it will stay "on", which was not desired.
- In fact, in this case all appliances except the TV will be on.

The same reasoning holds of course also for the correct solution D) B, D, C, E where we have the following number of presses for the appliances:

- Laptop: two (B, C) => off
- Washing machine: two (B, D) => off
- TV: three (D, C, E) => on
- Coffee machine: one (B) => on
- Vacuum cleaner: two (C, E) => off

It's Computational Thinking

Computational Thinking Skills: Decomposition, Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Data Collection, Specification, Algorithms, Digital Systems

Think of the appliances in this task in terms of individual digits that get incremented each time a button connected to them is pressed. However, in this case the digits only count up to one and then start again from zero.

This is actually the binary system used by computers, where all information is stored and manipulated using only two digits: 0 and 1. For example, a computer performs addition by adding each binary digit at a time and carrying a unit to upper digits every time the result is over 1, as in the table below:

A = 0 1	A = 0 0	A = 0 1
B = 0 1	B = 1 1	B = 1 0
A + B = 1 0	A + B = 1 1	A + B = 1 1

In this task, however, we were only concerned with addition on individual digits (gnoring any carry over), where the digits represent the appliances' status (on = 1, off = 0). So each press of a switch would add one to the current state (0 or 1) of an appliance:

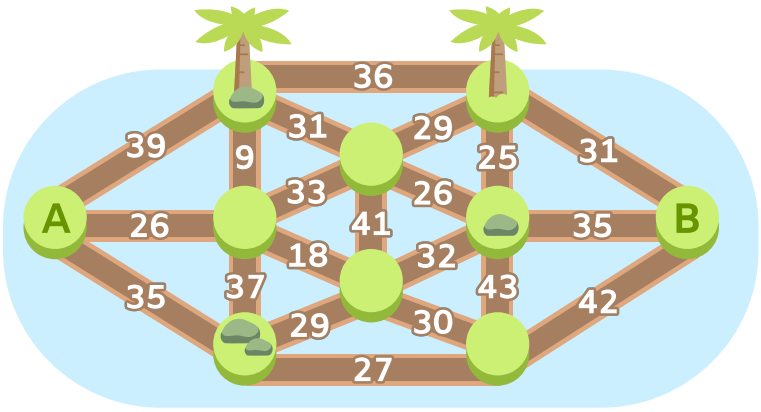
0 + 1 = 1

1 + 1 = 0

Beaverburg Delivery

Beaverland consists of a number of islands connected by bridges. Each bridge has a gross mass limitation and vehicles exceeding that weight can't cross it.

The map of the Beaverland islands below has islands represented as circles and bridges as segments. Each bridge has its gross mass limitation written on it.



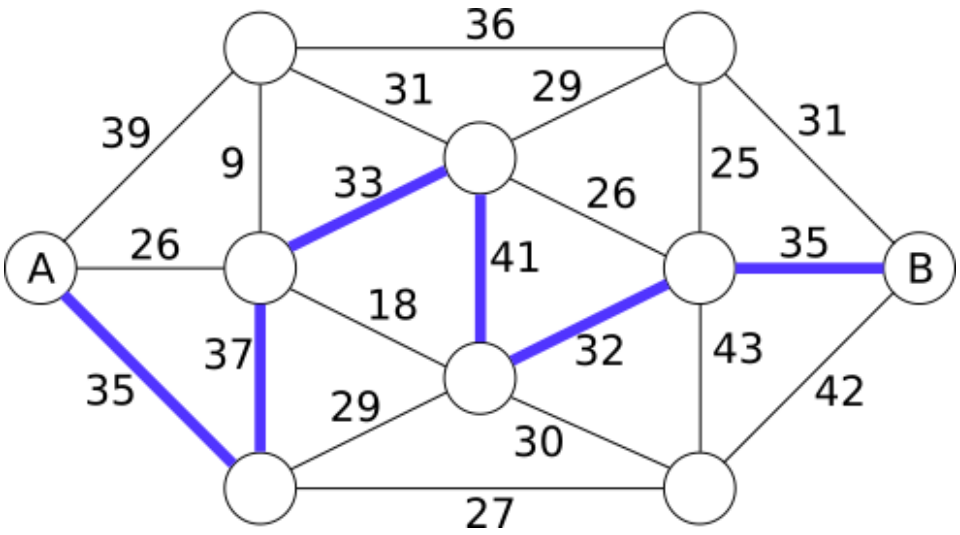
Question

Beaverland Delivery needs to move sand from Island A to Island B.

Which route should the truck take so that it can move the largest amount of sand?

Answer

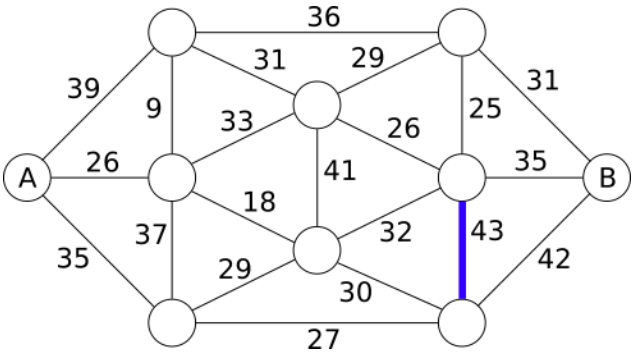
The answer is:



The maximum gross mass of a truck that could go from Island A to Island B is 32 tonnes, taking the above route.

To compute the answer, one approach is to try a heavy weight, see which bridges could be used, then reduce the weight until enough bridges allow passage to reach Island B.

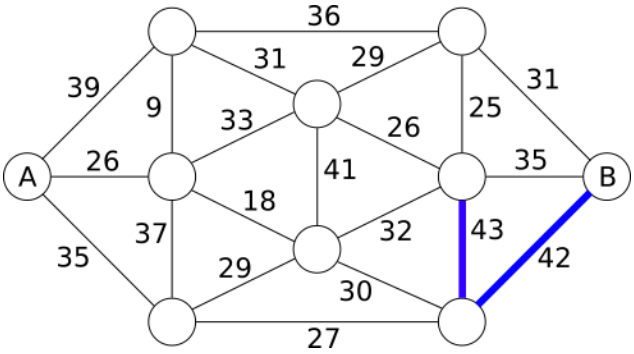
We can start by using the heaviest weight that at least one bridge can sustain: 43 tonnes.
Only one bridge can sustain this weight, and if we select it this is not enough to form a path from Island A to Island B.



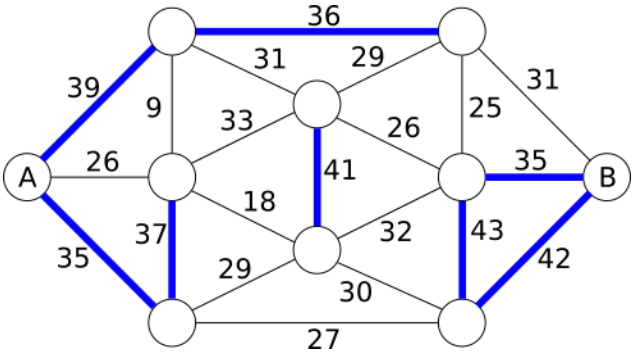
To gain access to more bridges, we reduce the weight to 42 tonnes, and add one more bridge:

Continued on next page

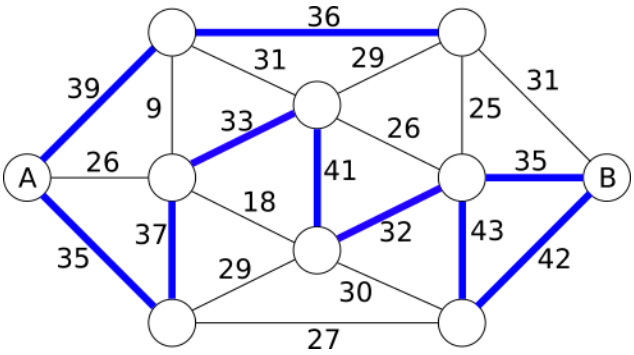
Beaverburg Delivery – continued



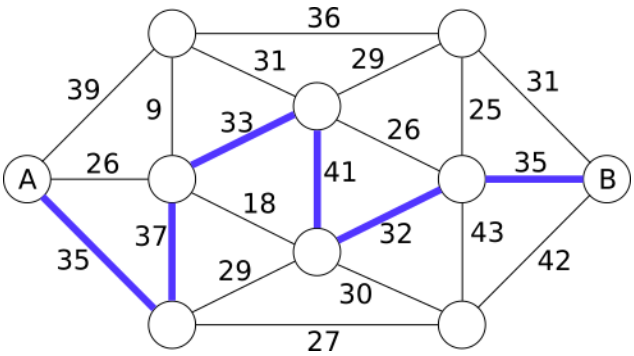
This is still not enough to connect A to B, so we continue reducing the weight, adding bridges that can sustain 41, then 39, 37, 36 then 35 tonnes.



At 35 tonnes, we have already selected 8 bridges, but there is still no path from A to B.



We continue to reduce to 33 tonnes, then 32 tonnes, which gives us this set of available bridges:



We now have enough bridges to travel from Island A to Island B. We can deselect the bridges that we don't need, so that only the correct path is selected.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Specification, Algorithms, Digital Systems

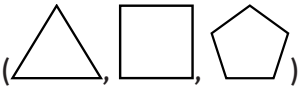
The problem given is an example of a widest path problem from graph theory. Its applications include finding maximal bandwidth between endpoints in a network and other "bottleneck" problems.

It can also be used as a subproblem for finding the maximum capacity path in a Ford-Fulkerson maximum flow algorithm. Our greedy algorithm is similar to the Kruskal's algorithm, which is used for computing the minimum spanning tree of a graph.

This question comes from
South Korea

Years 3+4
Years 5+6
Years 7+8 Hard
Years 9+10
Years 11+12

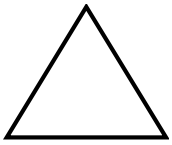
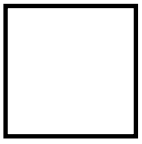
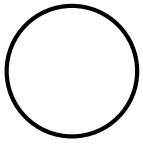
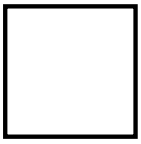
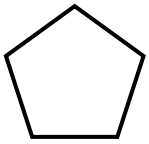
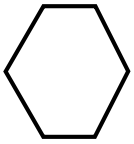
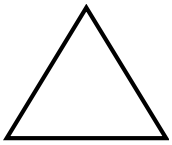
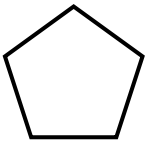
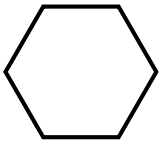
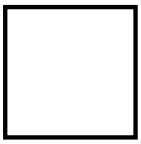
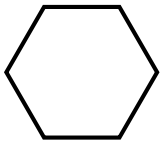
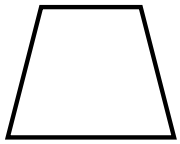
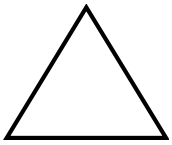
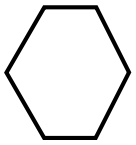
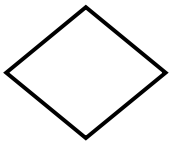
Wizard Beaver's Alchemy

The wizard beaver has a magic box that melts 2 pieces of different colours from 3 basic jewellery shapes () to make a new one.

The wizard beaver always uses his magic box to make the jewellery needed.

The new jewellery is made by the following rules:

- Only two different shapes and colours of jewellery can be melted in the magic box at the same time.
- Different shapes and colours of jewellery are formed after the melting process according to the following rules of the magic box.
- The melting process can be done more than one time.

Magic Box Rules for Jewellery Shapes				Magic Box Rules for Jewellery Colours			
1st Shape	+	2nd Shape	New Shape	1st Colour	+	2nd Colour	New Colour
	+			Blue	+	Yellow	Green
	+			Blue	+	Red	Purple
	+			Yellow	+	Red	Orange
	+			Red	+	Green	Black
	+			Purple	+	Yellow	Black

Question

Which of the following peices CANNOT be made from the four pieces of jewellery shown below?

Select from one of the following options.

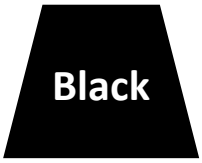
Continued on next page

Wizard Beaver's Alchemy – cont.

Answer

The answer is:

C



A black trapezoid cannot be made from any of the combinations. It is possible to make jewellery in the other shapes from the given pieces.

A

B

D

It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Abstraction, Algorithms, Evaluation

Concepts: Abstraction, Specification, Algorithms

In Computer Science, actions are performed from a starting situation, leading to a certain outcome. Different starting situations generally lead to different outcomes, but the relation between the starting situation and the outcome is well defined and determined. In this task, the shapes and colours of the original pieces represent the starting situation and the shape and colour of the resulting piece represents the outcome. The task requires us to reason about the relationship between the starting situation and the outcome.

When reasoning about such a relationship, shapes and colours can always be considered separately, since they are independent of each other. Breaking down problems like this is a typical strategy used in algorithmic thinking in order to conceptualise solutions.

In Computer Science, a *function* is a way to implement a relationship between the starting situation and the outcome. *Parameters* are used within a function to represent the starting situation and the *return value* of the function is the outcome. Functions can be reused and can be applied one after the other. As far as answer B is concerned, the outcome of the first *application of the function* becomes the starting point for the second one.

This question comes from
El Salvador

Car Factory

A car factory is designing a new fleet of cars to be sold in the future. These cars vary in (a) colour, (b) interior finish, (c) tyres, and (d) engine type.

Colour	Interior finish	Tyres	Engine
0. Blue	0. Black leather	0. Sport tyres	0. Diesel
1. Red	1. White leather	1. Urban tyres	1. Petrol
2. White	2. Black fabric	2. Road tyres	2. Hybrid
3. Black	3. White fabric	3. Traction tyres	3. Natural gas
4. Green	4. Red deluxe material	4. Deluxe tyres	4. Electric

The factory will assign a number from 0 to 624 to all possible combinations of car types. It will start by assigning the number 0 to the car with features 0 0 0 0, which is the blue car with black leather, sport tyres and a diesel engine.

Car 1 will be the car with features 0 0 0 1, which is the blue car with black leather, sport tyres and a petrol engine.

Car 2 will be the car with features 0 0 0 2, while car 5 will be car with features 0 0 1 0.

Example

The car with features 2 0 1 4 will be the white car, with black leather, urban tyres and an electric engine. This would be car number 259.

Question

What is the car number for a green car, with red deluxe interiors, deluxe tyres and a natural-gas engine?

Continued on next page

This question comes from
El Salvador

Car Factory – continued

Answer

The answer is: 623

The features that are listed have number 4 (Green), 4 (Red deluxe), 4 (Deluxe tyres), and 3 (natural gas). Written contiguously, this is 4 4 4 3, which can be read as a base-5 number which, converted to a decimal, is 623. This is the car number but how did we obtain this value?

We note from the example that each time they go up in in the specifications of the last option (the engine type), you add 1 to the model number. More interesting is that going up in the specifications makes us add not 1 to the car number (otherwise we would mix it up with a change in the engine type), but 5. Why 5? Because there are only 5 valid options for the engine, and after that, any following car number must have a change somewhere else – here, in the option for the tyres.

If we continue to think like this, we find that going up one level in the option for the interior will make us add 25 to the car number. Why 25? Because there are $5 \times 5 = 25$ possible options for the other two leftmost options. Similarly, choosing the next colour causes the car number to be incremented by $5 \times 5 \times 5 = 125$.

We can just then count how many times each option was "upgraded" from the base option, and add up, multiplied by the factors we have just discussed:

(Colour) 4 × 125 + (Interior) 4 × 25 + (Tyres) 4 × 5 + (Engine) 3 = 623

It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Abstraction, Modelling & Simulation

Concepts: Abstraction, Data Representation, Data Interpretation, Specification, Algorithms, Digital Systems

Base changing is very useful because we can use each to communicate information that in decimal form might not be as clear.

Decimal base, the one we use in our daily lives, is not the way that computers store numbers. In computers the base is binary: a base composed only of 0s and 1s. When you input a number into a computer, the computer transforms that number into binary, and then uses it to perform its calculations.

Actually, computers store not only numbers, but all data as a sequence of binary digits: sound, images, movies, 3D models, etc.

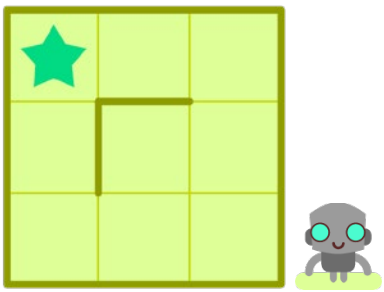
Bebras Challenge 2021 Round 2

Years 9+10

This question comes from Russia

Recover my Robot

Natasha lost her robot in a park. The park is a square which is made up of a grid of 3 by 3 smaller squares.



The robot could have been lost in any of the nine squares.

Natasha can manually send a sequence of commands to the robot. She can command it to move either one square UP, LEFT, RIGHT or DOWN.

If the robot is moving towards a wall, it won't be able to go further and stands still. The walls are drawn on the picture by a thick (green) line.

Natasha doesn't know where the robot is.

Question

What is the shortest sequence of commands that she can send to the robot, so that it will be guaranteed to finish in the square with a star?

DOWN – LEFT – DOWN – LEFT – UP – UP

RIGHT – UP – UP – LEFT – LEFT

RIGHT – UP – RIGHT – UP – LEFT – LEFT

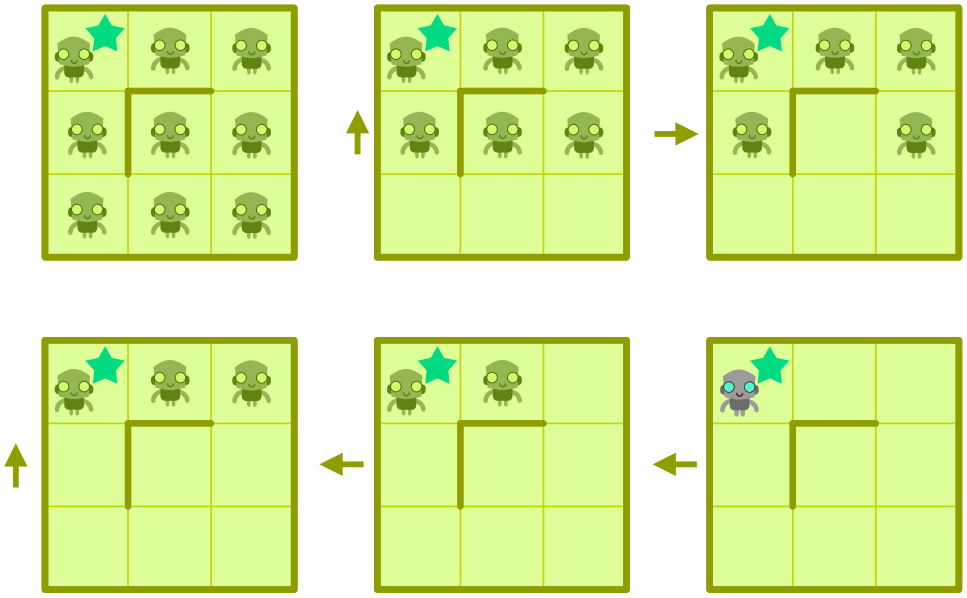
UP – RIGHT – UP – LEFT – LEFT

Answer

The correct answer is: UP – RIGHT – UP – LEFT – LEFT

To confirm that the correct answer is UP – RIGHT – UP – LEFT – LEFT, we will test the sequence of commands, starting from all the possible initial positions. After each step, we examine possible positions for the robot, and finally, there should be only one possible position for the robot.

The images below show the possible locations for the robot after each command in the correct answer is executed.



Continued on next page

Recover my Robot – continued

It is impossible to find a solution with 4 steps, because the square in the bottom right needs four commands to move to the square (UP – UP – LEFT – LEFT or LEFT – LEFT – UP – UP). Given these exact commands, a robot in the center square will not be able to move to the desired square.

Although the answers DOWN – LEFT – DOWN – LEFT – UP – UP and
RIGHT – UP – RIGHT – UP – LEFT – LEFT also move the robot the desired square, they require a longer sequence of commands (6).

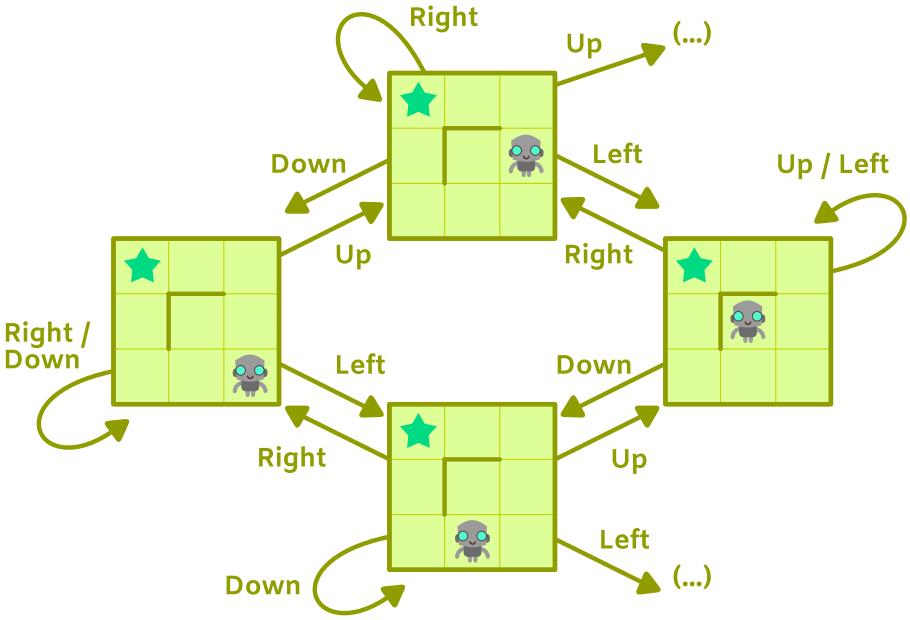
The answer RIGHT – UP – UP – LEFT – LEFT is incorrect since the robot will not reach the desired square if it has been lost in the bottom left square.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Data Interpretation, Specification, Algorithms, Implementation

This is a task about synchronising words in finite automata. At any time, the robot is in one square and can receive a command sent by Natasha. After each command, it either changes square or stays in the same square. The position of the robot is what we call the state, and a command can therefore change the state. We can represent these state changes graphically, with a drawing like the following one (which does not show all the possible state changes):



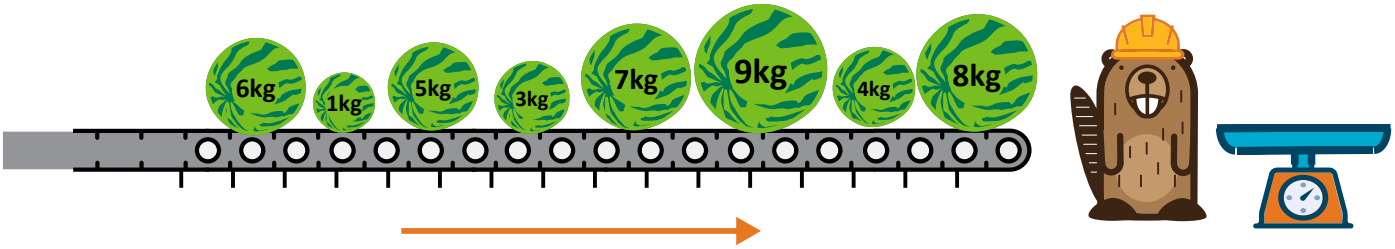
Such a representation is what we call a finite automata. Given an initial state, we can execute a sequence of commands to find out what will be the reached state. Such a sequence is what we call a word. To summarise, you can execute a word on a finite automata, to go from one state to another state. For this problem, we do not know the initial state. What we are searching for is a word to execute so that to always reach the same final state, no matter what is the initial state. This is exactly what is called a synchronising word. Executing a synchronising word ALWAYS lead to the same final state, no matter the initial state.

Conveyor Belt

Beaver works in a factory. His job is to weigh watermelons for shipment to customers.

Each shipment has to be exactly 20kg. Watermelons are placed on a moving conveyor belt.

Beaver is between the conveyor belt and the scale. He takes each watermelon off the conveyor belt and then places it on the scale.



Every time a new watermelon is placed on the scale, he checks the new total weight:

- If the total weight remains less than or equal to 20kg, he leaves the watermelons on the scale
- if the total weight exceeds 20kg, he takes the new watermelon off the scale and does not ship it.

Once the total weight on the scale is equal to 20kg, Beaver will ship the watermelons to the customer.

Question

Given the current state of the conveyor belt, how many watermelons will be shipped?

3 4 5 6

Continued on next page

Conveyor Belt – continued

Answer

The correct answer is: 4

The first watermelon weighs 8kg, and is placed on the scale. Therefore the total weight on the scale will be 8kg. The watermelons will be left on the scale.

Then he adds the next watermelon with a weight of 4kg. Now the total weight on the scale will be 12kg. The watermelons will be left on the scale.

The next watermelon is 9kg and will cause the total weight on the scale to be higher than 20kg. The 9kg watermelon will be placed aside and not shipped. The weight of watermelons on the scale is still 12kg.

The next fruit is 7kg making the new total on the scale 19kg. The watermelons will be left on the scale.

The next two fruits have a weight of 3kg and 5kg. Either of them will cause the total weight on the scale to be higher than 20kg, and so they will both be discarded.

The next watermelon is 1kg. This make the total total weight 20kg. The 4 fruits on the scale (8kg, 4kg, 7kg, 1kg) will be shipped.

The remaining watermelon is 6kg, and will not be enough to create another shipment.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Abstraction, Algorithms, Evaluation

Concepts: Specification, Algorithms

A while loop consists of a block of commands and conditions. First, the condition is evaluated. If it is true, commands in the block are executed. These commands are repeated until the condition is no longer correct. In this case, the box will be filled as long as the total weight is less than 20kg.

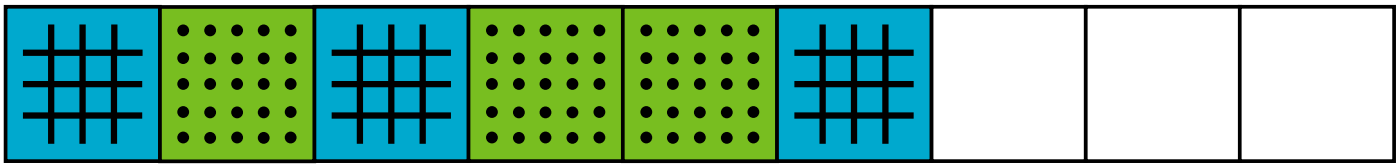
A conditional rule also consists of a block of commands and conditions (if then). First the condition is evaluated, if it is true, the commands in the block are executed. In this case, when a watermelon is about to be added to the box, it is checked to make sure that it will not cause the box to weigh more than 20kg. If it would not, it is added to the box, otherwise it is discarded.

```
a=0
k=0
while a<20:
    b=int(input("What is the weight of the watermelon?"))
    if a+b<=20:
                a=a+b
                k=k+1
print(k)
```

```
int b,a=0,k=0;
while (a<20){
    cout<<" What is the weight of the watermelon?";
    cin>>b;
    if ((a+b)<=20){
      a+=b;
      k++;
    }
}
cout<<k;
```

Permutations

Three blue and three green cards are located in a row of 9 cells in the following initial order:



Performing an operation means moving some of the cards from one cell to another (at the same time).

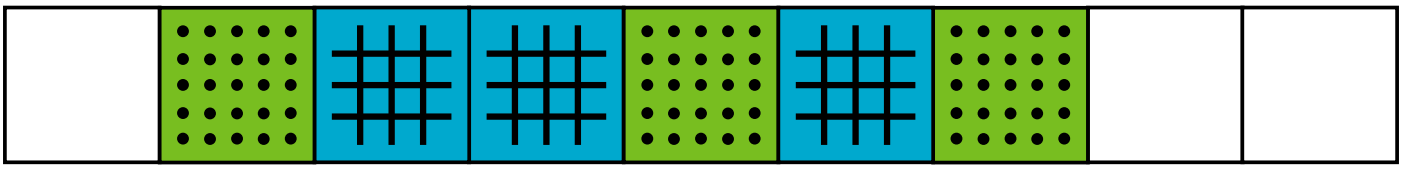
The following two types of operation are allowed. Cells and cards are numbered from left to right:

- **Operation 1:** Cards 1 and 4 are moved three cells to their right.
  This means that card 4 will move three cells to its right, and card 1 will move to the original cell of card 4. But be careful! Note that empty cells do not count as cards when working out the positions of the cards!
- **Operation 2:** Cards 1, 3 and 5 are moved two cells to their right in a similar manner.

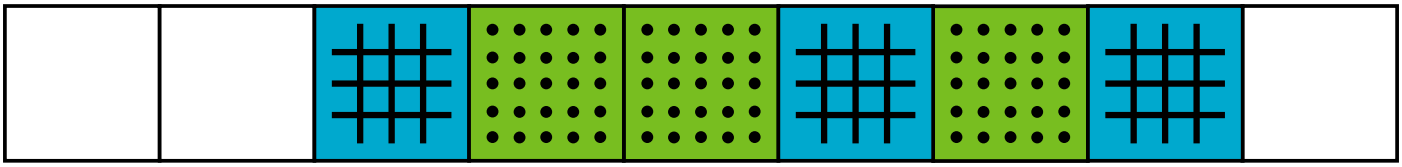
You can do multiple operations one after another.

For example, starting from the initial sequence, doing "Operation 1; Operation 2" would result in the following:

After Operation 1:

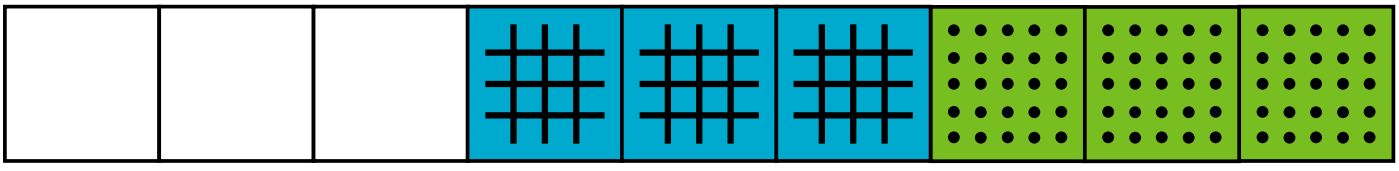


After Operation 2:



Question

Which group of operations, starting from the initial sequence of cards, will lead to the following sequence of cards?



Operation 1; Operation 2; Operation 1;

Operation 2; Operation 2; Operation 1;

Operation 1; Operation 1; Operation 2;

Operation 1; Operation 1; Operation 1;

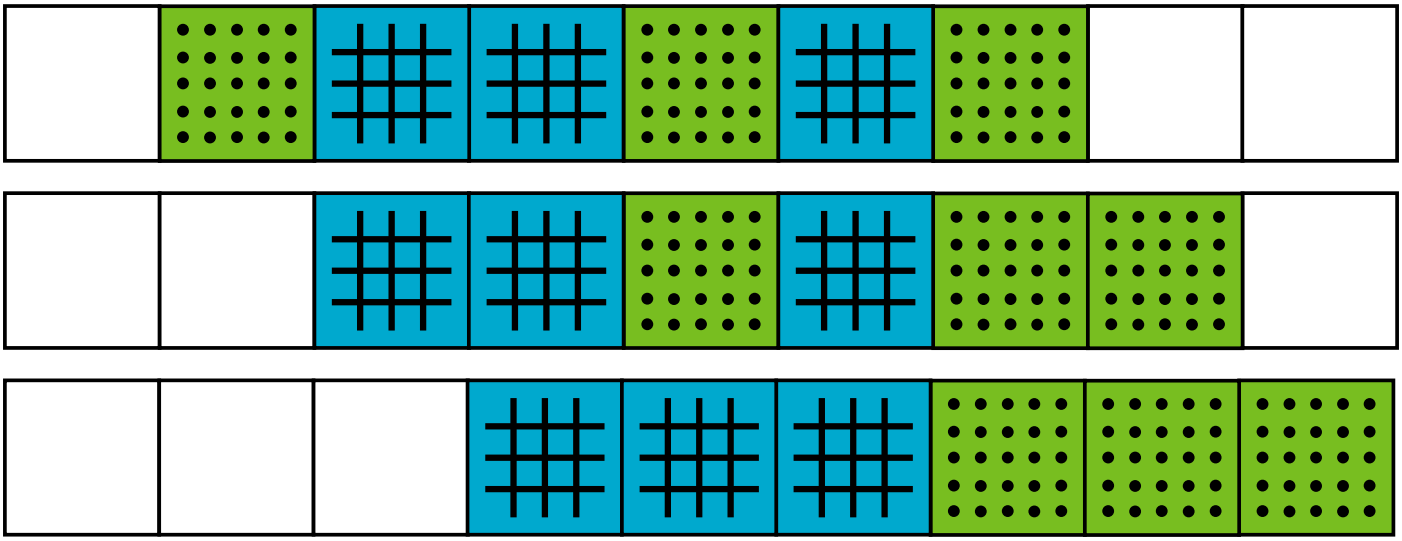
Continued on next page

Permutations – continued

Answer

The correct answer is: Operation 1; Operation 1; Operation 2

The state of the cards after each operation is shown in the following images:



The correct answer could be found (and the incorrectness of all others verified) by applying each sequence and checking the resulting state. One way to reduce the amount of work is to find a condition that identifies a wrong sequence without having to apply it completely. For example, any sequence that places a blue card in any of the three rightmost cells cannot lead to the required final state. This is because both operations will only move cards to right and in the final state the blue cards are in the middle three cells.

In the example, Operation 1 is followed by Operation 2. The example shows us that this results in a blue card in cell 8. This means that any sequence that begins with “Operation 1; Operation 2” cannot be correct, and so the first option must be wrong.

Applying Operation 2 first places blue cards in cells 5 and 6. We can quickly see that applying another operation will place a blue card in either cell 7 or cell 8 (for Operation 1 or Operation 2, respectively). This means that the second option cannot be correct either.

The difference between the third option and fourth option is in the final step, so we cannot avoid testing those options fully.

It’s Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Data Interpretation, Specification, Algorithms

Permutations are the different orders in which some elements can be (re)arranged – a process called permuting. These ordered elements form tuples. For instance, there are six permutations of the set {1,2,3}, namely the tuples (1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2) and (3,2,1).

Permutations are very important in mathematics and also in computer science. For example, they are used for analyzing sorting algorithms, which are extremely important in programming.

This task is related with resorting or rearranging elements. It develops the skills of designing and executing an algorithm to accurately do that job, which is not always easy. The task also shows how changing the order in which commands are executed can significantly affect the final result.

Party Message

Beaver Ann and her friends use a special code to send notes to each other. Now Ann is preparing a party and wants to use a secret message to send the entrance password to her friends:

1	1	1	0	0	1	0	0	1	1	1	0	1	0	1
0	0	1	0	1	1	0	0	0	0	1	0	1	0	1
1	1	1	0	0	1	0	0	0	1	0	0	1	1	1
0	0	1	0	0	1	0	0	1	0	0	0	0	0	1
1	1	1	0	1	1	1	0	1	1	1	0	0	0	1

Secret message:

```
1:321231111
0:2112411111
1:3213123
0:21212151
?
```

Question

What will be the last line of the secret message?

0:3131331 **1:321231111** **1:3131331** **1:231231111**

Continued on next page

Party Message – continued

Answer

Option 3 is the correct answer.

The first digit on each line of the message tells whether the corresponding line of the image starts with a white square or a blue square (0 for white and 1 for blue). The digits after the ":" tell how many squares of each alternating colour will follow.

The last line of the image begins with a blue square (1). The ":" is followed by a digit showing the number of squares in the starting colour (3 for three blue squares). This is followed by 1 white followed by 3 blues followed by 1 white and so on.

This means that the correct code is: 1 : 3 1 3 1 3 3 1

Option 1 can not be correct because it begins with "0:" means that the first square would be white. In fact this line in the message would describe a line that is the exact opposite of what Ann wants.

Option 2 can not be correct because it would describe a line of the image identical to the first one: after 3 blue squares and 1 white square there would be 2 white squares and so on.

Option 4 can not be correct because the last line of the image would start with 2 blue squares

It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Modelling & Simulation, Algorithms

Concepts: Data Collection, Data Representation, Data Interpretation, Specification, Algorithms, Digital Systems

This task introduces several data encoding concepts in sequence:

- first the password (a text) is represented as an image;
- then the image is encoded as a bitmap;
- finally, the bitmap is turned into the secret message using run-length encoding.

Having several different ways to represent the same information and needing to convert from one representation to another is common in data processing.

The run-length encoding (RLE) introduced in this task is a very simple form of lossless data compression. The RLE algorithm exploits sequences having the same value occurring many consecutive times. In general, each such sequence is encoded by storing the value only once, accompanied by the repetition count. Note that while in most cases the size of output is smaller than the size of the input, in the worst case it may actually be larger.

In this task storing of the values is also omitted, by assuming that for each count the value is the opposite of the previous one. (Can you think of an example image where this approach would get Ann in trouble? Can such images appear as representations of passwords?)

The task requires from the student the skills of generalisation and algorithmic thinking. First the student has to deduce the RLE algorithm by comparing the corresponding lines of the image and the message in the example, as the algorithm is not explicitly given in the task. The student then has to apply this algorithm to the last line of the image to obtain the last line of the message.

Finally, it should be noted that RLE is not a strong encryption algorithm. After seeing a few secret messages, it would no be very difficult to deduce the algorithm even without knowing the corresponding passwords. So, this should not be used for any really important secrets.

This question comes from Japan

Glowing Panels

The king of the Bebras Kingdom decided to make a piece of giant art using glowing panels and displays it in the square to celebrate the Bebras Kingdom National Holiday. The art piece is created using a total of 36 panels as shown below in *Figure 1*.

Each panel switches between on and off when stepped on.

The king starts from the "IN" panel in the figure. He can step on adjacent panels up, down, left or right (not diagonally), and proceeds to the "OUT" panel to create various glowing patterns. He can step on the same panel more than once.

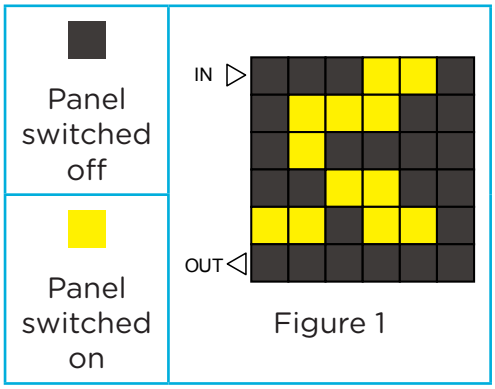
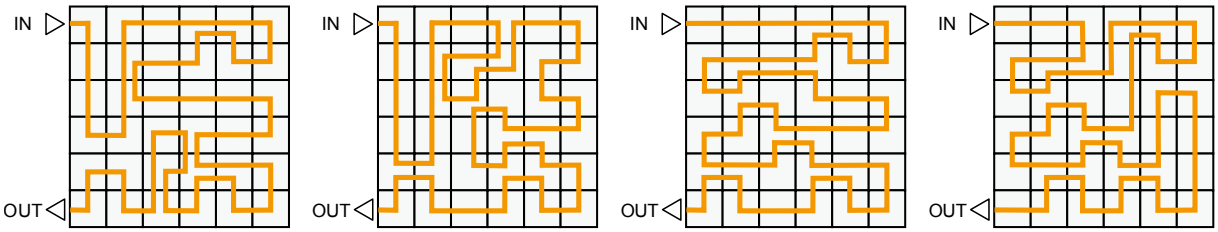


Figure 1

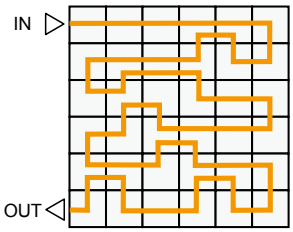
Question

The king made the pattern shown above, but he didn't like it, so he wanted to turn on all the panels to start over. Choose which of the following routes he can follow to light up everything.

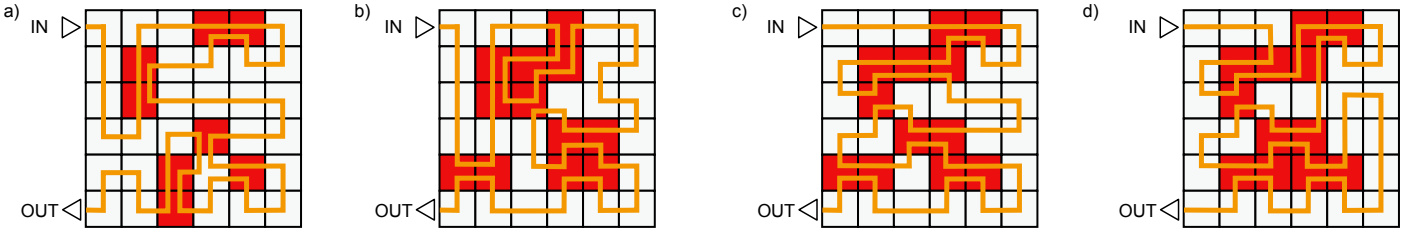


Answer

The correct answer is:



If you select a route that passes every glowing panel twice and the others once, all of the panels will be turned on. Below each of the paths has been checked, and panels are coloured red if they are stepped two times.



It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Modelling & Simulation, Evaluation

Concepts: Data Representation, Data Interpretation, Specification, Algorithms, Implementation, Digital Systems

The ability to trace a procedure is important in programming. This problem is found by tracing how the lighting pattern (that is, the data handled by the program) is changed by a given route.

However, tracing everything takes time, so it is also important to narrow down the answers by understanding the lighting change patterns.

This question comes from
Czechia

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Medium
Years 11+12

Echo Cypher

Do you know how to add two letters? It's easy: convert both letters to the number of their place in the alphabet. Each letter's number is given below.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

The result of adding two letters is a letter whose position in the alphabet is a result of addition of those numbers.

Thus, for A+A, we find 1+1 = 2, and 2 gives us B. Thus A+A = B.

A ↔ 1
\+ A ↔ + 1
= B ↔ = 2

Similarly, A+B=C, and C+E=H.

If the resulting number is higher than the number of letters in the alphabet, we start counting again from the beginning of the alphabet. Thus, Z+A = A, and Y+C = B.

Jaroslav uses this kind of addition to encrypt his notes. His process for encrypting is a simple one: first he thinks up one number – he calls it '*modus*'. He writes one word and then writes the same word below the first one but shifted to the right by a number of many positions equal to the *modus*.

For example, if the *modus* is 3, this process looks like:

R	A	B	B	I	T			
			R	A	B	B	I	T

He writes the result of the addition of the letters in the same column in a 3rd row. The text in the 3rd row is the encrypted text.

Example: Taking a modus of 2, and applying the encryption to the word 'BEAR'

B	E	A	R		
		B	E	A	R
B	E	C	W	A	R

So the word 'BEAR' upon encryption with modus 2 becomes 'BECWAR'.

Question

Jaroslav's notes contain some encrypted text. The first 9 letters of the text are ACDGDQGPE. We don't know the modus for the encryption he used. We know that the start of the original text is one of the four options given below. Which one is it?

ACDCMETALL **ACCUMULATE** **ABBEYROAD** **ABBREVIATE**

Continued on next page

This question comes from
Czechia

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Medium
Years 11+12

Echo Cypher – continued

Answer

The correct answer is ABBEYROAD.

We don't know the modus but we know that the original word is shifted to the right a number of positions equal to the modus. So the first several letters of both the original text and the encrypted text must be the same. Using this to work out the possible options for the modus will enable us to narrow down the answer.

If modus is 4 (or more), the first 4 letters of both the original and encrypted versions must be the same so the original text must start with ACDG. There is no such option starting with ACDG. So the modus cannot be greater than 3.

If modus is 3, the first 3 letters of both the original and encrypted versions must be the same. So the original text must start with ACD. The only such option is ACDCMETALL. The 4th letter of the encrypted text is G, which means the 4th letter of the original text must be F, as A + F = G. But ACDCMETALL has the letter C in the fourth position. Thus, modus cannot be 3.

If modus is 2, the original text must start with AC. Options ACDCMETALL and ACCUMULATE meet these requirements. The 3rd letter of the encrypted text is D, which means the 3rd letter of the original text must be C as A + C = D. Then, only ACCUMULATE could be correct. But the 4th encrypted letter is G, which means the 4th letter of the original text must be D, as G = C+D. But the 4th letter in option ACCUMULATE is U. Thus, modus is not 2.

This means that modus must be 1.

Since the 2nd letter of the encrypted text is C, the 2nd letter of the original text must be B, as A+B = C. Further, the 3rd letter of the encrypted text is D, meaning that the 3rd letter of the original text is B, as B+B = D. The original text must start with ABB. Options ABBEYROAD and ABBREVIATE could be correct. Now, we look at the 4th encrypted letter which is G. This tells us that the 4th letter of the original text must be E, as B+E = G.

Thus, the correct answer is ABBEYROAD, and the 10th letter will be D.

A	B	B	E	Y	R	O	A	D	
	A	B	B	E	Y	R	O	A	D
A	C	D	G	D	Q	G	P	E	D

It's Computational Thinking

Computational Thinking Skills: Pattern Recognition, Abstraction, Algorithms, Evaluation

Concepts: Abstraction, Data Representation, Data Interpretation, Specification, Algorithms, Digital Systems

Often, we do not want messages sent over networks to be easily read by anyone who intercepts them. These messages may contain passwords and other private information which need to reach the recipient alone. Therefore, messages are encrypted, which turns them into secret messages. For this to work, the recipient of the message must be able to decrypt the secret message and reveal its original text. However, it should not be possible for unauthorised persons who capture a secret message to be able to translate it.

The encryption and decryption of secret messages is done using ciphers. There are many different ciphers. The cipher used in this task replace each letter in the plaintext by a letter some number of positions down the alphabet. This number is given by positions of individual letters of the same plaintext in alphabet and it is not fixed. This cipher is easy to break for computers when the text is long enough because it does not change a proportion of letters used in the language.

The science dealing with encryption is called cryptography. Modern cryptography is a large area of research that involves many other ciphers based on complex mathematics.

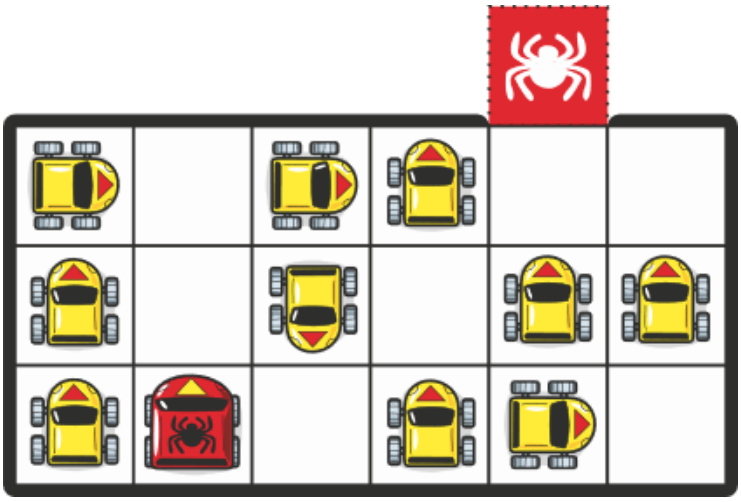
Robot Parking

Robot cars in a fenced area can move around according to the following rules:

In one action, a robot can either:

- move one square forward
- move one square backward
- turn left (90 degrees) in the current square, or
- turn right (90 degrees) in the current square.

In addition, only the spider car can move outside the fenced area to the spider square.



Question

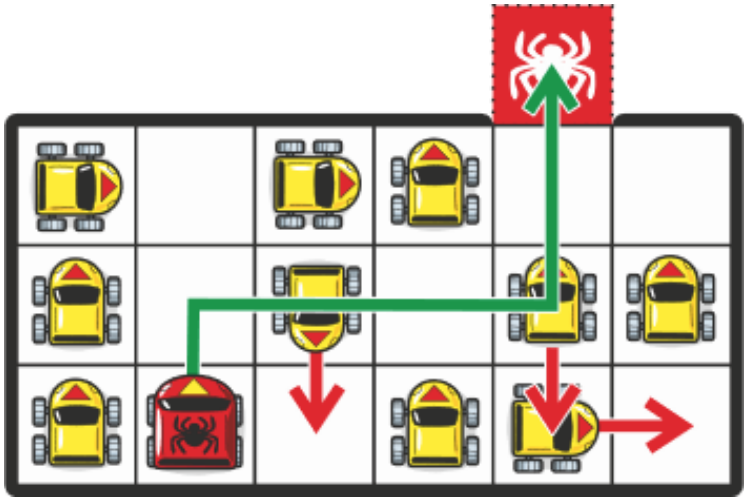
What is the minimum number of robot actions needed to get the spider car onto the spider square?

9 **11** **13** **15**

Answer

The correct answer is 11.

The spider car can reach the spider square in 11 actions as shown:



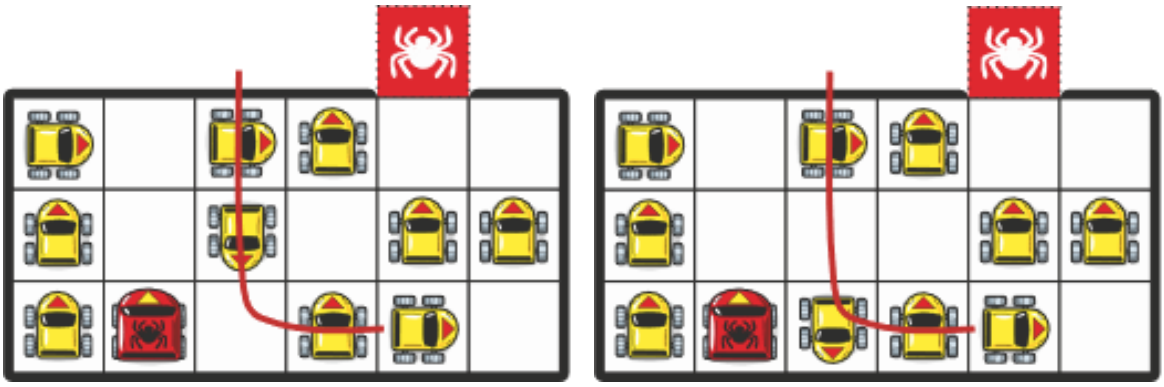
It now remains to prove that 11 is the minimum number of actions needed.

Suppose the spider car was the only car in the fenced area. To reach the spider square it must move up 2 times, right 3 times, and turn 2 times. These actions can be done in a variety of different combinations, but the spider car will always require these 8 actions. Of course, the spider car is *not* the only car in the fenced area and more moves are needed to unblock its path.

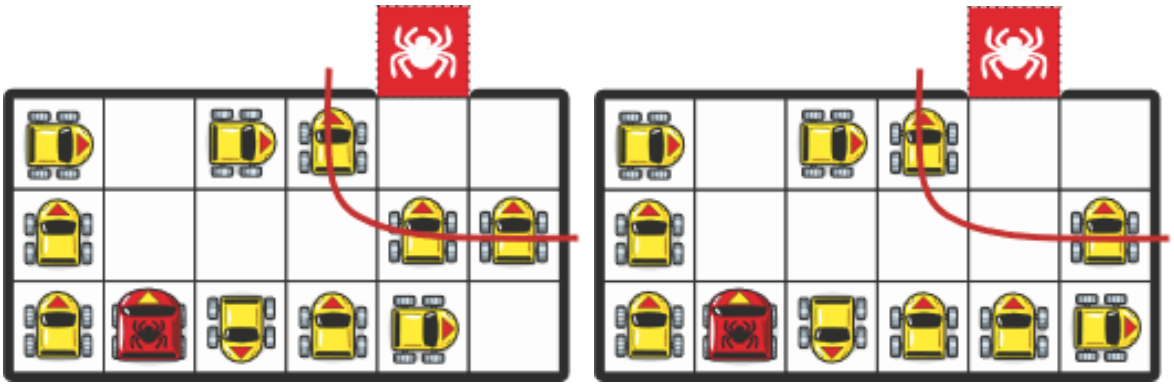
Continued on next page

Robot Parking – continued

First we need to create a way through this "L-shaped" barricade. This can be done using one action as shown:



Now we need to create a way through this second "L-shaped" barricade. This can not be done using one action (without blocking the exit). It needs at least two:



Therefore, the minimum number of robot actions needed is 8+1+2=11.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Specification, Algorithms

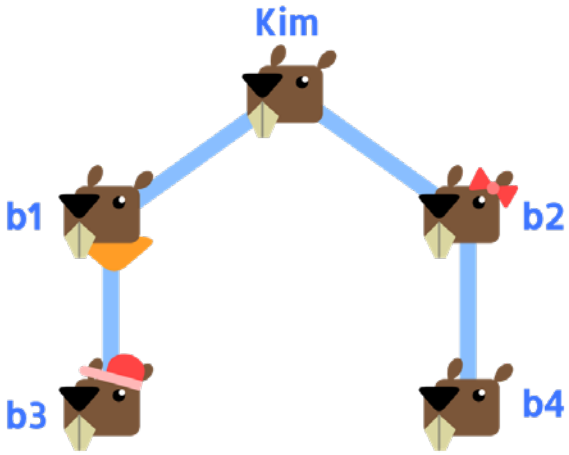
Proving a solution is minimal (or optimal) can be a very difficult thing to do. Often the only way to do it is to list *all* solutions, in order to shown that no solution exists that is better than the one you found. This is known as *exhaustive search*. Exhaustive search can be infeasible to do by hand in many instances but is a straightforward and fast task to do using a computer.

Sometimes the number of possible solutions to a problem is so large that, even using a computer, exhaustive search takes too long. When this happens there are other computer science algorithms that can be applied in order to simplify the search. *Branch and bound* and *greedy* are examples of two such algorithms.

Beaver Meetings

Beaver Kim went to the market. While shopping, he met two other beavers there, Beaver 1 and Beaver 2. Beaver 1 and Beaver 2 also met two beavers at the market. How many beavers could have been at the market so that Beaver Kim, Beaver 1 and Beaver 2 were able to meet two beavers each?

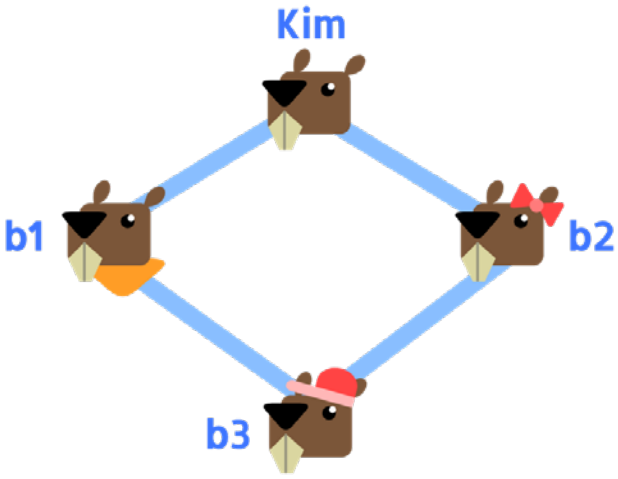
We can represent the beavers and the potential beavers they came into contact with at the market with a graph, as shown below:

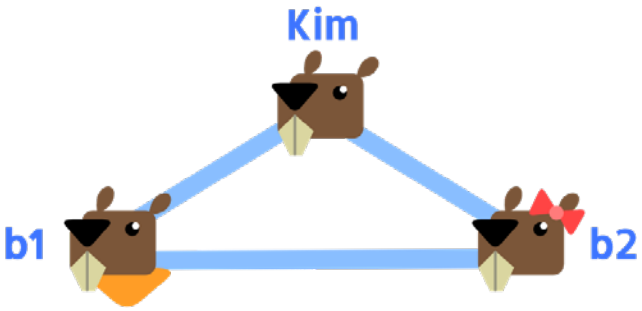


- Beaver Kim met Beaver 1 and Beaver 2.
- Beaver 1 met Kim and Beaver 3.
- Beaver 2 met Kim and Beaver 4.

In this case, at least 5 beavers were at the market.

There are two other potential solutions to the same question:





In the cases above, at least 4 beavers (left graphic) or 3 beavers (right graphic) were at the market.

The following week, Beaver Kim went to the market again and met three other beavers there. These three beavers also met other beavers at the market. These beavers met two, five and five beavers respectively.

Question

What is the smallest possible number of beavers that were present at the market?

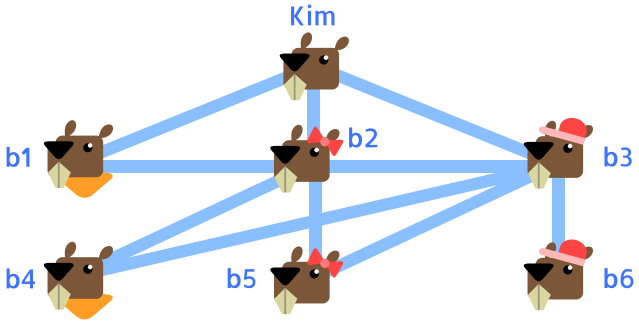
5 6 7 8

Continued on next page

Beaver Meetings – continued

Answer

The answer is 7 beavers, as shown in the graph below.



Counting only Beaver Kim and the beavers he met, i.e. Beaver 1, Beaver 2 and Beaver 3, we already have 4 beavers.

Let's look at Beaver 1 next. Beaver 1 met 2 beavers. We need to calculate the smallest number of beavers, so we have to assume that Beaver 1 met Beaver Kim and either Beaver 2 or Beaver 3. The minimum number of beavers at the market would still be 4 at this point, since we can meet the requirements of Beaver Kim and Beaver 1 with the 4 beavers we already have.

Now we look at Beaver 2 who met five beavers. We know that Beaver 2 met Beaver Kim and potentially Beaver 1. If we assume that Beaver 2 met Beaver 1 and Beaver 3, there must have been at least two additional beavers (Beaver 4 and Beaver 5) that Beaver 2 met. The minimum number of beavers at the market thus increases to 6.

The total number cannot be 6, however. Because Beaver 1 only met 2 beavers, they could only have met Kim and Beaver 2 and Beaver 3, since that would be 3 beavers that they met. Beaver 1 met either Beaver 2 or Beaver 3, or neither of them. We assumed at the start that Beaver 1 met Beaver Kim and Beaver 2, so they cannot have also met Beaver 3. In this case, Beaver 3 can have met Beaver Kim and Beaver 2 as well as Beaver 4 and Beaver 5. In order to have met 5 beavers, however, Beaver 3 must have met another beaver who is not in our list already, Beaver 6.

Including Beaver Kim, this gives us 7 total beavers at the market. The answer is at least 7.

The graph above with seven nodes can be used to help solve this problem and to prove that a solution with seven beavers exists.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

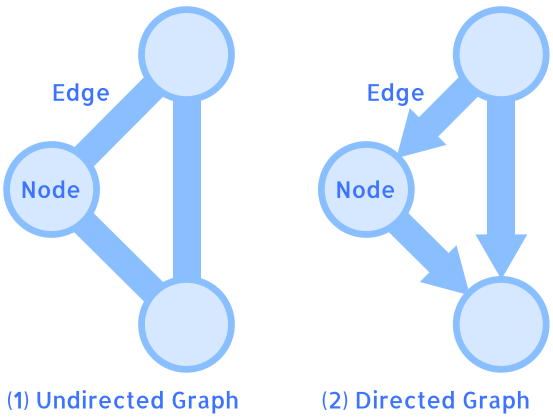
Concepts: Data Interpretation, Specification, Algorithms, Digital Systems, Interactions

In this task, we use a "graph" data structure consisting of nodes (also called vertices or points) and edges (also called links or lines) to help solve the problem.
There are two types of graphs:

1. undirected graphs, where edges link nodes symmetrically, i.e. there is no hierarchy between the nodes and the connection has no direction;

and

2. directed graphs, where edges link nodes asymmetrically, i.e. there is a potential hierarchy between the nodes and the connection goes in a certain direction.



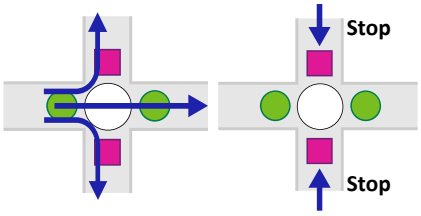
(1) Undirected Graph (2) Directed Graph

This question comes from
Taiwan

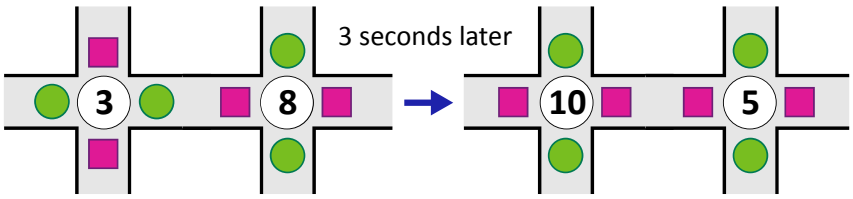
Years 3+4
Years 5+6
Years 7+8
Years 9+10 Medium
Years 11+12

Traffic Lights

There are traffic lights in every intersection in Beaver City. The following are two important details of the traffic regulations:

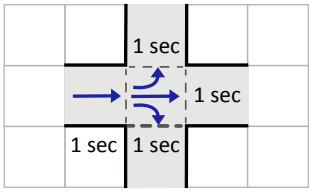


When the light is green, beavers can either go forward or make a right or left turn.
When the light is red, beavers must stop and wait for the light to turn green again.

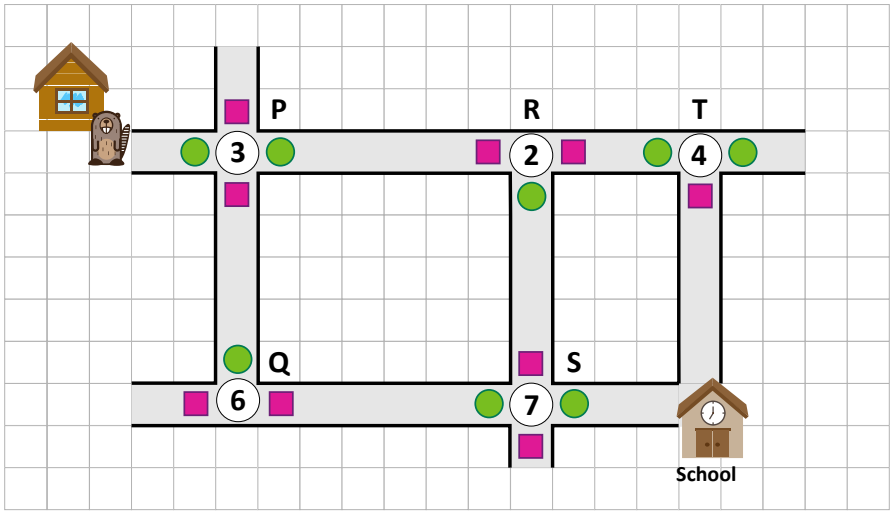


As shown in the picture, the lights alternate between green and red every 10 seconds.
Each set of traffic lights at an intersection has its own timer which keeps counting down from 10 to 0.
Beavers can see the timer, and the lights will switch color when the timer hits 0.

When Little Beaver walks, it takes one second to advance one square; it also takes him one second to either cross the road or make a turn at an intersection.



The map below shows the roads connecting Little Beaver's home and school. The status of each timer at the moment Little Beaver leaves his house is shown on the map.



Question

Based on the timers, if Little Beaver wants to walk to school without stopping for any red lights and without crossing the same intersection twice, which route should he take?

P→R→T→SCHOOL

P→Q→S→SCHOOL

P→R→S→SCHOOL

P→Q→S→R→T→SCHOOL

Continued on next page

Traffic Lights – continued

Answer

The correct answer is P→R→S→SCHOOL.

We need to consider whether the light is green when Little Beaver arrives at an intersection. The timer at P shows 3 seconds remaining before an west-east red light at the beginning. When Little Beaver arrives at P, he has traveled 2 squares in 2 seconds. Therefore, the west-east light is still green and Little Beaver can either cross the street or make a turn. When Little Beaver arrives at P, the timers look like this:

	P	Q	R	S	T
Timer	1	4	10	5	2
West-East Light	Green	Red	Green	Green	Green
North-South Light	Red	Green	Red	Red	Red

Little Beaver can now either turn right to go to Q, or go straight to go to R. If he turns right, it will take him 6 seconds to get to Q: 1 second to cross the P intersection plus 5 seconds to travel the distance between P and Q. Since the north-south light at Q will turn red in 4 seconds, going to Q is not an option since it will result in Little Beaver having to stop for a red light. Knowing that Little Beaver cannot travel directly from P to Q without getting stuck at a red light, we can eliminate P→Q→S→SCHOOL and P→Q→S→R→T→SCHOOL.

If instead, Little Beaver went straight ahead to R, it would take him 7 seconds to reach it and the timers would look like this when he arrived:

	P	Q	R	S	T
Timer	4	7	3	8	5
West-East Light	Red	Green	Green	Red	Red
North-South Light	Green	Red	Red	Green	Green

So Little Beaver would find a green light at R. Now he has once again a choice of going straight to T or turning right to S. Since the timer at T is currently at 5 with a west-east red light and it would only take him 4 seconds to get to that intersection, Little Beaver knows that he would arrive at T and have to wait at a red light for 1 second, so going to T is not an option, which eliminates the answer P→R→T→SCHOOL. On the other hand, it would take him 6 seconds to reach S, and since the north-south light is green and the timer shows 8, he would still find the light green by the time he reaches S, and he would be able to turn left to get to school.

Therefore, the answer is: P→R→S→SCHOOL.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Data Interpretation, Specification, Algorithms, Implementation, Digital Systems, Interactions

In this problem, we performed a simulation of Little Beaver moving around town, keeping track of the timers and the green and red lights. At any point, the contents of the timer table, along with the time taken so far and the position of Little Beaver in the map, is called *the state of the system*.

We also used an Informatics concept called **Graphs**. Each intersection is called a *vertex* and each street segment connecting those intersections is an *edge*. In this problem, we were asked to find a path through the graph that took us from Little Beaver's house to his School, taking into account the restriction of not finding a red light by the time beaver reached a certain vertex.

In our daily life, "planning" is very important. It helps us do things more efficiently. Sometimes, we may have to plan the best traveling route like this task; sometimes, we may have to plan what to put in our backpacks.

This question comes from the United Kingdom

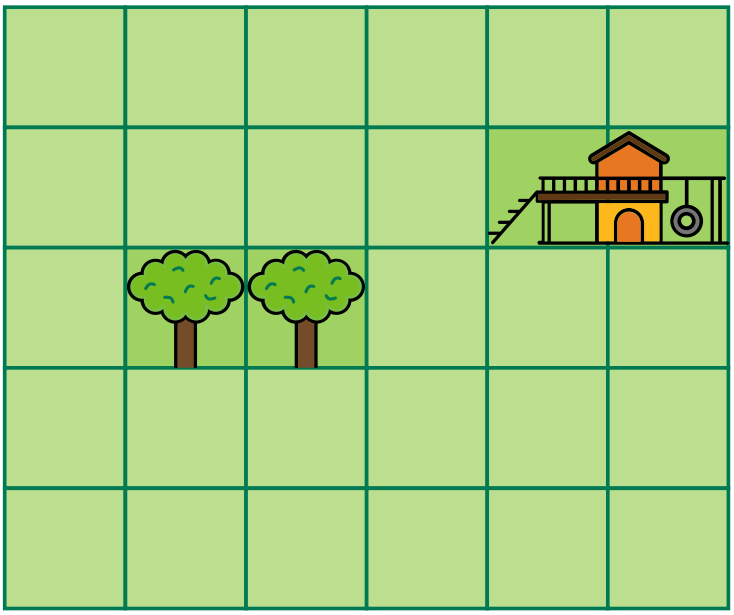
Squares

The beavers are playing a game called squares, and they all want to join in!

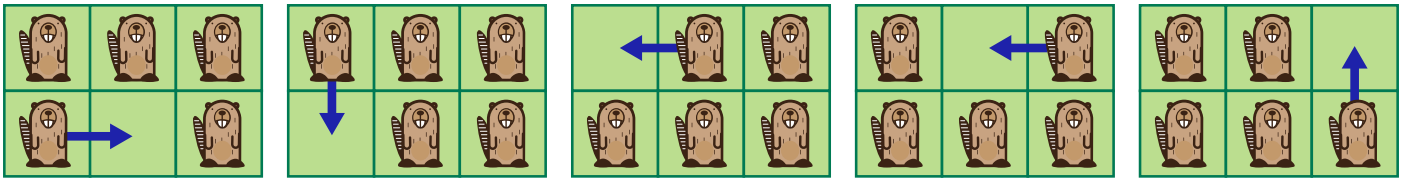
The park has a grid marked as shown below. The park also contains a playhouse and some trees. No beavers can enter the squares where the house or trees are.

There are 26 empty squares to start with.

The object of the game is for beavers to enter an empty square and then for everyone to try and move around so that they can all visit every square, but there must never be more than one beaver in any square at any one time. The rules also state that they cannot move diagonally.



Example movement showing how beavers can shuffle around:



Question

What is the maximum number of beavers that can fit in the park and still make it possible for the beavers to play the game? It must be possible for ALL beavers to be able to move to any other square in the park by shuffling other beavers out of the way.

13 | 23 | 24 | 26

Continued on next page

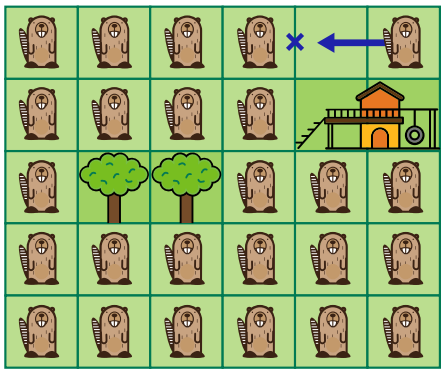
Squares – continued

Answer

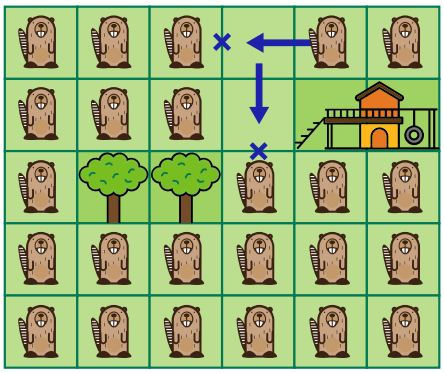
The correct answer is 23.

The key to this problem is recognising that this situation is similar to one of the sliding tiles puzzles found in toy shops. In those games all tiles can be moved to any position if only one is left empty. There will be gridlock if all squares are full but you only need to have one empty square to allow all the beavers to shuffle about. This immediately eliminates 26 as an answer.

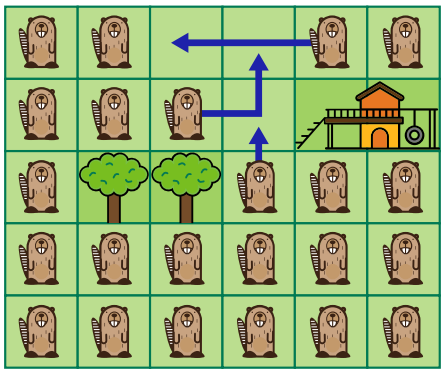
However, in computing we must consider unusual circumstances when looking at problems. The two squares at the top right behind the playhouse will trap any beavers on these squares if there is only one empty square or only two empty squares. Even when there are two empty squares, the top rightmost beaver can visit three squares (3 right top squares). Thus, it is not possible to play this game with 24 beavers in the park.



[One empty square]



[Two empty square]



[Three empty square]

As shown in the pictures above, at least three empty squares will allow the other beavers to shuffle around and allow beavers behind the house to escape into the rest of the play ground. So the correct answer is 23.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Data Interpretation, Specification, Algorithms

This is a problem where students need to apply their analysis skills to solve it. First they need to look at the overall problem but then, as is common when solving computational problems, the special cases or 'boundary data' has to be looked at carefully to see if it requires the initial solution to be modified. These kinds of skills are particularly useful in software testing. In this task the two squares at the top right are the special cases and boundary data to check carefully.

This question comes from Cyprus

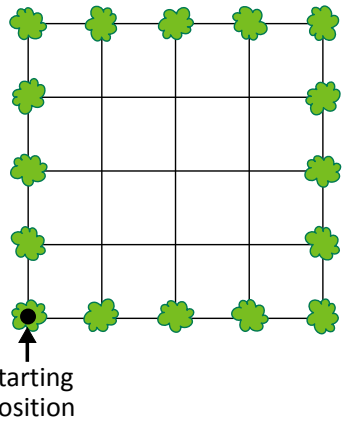
Years 3+4
Years 5+6
Years 7+8
Years 9+10 Hard
Years 11+12

Robot Tree Planter

A robot helps plant trees.

The robot programming language consists of the following commands:

- **Start:** Turns the robot on.
- **Forward(X):** Moves the robot forward by X meters.
- **Backward(X):** Moves the robot backward by X meters.
- **Left(X):** Turns the robot X degrees to the left.
- **Right(X)**: Turns the robot X degrees to the right.
- **Plant:** Plants a tree.
- **Repeat X {commands}:** Repeats the commands in the brackets X times.
- **Stop:** Turns the robot off.



The programmer wants 16 trees to be planted on the edge of the field, as shown in the diagram. Each side of the field is 8m long and consecutive trees will be separated by a distance of 2m. The robot starts at the starting position, facing the same direction as the arrow (up in the diagram). The robot can pass by an already planted tree.

Question

Which of the following programs will allow the robot to plant all of the trees in the figure?

```
Start
Repeat 4{
  Repeat 4{Plant, Forward(2)},
  Right(90)}
Stop
```

```
Start
Repeat 4{
  Repeat 4{Plant, Forward(2)},
  Left(90)}
Stop
```

```
Start
Repeat 4{
  Repeat 4{Forward(1), Plant},
  Right(90)}
Stop
```

```
Start
Repeat 4{
  Repeat 4{Forward(2), Plant},
  Left(90)}
Stop
```

Answer

The correct answer is:

```
Start
Repeat 4{
  Repeat 4{Plant, Forward(2)},
  Right(90)}
Stop
```

The robot will start from Repeat 4{Plant, Forward(2)} – this will plant the first 4 trees above it, leaving a 2m gap between each. Then it will turn right after executing Right(90)

It will repeat these commands for another 3 times according to Repeat 4, planting all 4 sides of the square.

Answer C will not work because the Forward(1) does not advance the robot the far enough to reach the next tree.

Answers B and D will not work because they turn the robot Left(90), which turns it away from the field if it is pointing in the direction of the arrow.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Specification, Algorithms, Digital Systems

The task is about understanding algorithms as implemented in computer programs. Here, the algorithm involves a sequence of steps to be carried out by the robot to complete the desired task. Thinking about such sequences of steps is an important skill for programming robots.

Notice also that there is a loop inside another loop, what is called nested loop. Normally, a loop is used for iterative matters. When you have a plain array or only one dimension, you will use a simple loop. But in the case of working in two dimensions (like in a matrix), you will need a nested loop.

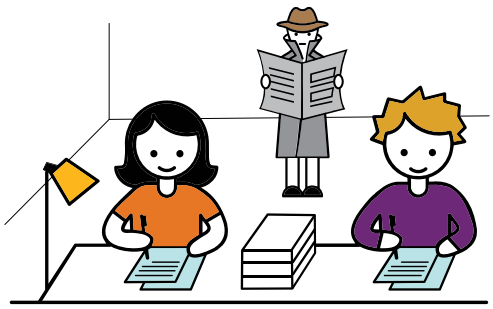
This question comes from the United Kingdom

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Hard
Years 11+12

Secret Siblings

You have intercepted a secret message that has been encrypted!

Dave sent it to his sister Janet. The message is:



First, see if you can decode the message.

Question

Using the same encryption system that Dave used, encode the message 'eggs'.

(Hint: Make sure you encode the message 'eggs' and do not decrypt it!)

Answer

The correct answer is: ehiv

Decrypting coded messages requires searching for patterns and clues. The enigma code for a given day was often solved by using cribs. Cribs are guessed words or phrases that you hope might appear. At one point in the Second World War it was very helpful to the decoders at Bletchley Park that a coded, formal weather forecast was produced at roughly the same time each day. This meant that they could look for words like "wetter" – the German word for weather. In Dave's message it quickly becomes clear that Dave has signed his name.

Further analysis shows that the letters have been shifted in the alphabet like this:

- D +12 becomes P
- a +13 becomes n
- v +14 becomes j (after wrapping round)
- e +15 becomes t

Tracing back, we can see that the message starts with the correct letter and then each subsequent letter is shifted by one additional position, giving us the decoded message as:

What is for tea?
– Dave

Doing the same for "eggs" gives:

- e +0 = e
- g +1 = h
- g +2 = i
- s +3 = v

ehiv is the answer to the problem.

As has happened many times in history, a very clever encryption system has been compromised by a less than clever use of it.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Algorithms, Evaluation

Concepts: Specification, Algorithms, Digital Systems

This task is about spotting patterns. Data encryption and decryption are important areas of study among computer scientists!

This question comes from Germany

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Hard
Years 11+12

Lost Car

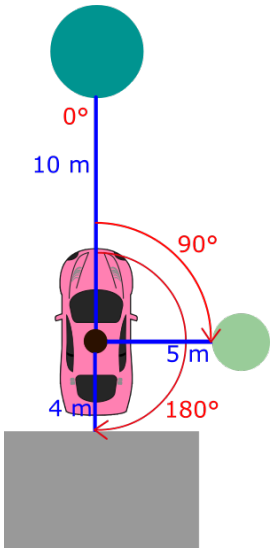
Bad luck! An autonomous car did not return home but stayed somewhere in the city.

Just before the battery died, the autonomous car found a parking place and sent home the positions of some (but not all) of the objects close to the car which were recognised by the sensor. Each object is represented by two values:

1. the angle (related to the 360°-sensor on top of the car; the angle 0° is in front of the car – look at the picture) and
2. the distance of the object to the car's sensor.

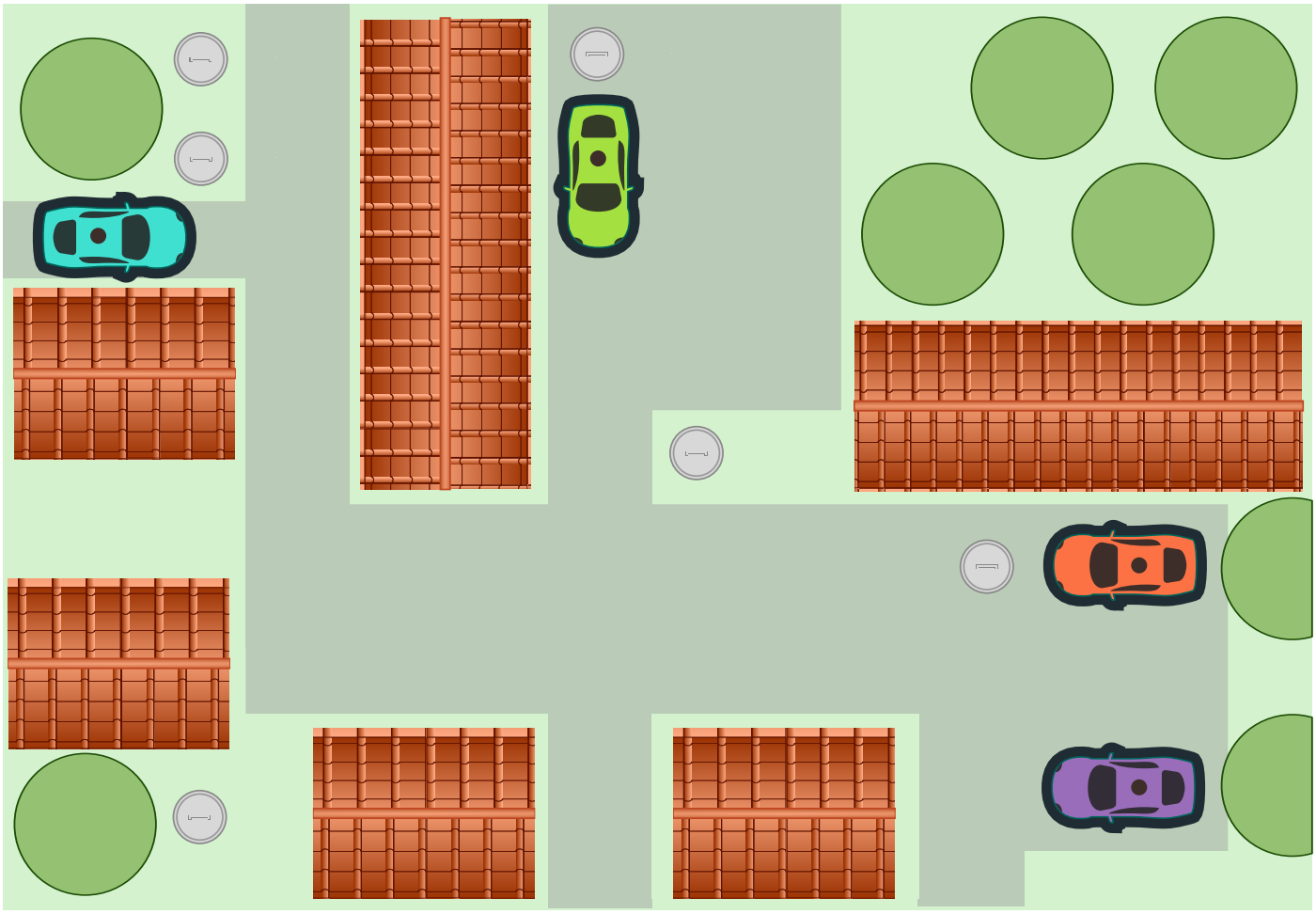
In the example below, the car records: [(0, 10), (90, 5), (180, 4)]

The lost car sent this representation of its physical environment: [(0, 5), (90, 4), (180, 5), (270, 12)]



Question

Find the lost car on the map!



Continued on next page

Lost Car – continued

Answer

The red car (in the middle of the right side on the map) is the lost car.

The lost car has an object 5m in front of it, an object 4m away on its right side, an object 5m behind it and an object with distance 12m away on its left side.

The grey car is not the lost car, because the objects on its right and left side are equally close.

The yellow car is not the lost car, because the object behind it (green circle) is much closer than the object in front of it (grey block).

The blue car is not the lost car, because the object in front (green circle) is much closer than the object behind (small black circle).

The red car is the lost car: the objects in front and behind are equally close, and the object on its right side (grey block) is much closer than the object on its left side (blue car).

It's Computational Thinking

Computational Thinking Skills: Decomposition, Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Data Collection, Data Representation, Data Interpretation, Specification, Algorithms, Digital Systems, Interactions, Impacts

Autonomous cars use the LIDAR (light detection and ranging) method to scan their environment, involving laser technology. Their navigation software creates a complex 3D model of all objects up to a distance of several hundred meters.

In contrast, the model in this task is very simple, and records only the closest objects. In general, a model is an abstraction of reality, and includes only those aspects of reality that are important for a certain purpose. In case of autonomous driving, the model can be used for collision prevention. For this purpose the relative location of surrounding objects is needed. Other aspects like the color are not important and can be ignored.

This question comes from Romania

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Hard
Years 11+12

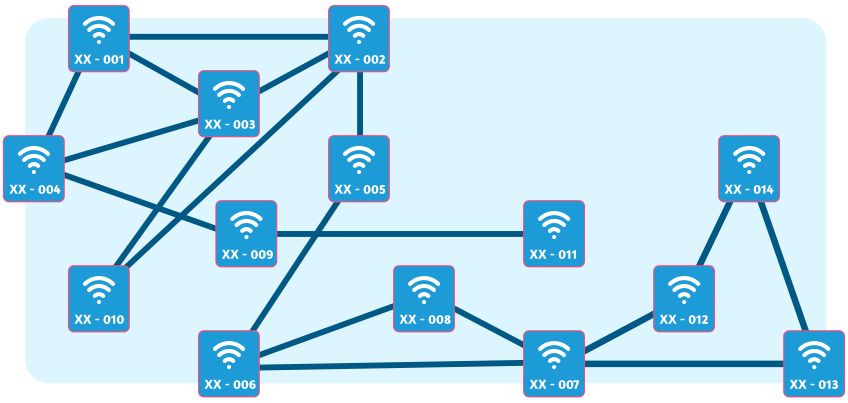
Key Points

This is a Wi-Fi network in a small company that contains 14 intranet access points. In this network some of the access points are called “key points”. This refers to access points which, once eliminated (are broken), determine the loss of access to the intranet for other access points.

For example, access point XX-009 is a “key point”. If XX-009 was broken, XX-011 will no longer have access to the intranet.

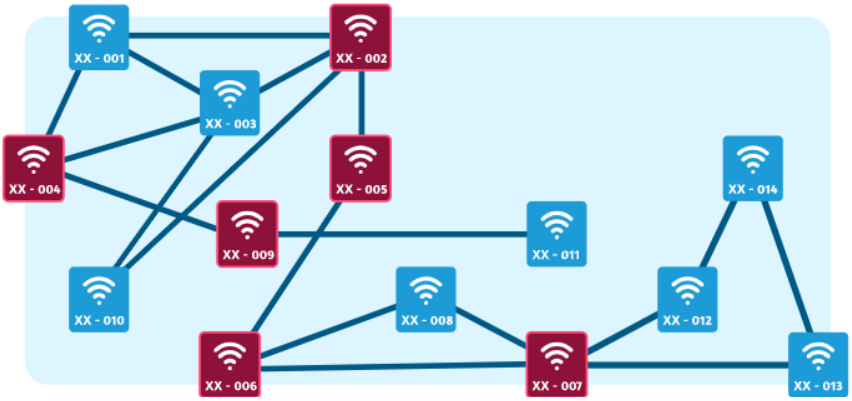
Question

Which access points are the “key points”?



Answer

The correct answer is: XX-002, XX-004, XX-005, XX-006, XX-007, XX-009, as shown in the diagram below.



It’s Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Data Representation, Data Interpretation, Specification, Algorithms, Implementation, Digital Systems, Interactions, Impacts

In Computer Science, it is common practice to represent networks by using graphs as data structures, which are simply sets of connections. The task consists of finding, in a connected graph, the vertices that are separating the graph into 2 or more related components, called articulation points. In order to determine the articulation points within a non-oriented graph, it used a modified depth-first search algorithm, storing additional information for each node.

Let’s presume that T is a graph. Then, a node v is an articulation point if:

1. v is the root of T and v has two or more children; or
2. v is not the root of T and has a child u in T so that no node in the sub-graph dominated by u is connected with an ancestor of v through a back edge (his children can not reach on a higher level in the graph by using another path).

This question comes from Slovenia

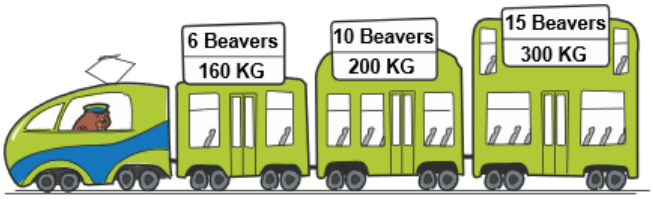
Years 3+4
Years 5+6
Years 7+8
Years 9+10 Hard
Years 11+12

Train Trip

Eight beaver families would like to go on a train trip. The train has a limited number of seats and limits the weight of luggage in each carriage. The families' data is presented in the following table:

Family	Number of members	Luggage weight [kg]
Avsec (A)	3	50
Bizjak (B)	4	80
Cerar (C)	5	110
Dolenc (D)	4	80
Erjavec (E)	2	40
Furlan (F)	3	70
Gabric (G)	6	130
Hacin (H)	5	100

The train company has the following train carriage composition. The number of available seats and the luggage limits for each carriage is written on the diagram:



Question

How many families can fit on the train if:

- every beaver must sit on their own seat,
- all members of the same family must sit in the same carriage,
- the luggage has to be in the same carriage as the family members, and
- the weight of the luggage has to be within the limits of each carriage?

Answer

The correct answer is 7.

There are 32 members in all the families combined, while the train has only 31 seats. Therefore it is not possible for all 8 families to go on the train.

One of the options for 7 families to go on the train trip whilst meeting the constraints is:

- 1st carriage: family G – 6 beavers, 130 kg
- 2nd carriage: families A, C, E – 10 beavers, 200 kg
- 3rd carriage: families B, D, H – 13 beavers, 260 kg

It's Computational Thinking

Computational Thinking Skills: Decomposition, Abstraction, Algorithms, Evaluation

Concepts: Abstraction, Data Interpretation, Specification, Algorithms

You are given a number of families with how many members are in each and the weight of each family's luggage. There are limits not only on seats (people) in total, but also within each carriage. In addition carriages have weight restrictions and families must not be separated. A feasible solution is any solution that meets the constraints. From the feasible solutions, the best (or optimal) solution is one that not only meets the constraints but also allows the largest number of families that to travel.

This problem is related to the knapsack problem which is a well known problem in combinatorial optimisation.

Bebras Challenge 2021 Round 2

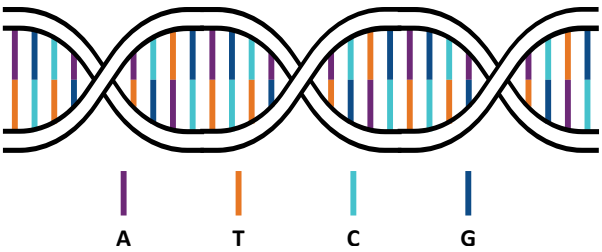
Years 11+12

This question comes from Indonesia

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Easy

DNA Sequence

Every creature has a sequence of DNA that determines the genes of a creature. The sequence of DNA is formed from several nitrogen bases which can be formed from 4 types, namely Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). DNA can mutate to form a new sequence that is different from the original sequence.



A Vormi is a creature that can mutate in 3 ways, as follows:

1. Substitution: Changes one base component with another base component in the DNA sequence.
   Example: AGGTC becomes AGGTA (change C to A)
2. Deletions: Elimination of one of the base components of the DNA sequence.
   Example: AGGTC becomes AGTC (delete one G)
3. Duplication: Addition of base components to the DNA sequence repeatedly.
   Example: AGGTC becomes AGGTTC (duplicate T)

Question

Below are four DNA sequences of Vormi creatures. Which sequence cannot be generated from 3 gene mutations if its initial DNA sequence was "GTATCG"?

"GCAATG" **"ATTATCCG"** **"GAATGC"** **"GGTAAAC"**

Answer

The correct answer is "GGTAAAC" (Option D).

- Option A: substitution (T to C – GCATCG), substitution (T to A – GCAACG), substitution (C to T – GCAATG)
- Option B: substitution (G to A – ATATCG), duplication (T – ATTATCG), duplication (C – ATTATCCG)
- Option C: substitution (T to A – GAATCG), substitution (C to G – GAATGG), substitution (G to C – GAATGC)
- Option A, B and C require 3 steps of gene mutation, as shown above while option C needs to undergo at least 4 steps: duplication (G, GGTATCG), duplication (A, GGTAATCG), substitution (T to A, GGTAAACG), deletion (G, GGTAAAC)

It's Computational Thinking

Computational Thinking Skills: Decomposition, Algorithms, Evaluation

Concepts: Data Interpretation, Specification, Algorithms

To solve this problem, we need to find the distance between one word (DNA sequence) and another. We have to try all the choices and calculate the minimum distance of each given DNA sequence. To find minimum distance between two words is known as Levenshtein distance.

Levenshtein distance may also referred as edit distance because this metric measures the minimum number of single-character edits. The known operators for this metric are insertion, deletions, dan substitutions. But in this task, the insertion operation is replaced by duplication operation.

In this task, we can assume the DNA sequences is one string and mutated DNA sequences is another string. Then we can measure the Levenshtein Distance between these two strings. The combination which doesn't have Levenshtein Distance equal to 3 is the answer.

This question comes from India

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Easy

Dinner Table

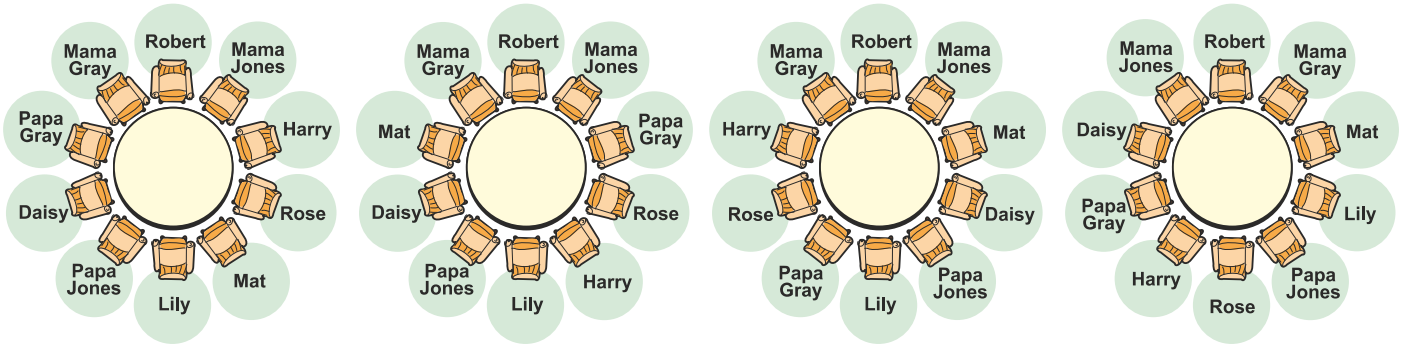
Mat is hosting a dinner party for some beaver families. Mat has invited his friend Robert and two other beaver families. Mama and Papa Jones with their son, Harry and their daughter, Rose and Mama and Papa Gray with their daughters Lily and Daisy.

Mat has laid down some rules for the seating arrangement at the round dinner table to encourage the beavers to start new conversations.

- Two female beavers cannot sit next to each other.
- Mama and Papa beavers in the same family cannot sit next to each other.
- Young beavers cannot sit next to their parents.

Question

Which of the following is a correct seating arrangement according to Mat's rules?



Answer

The correct answer is B.

In option A, Mama Gray and Papa Gray are seated next to each other which breaks rule #2. Daisy is also seated next to her father, Papa Gray which breaks rule #3.

In option C, Lily is seated next to Papa Gray, her father, which breaks rule #3.

In option D, Daisy is seated next to Mama Jones which breaks rule #1. Rose is seated next to her father, Papa Jones and Daisy is seated next to her father, Papa Gray which breaks rule #2.

All the rules (constraints) are satisfied by option B so that is the correct answer.

It's Computational Thinking

Computational Thinking Skills: Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Data Representation, Data Interpretation, Specification, Algorithms, Implementation, Digital Systems

The Dinner Table problem is in fact a simpler version of a more general 'constraints' problem, where we are given a set of constraints (rules or restrictions), a domain of values, and must find a solution that satisfies all the constraints.

We do this by creating a model from the data and then build the solution incrementally. We decide the variables, what values they can take, and which constraints apply. In this problem, the variables are the seats occupied by each person.

We then try a step-by-step solution by assigning valid values to one variable at a time, 'propagating' the effect of the assignment to other variables and proceeding. In some cases, if an assignment results in failure, we backtrack, assign a different value and try again.

In this problem our task is easier: we are given the values – in the form of multiple choice answers – and just have to verify whether or not they meet all the constraints.

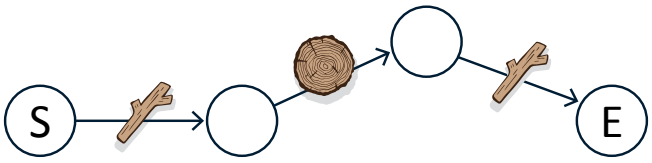
Passwords

Beavers make a set of passwords to secure their lodge. The passwords consist of only two symbols:  and 

A password checker makes sure that a given password is acceptable.

Example

A checker:



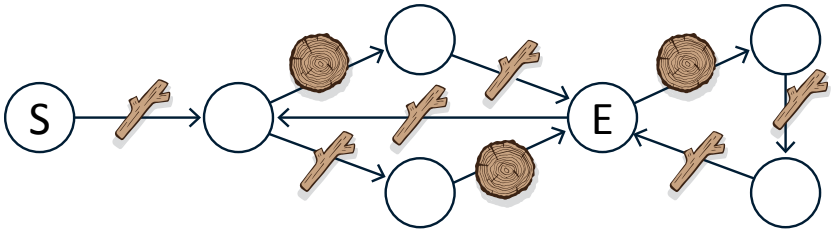
A provided password that will be accepted: 

The beavers use diagrams with circles and arrows to describe how a checker works:

- A checker reads in the symbols from the provided password, one symbol at a time, from left to right.
- A checker always starts at the circle labelled “S”.
- At each circle, the checker reads one symbol.
- If at the current circle there is an outgoing arrow labelled with the current symbol, the checker follows that arrow to the next circle; otherwise the checker stops and does not accept the password.
- If there are no more symbols to read, the checker stops. The checker will only accept the password if it stops at the circle labelled ‘E’.

Question

The beavers come up the following password checker.



From which set(s) of symbols can you create a password that the checker will accept? You must use all of the symbols in the set to create a password.

Continued on next page

Passwords – continued

Answer

Answers 4 x  8 x  **and** 2 x  4 x  **are correct. Accepted passwords will always consist of twice as many**  **as**  **.**

Answer 5 x  8 x  is incorrect: The password consists of 13 symbols, but for a password to be accepted, the length must be divisible by 3.

Answer 5 x  7 x  is incorrect:  appears 7 times, and  appears 5 times, but accepted passwords must consist of twice as many  as  .

It's Computational Thinking

Computational Thinking Skills: Pattern Recognition, Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Data Collection, Data Representation, Data Interpretation, Specification, Algorithms, Digital Systems

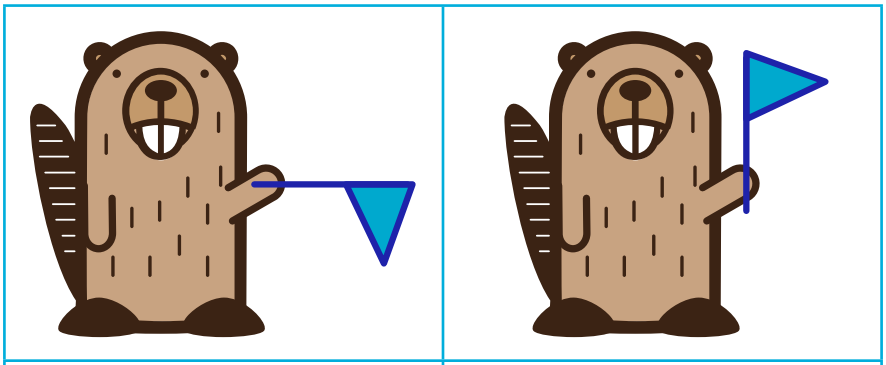
The password checkers shown in this Bebras task are each modeled as a deterministic finite-state machine (FSM); this is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any time. The FSM can change from one state to another in response to some inputs; the change from one state to another is called a transition. An FSM is defined by a list of its states, its initial state, and the inputs that trigger each transition.

The behavior of state machines can be observed in many devices in modern society that perform a predetermined sequence of actions depending on a sequence of events presented to them. Examples are:

- vending machines, which dispense products when the proper combination of coins is deposited,
- elevators, whose sequence of stops is determined by the floors requested by riders,
- traffic lights, which change sequence when cars are waiting,
- combination locks, which require the input of a sequence of numbers in the proper order.

Flag Semaphore

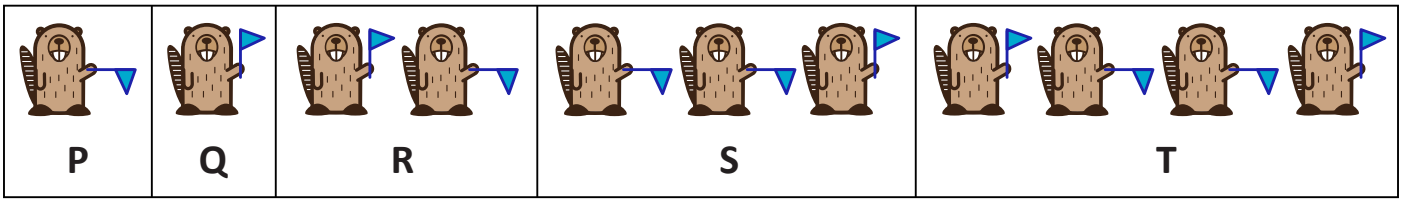
Beavers in the town of Achi communicate by holding flags. They either hold the flags horizontally or vertically.



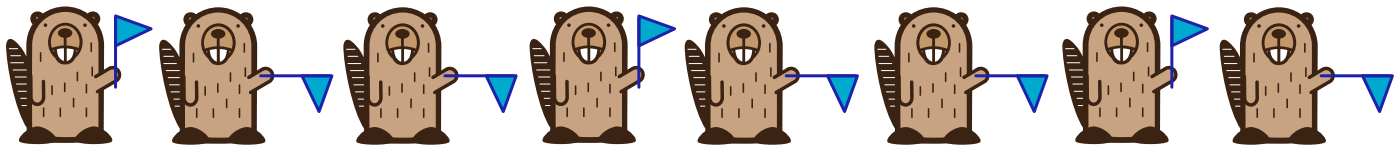
Position 1: Horizontal	Position 2: Vertical

The beavers use this system to send five different letters, P, Q, R, S and T to their friends.

They send each letter by holding the flags in different positions one after another in the following way:



Beaver Adanma shows their friend the following combination of flag positions:



Question

Which of the letter combinations below did she send?

TSQ **RPQSR** **RPSP** **QPPTP**

Continued on next page

Flag Semaphore – continued

Answer

The correct answer is QPPTP.

The horizontal position can be encoded as 0

The vertical position can be encoded as 1

So a series of flag positions: vertical, horizontal, horizontal, vertical, horizontal, horizontal, vertical, horizontal, can be encoded as 10010010. The letter P is encoded as 0, the letter Q is encoded as 1, the letter R is encoded as 10, the letter S is encoded as 001, and the letter T is encoded as 1001. Therefore, only the option (D) is correct. 1(Q)0(P)0(P)1001(T)0(P). All the other options contain an error.

TSQ → 10010011 (error)
RPQR → 100100110 (error)
RPSP → 1000010 (error)

It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Abstraction, Algorithms, Evaluation

Concepts: Abstraction, Data Interpretation, Specification, Algorithms

Sending messages using flag positions is an old practice in naval communication (this is called the semaphore system of communication). To send messages in this way, the code used must be unambiguous. (The code is the system used to assign sequence of flag positions to individual letters.) One way to make a code unambiguous is to have it be prefix-free. This means that no code of a letter is a prefix of code of some other letter.

In the problem, the codes are not prefix-free because:

- the code for P is a prefix of the code for S,
- the code for Q is a prefix of code for R and of code for T, and
- the code for R is a prefix of code for T.

The code of the problem is ambiguous. In the task the transmitted message can be decoded in different ways:

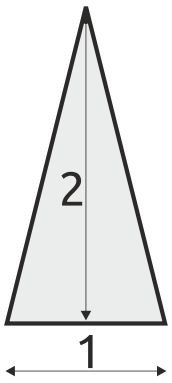
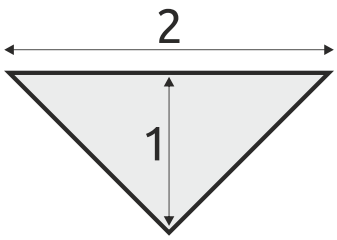
QPPQPPQP, QSSP, QPPTP, TSP, RPQSP, RPTP, etc.

Therefore, it is easy to cause a situation where the message is misinterpreted. Note, that if codes are of the same length, the property of prefix-free is satisfied when all codes are different.

Encoding and decoding belong to the computer science domain called information theory.

Interpret Programs

The following commands will draw the triangles shown beneath them.

draw (up, 1, 2)	draw (down, 2, 1)
	

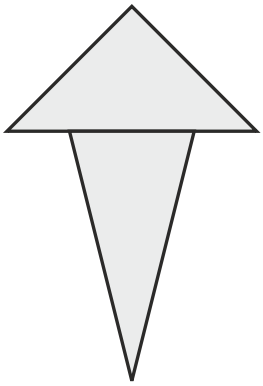
After the triangle has been drawn, the drawing tool returns to the starting point.

Question

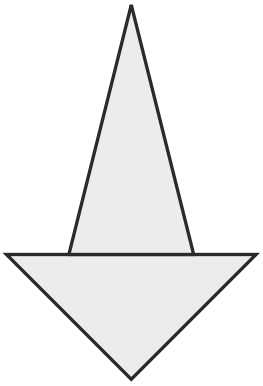
Below there are four sets of commands:

1	2	3	4
draw (down, 2, 1) draw (up, 1, 2)	draw (down, 1, 2) draw (up, 2, 1)	draw (down, 2, 1) draw (up, 2, 1)	draw (down, 2, 1) draw (up, 1, 1)

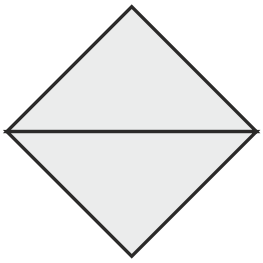
Match each set of instructions to the picture it will draw.



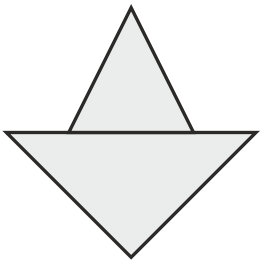
A



B



C



D

Continued on next page

This question comes from
Lithuania

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Easy

Interpret Programs – continued

Answer

Command set 1 will draw option B.

First the “flat” triangle is drawn down, then the “thin” triangle is drawn up.

Command set 2 will draw option A.

First the “thin” triangle is drawn down, then the “flat” triangle is drawn up.

Command set 3 will draw option C.

It draws two “flat” triangles, one up and one down.

Command set 4 will draw option D.

First “flat” triangle is drawn down, then the smaller “thin” triangle is drawn up.

It’s Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Algorithms, Evaluation

Concepts: Data Interpretation, Specification, Algorithms

In Computer Science, in programming we mostly use instructions again and again, sometimes making small changes. This task defines a function, a drawing procedure where we can use tree parameters. The drawing depends on these parameters. It is not important for us (who use the function) how it works, we know nothing about the main part, about inner instructions. We only “call” the function more times with different values – so we can draw different triangles.

The parameters and the method we use to pass them are important parts of CS.

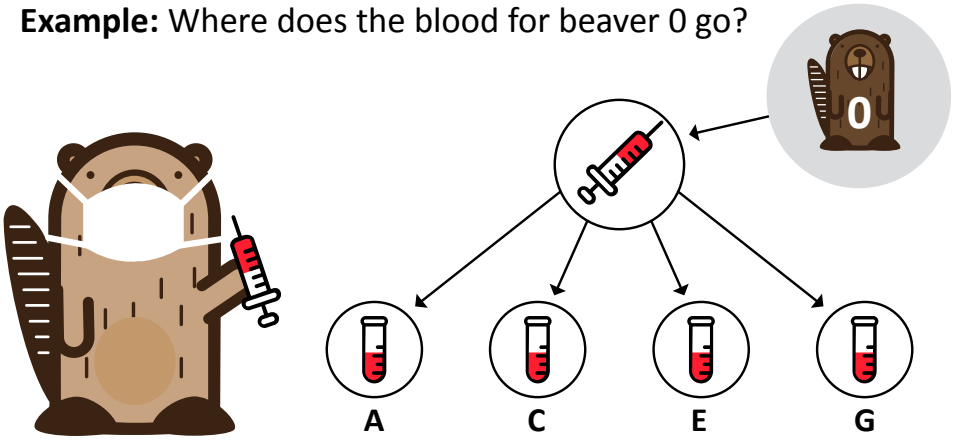
As we know, a computer is an information processing machine. Computers process data to produce information. In computing, a command is a directive to a computer program to perform a specific task. For example, to draw a triangle. Specifically, the term “command” is used in imperative or procedural computer languages like Python, Pascal, C++. The name arises, because statements are usually written in a manner to natural languages. Commands allow parameters or arguments. In computer programming, a parameter is a special kind of variable, used in a function, procedure, or routine to refer to one of the pieces of data provided as input.

Mixed Results

Dr Beverley knows that exactly one of her 16 beaver patients is sick. She has a blood sample from each patient, but she only has 8 test tubes A, B, C, D, E, F, G, and H. By mixing the blood from 16 patients into 8 test tubes, she knows she only needs to use 4 blood tests to find which of her 16 patients is sick.

To find the sick beaver, she puts takes a blood sample from each of her patients and places it into four different test tubes. She carefully follows the Test Tube Distribution plan. For example, beaver number 0 has their blood put into test tubes A, C, E, and G

Example: Where does the blood for beaver 0 go?



So far, Beverley has tested tubes A (infected), C (healthy), and E (healthy). She has only one test left.

Test results so far

Test tube **A** – infected

Test tube **C** – healthy

Test tube **E** – healthy

Test tube distribution

0	ACEG
1	ACEH
2	ACFG
3	ACFH
4	ADEG
5	ADEH
6	ADFG
7	ADFH
8	BCEG
9	BCEH
10	BCFG
11	BCFH
12	BDEG
13	BDEH
14	BDFG
15	BDFH

Question

Which of these 4 test tubes can Dr Beverley test to identify the sick beaver?

Test tube B | **Test tube D** | **Test tube F** | **Test tube H**

Continued on next page

Mixed Results – continued

Answer

The correct answer is: Test tube H

In the table above, for each patient the four columns marked with an 'X' show which test tubes have that patient's blood. Based on the results of each of the three tests so far, the coloured shading indicates which beavers are potentially the sick patient. For an infected test tube, all patients whose blood is in that test tube are possibly the sick patient. For a healthy test tube, all patients whose blood is ***not*** in that test tube are possibly the sick patient. On the basis of the tests so far, therefore, either beaver patient 6 or beaver patient 7 is ill. Patient 6's blood is in tube G, and patient 7's is in tube H. Since we know exactly one patient is sick, testing G or H (whether we get an infected or healthy result) will allow us to determine which of the two patients is sick. G is not one of the options given, therefore the correct answer must be H.

	A	B	C	D	E	F	G	H
0	X		X		X		X	
1	X		X		X			X
2	X		X			X	X	
3	X		X			X		X
4	X			X	X		X	
5	X			X	X			X
6	X			X		X	X	
7	X			X		X		X
8		X	X		X		X	
9		X	X		X			X
10		X	X			X	X	
11		X	X			X		X
12		X		X	X		X	
13		X		X	X			X
14		X		X		X	X	
15		X		X		X		X

Infected **A**
Healthy **C**
Healthy **E**

← These two beavers meet all conditions

It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Algorithms, Evaluation

Concepts: Abstraction, Data Interpretation, Specification, Algorithms

Group Testing is a technique that can be traced back to blood tests by Richard Dorfman during World War II. Since then, the technique has been applied to many areas, not only in biology, but also engineering and computer science, particularly in relation to defect detection.

There are several algorithms for group testing. Which one is most suitable depends on the particulars of the problem being tackled. The *generalised binary-splitting algorithm* is a good point for the interested reader to explore once they are comfortable with binary search.

The test-tube distribution shown in this example is actually based on the binary representation of the beaver number ($0=0000_2$,$1=0001_2$,...,$15=1111_2$) where A,C,E,G are all 0 and B,D,F,H are 1.

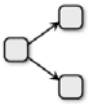
This question comes from Italy

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Medium

Production Units

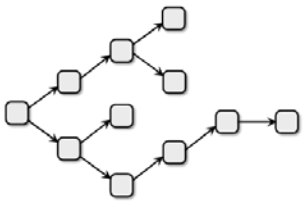
A factory produces several kinds of products. Each product undergoes a series of processes before it is finished. These processes occur in different units. All production processes at the factory are summarised in a diagram, where a square represents a unit (▢), and an arrow (⟶) represents the flow of a product from one unit to the next.

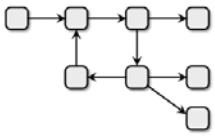
There are two types of units: production and packaging units. The rules that are used to set up the factory are as follows:

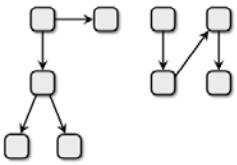
- From each production unit, products can be sent to two other units at most ().
- From the packaging unit, products are sent out of the factory. This unit is shown with no outgoing arrow, only a receiving arrow (i.e. ⟶▢).
- The name of the unit where all production starts is called Unit 1. There is only one such unit.
- Every unit apart from Unit 1 must recieve products from exactly one other unit.

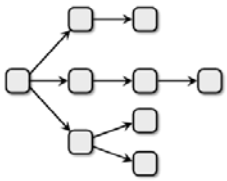
Question

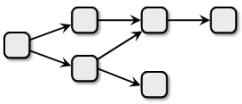
Which one of the following diagrams shows the factory's production processes correctly?





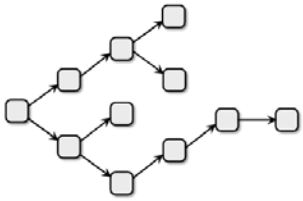






Answer

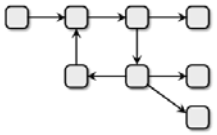
The correct answer is :



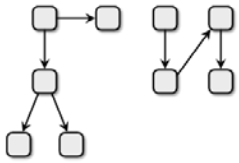
The leftmost unit is Unit 1; each unit receives products from only one unit; the packaging units are those without outgoing arrows; the other units send products to either one or two other units;

Continued on next page

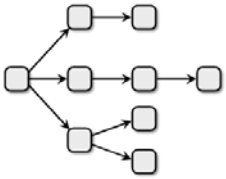
Production Units – continued



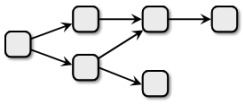
is not correct because there is a unit that receive products from multiple units, which violates the rules that were used to set up the factory. Additionally, there is one unit that sends products out to three other units, which also violates the rules.



is not correct because there are multiple units that could be Unit 1, which is where all production starts and has no input arrows. This is in violation of the rule indicating that only one unit can act as Unit 1.



is not correct because there are squares with more than 2 outgoing arrows – specifically Unit 1, which sends products out to 3 different units. This violates the rules of the factory.



is not correct because there is a unit on the upper line that receives products from more than one other unit, which is in violation of the rules.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Specification, Algorithms, Digital Systems, Interactions

In Computer Science, the diagrams depicted in the task are called *graphs*. This is a data structure that can be used to represent a *binary relation* between entities. In this case the entities are the production units and two units are related whenever one unit can send the products to the other unit. Graphs are often used in computer science to model different situations; for instance networks are usually modeled using graphs. Modeling a real situation with abstractions like graphs allows computer scientists to apply graph properties and graph algorithms to many different real-life problems.

The properties of the production process determine the properties of the graph. In this case the graph is a *tree* and, in particular, it is *connected* and it has no *cycles*. Trees can be used to model hierarchical relations, as in a family tree or in an organisation chart.

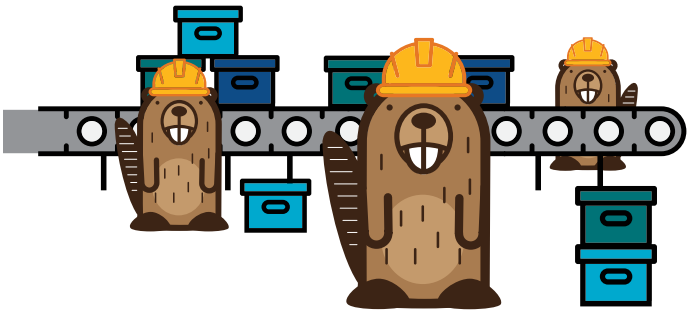
This question comes from Thailand

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Medium

Three Workers

Alice, Buddy and Catherine are the only 3 workers in a factory. They go to work according to certain rules. Each Monday, exactly 2 workers must go to work. The rules they follow when going to work are as follows:

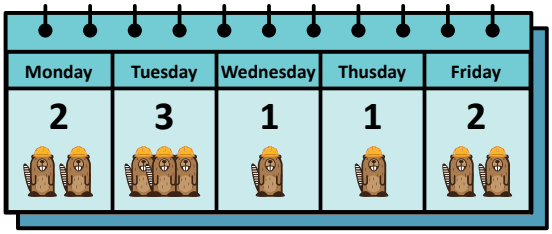
- Rule 1: if Alice goes to work, then Buddy will stay at home the next day.
- Rule 2: if Buddy goes to work, then Catherine also goes to work, but she will stay at home the next day.
- Rule 3: if Catherine stays at home, then Alice will stay at home the next day.



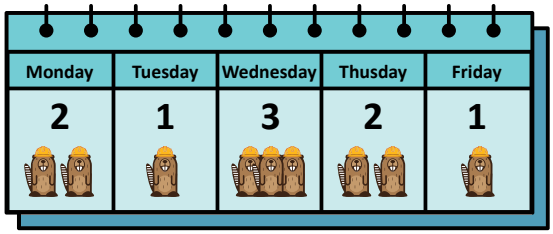
Question

Which of the weekly schedules below can be achieved with these rules?

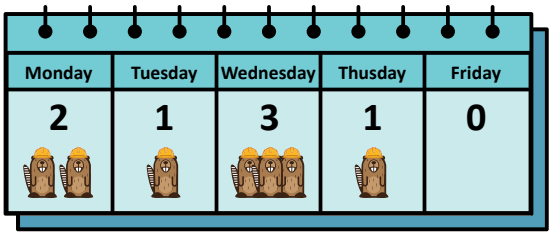
A



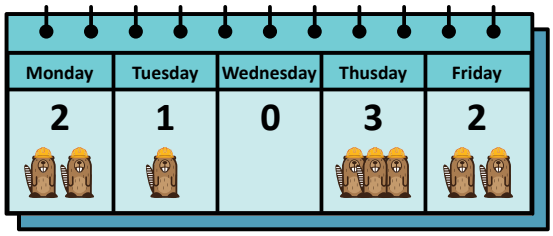
B



C



D



Answer

We can model the schedule by using letters to represent the workers – A for Alice, B for Buddy, and C for Catherine. We can start by checking which two workers can work on Monday. Given the 3 options, possible teams of 2 are A-B, A-C and B-C. We can exclude A-B, because Rule 2 would require that C is working as well and there must be 2 workers on a Monday, not 3.
So we start the week with either A-C or B-C:

Case 1:

	Monday	Tuesday	Wednesday	Thursday	Friday
At work	A-C	?	?	?	?
At home	B	B	?	?	?

Case 2:

	Monday	Tuesday	Wednesday	Thursday	Friday
At work	B-C	?	?	?	?
At home	A	C	A	?	?

Continued on next page

Three Workers – continued

In Case 1, B is not allowed to work on Tuesday per Rule 1. In Case 2, C is not allowed to work on Tuesday per Rule 2. In both cases, it is impossible to have 3 people working on Tuesday. This excludes answer a).

Case 2 also shows an application of Rule 3: A stays at home the day after C stays at home. So if there is a day where no one is working, all 3 workers cannot work the next day as we can be sure that A will be at home. This excludes answer d).

Answer b) works only until Wednesday: we can start as per Case 1 with A-C working. C can then work on Tuesday and everyone can work on Wednesday. So until Wednesday, the only solution would be:

	Monday	Tuesday	Wednesday	Thursday	Friday
At work	A-C	C	A-B-C	?	?
At home	B	A-B	–	B-C	A

But according to the rules, if all 3 work on Wednesday, then B (Rule 1) and C (Rule 2) are at home on Thursday. So, at most, A could work on Thursday. This excludes Schedule b) as well.

The only remaining possible schedule is c). Here is the only assignment that obeys all the given rules:

	Monday	Tuesday	Wednesday	Thursday	Friday
At work	A-C	C	A-B-C	A	–
At home	B	A-B	–	B-C	A-B-C

It's Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Specification, Algorithms

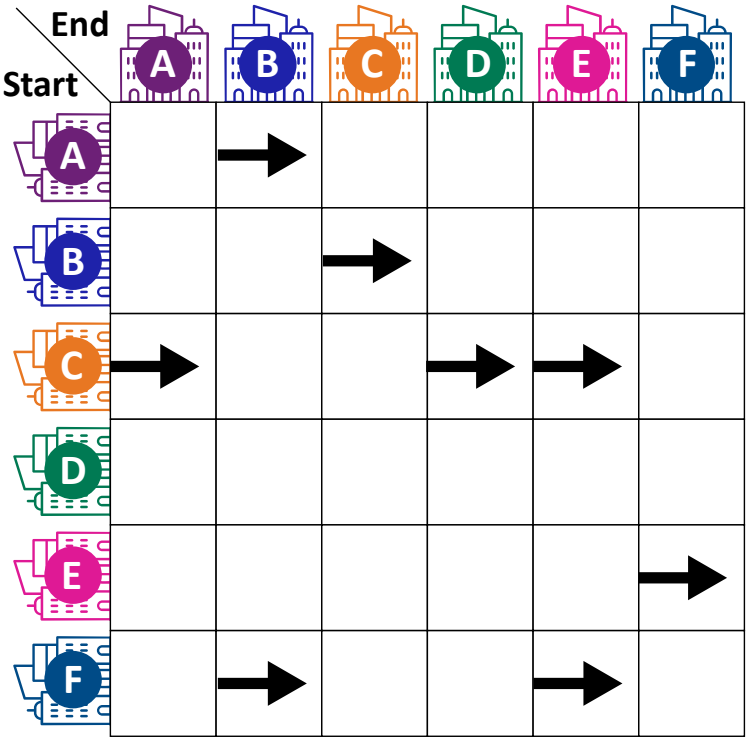
Several strategies can be used to solve this task. Demonstrated in the answer above is an example of *backtracking*: we make a hypothesis (for instance, "A and B are working on Monday") and check whether this works with the rules – in technical terms, whether it can *satisfy the constraints*. If so, we continue reasoning one step further under that hypothesis, possibly making other hypotheses along the way. If not, we go back ("backtrack"), make another hypothesis, and try to solve the problem with this new hypothesis.

For humans, it is difficult to keep track of where we are in the solving process and to be sure to make hypotheses in a systematic way that ensures that we reach the solution, should it exist. A good example of this is solving a sudoku. For more difficult sudokus, the solution can be difficult to find because it requires making (and, most probably, invalidating) many hypotheses along the way.

Computers, on the other hand, are very good at being systematic, and many concrete algorithms based on backtracking have been developed over the years to solve problems.

Decryption Map

A beaver signed up to play a map game. During the game, all the roads between cities were treated as one-way roads, but players were not given a map. Instead, they were given the table below.



There are six cities: A, B, C, D, E, and F.

When there is an arrow shown in the table, it indicates that there is a one-way road that starts from the city corresponding to the row and ends at the city corresponding to the column. When there is no arrow, it means that no road directly connects these two cities. For example, there is a one-way road from City B to City C but there is not a one-way road linking City B to City D.

Question

Which of the following statements is true?

The beaver needs to use at least three one-way roads to get from City C to City B.

The beaver needs to use at least three one-way roads to get from City A to City D.

The beaver needs to use at least three one-way roads to get from City A to City C.

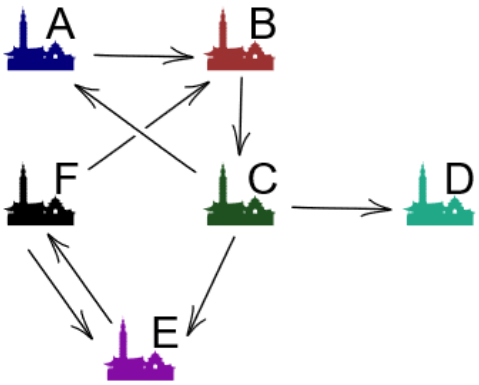
The beaver needs to use at least three one-way roads to get from City B to City E.

Continued on next page

Decryption Map – continued

Answer

The answer is: The beaver needs to use at least three one-way roads to get from City A to City D.



We can create the map showing one-way roads between cities as shown above. Now consider each of the given answers:

- The beaver can travel from City C to City A to City B using a total of only two roads. So this statement is false.
- From City A, the beaver can only go to City B and then only to City C. This means the beaver must use at least one more road to reach D, for a total of at least three roads. So this statement is true.
- The beaver can travel from City A to City B to City C using a total of only two roads. So this statement is false.
- The beaver can travel from City B to City C to City E using a total of only two roads. So this statement is false.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Data Interpretation, Specification, Algorithms, Digital Systems

The map shown in the solution is a *directed graph* (also known as *digraph*). A directed graph consists of *vertices* (cities in this task) and *directed edges* (one-way roads in this task) showing how the vertices are connected.

There are many other ways to represent a directed graph. For example, we can use an *adjacency matrix* much like the table in this task.

Representing data in multiple ways is very common in computer science. It is important to understand the advantages and disadvantages of each choice. For example, pictures as in the solution are visual and nice for humans to read but mathematical versions of the table can be easier for computers to manipulate.

This question comes from the United States

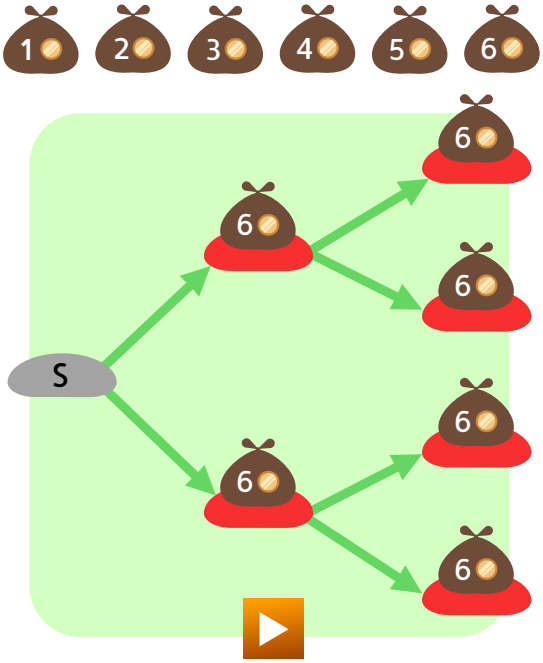
Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Medium

Coin Collector

Alice wants to play a game with her friend. She has made several paths in the woods with large stones along the way. At each stone, Alice has placed a bag of gold coins. The number of coins in each bag is anywhere between 1 and 6. Alice's friend wants to collect as many gold coins. He starts at the stone marked with an 'S'. At every stone, he is allowed to choose either the stone at the left path or the stone on the right path. He then moves to the stone that he has chosen and collects the coins.

Every time Alice's friend wants to select a stone, he is able to see how many coins are in the bag at each of the two stones. He came up with a greedy strategy to always choose the stone with more coins. Alice wants to teach her friend that his strategy does not always lead to getting the most coins.

Below is a map of the paths that Alice has made.



Question

Place the coins so that her friend's strategy will not result in them getting the highest number of coins. Note that there are multiple correct solutions.

Answer

There are many solutions to this problem. One way of looking at this is to force Alice's friend to make a wrong choice at the first decision. By making the first choice between '1' and '2', we can be sure that the friend will choose '2' because of his greedy strategy. Now, by putting the remaining bags with the highest numbers (5 and 6) behind the '1' choice, and putting the remaining low valued pouches (3 and 4) behind the '2' choice, we can force them to select 2 and then 4 coins, giving them a total of 6.

Had Alice's friend followed a different path, selecting the lowest number of coins first (1) and then the highest number of coins (6), they would have finished with 7 coins.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Specification, Algorithms, Implementation, Digital Systems, Interactions

A greedy algorithm is an algorithm where each time you have to make a choice, you make the 'locally optimal' choice. This means you don't look at the bigger picture, only at the information that is immediately near you. Sometimes greedy algorithms work well. But there are cases (such as this problem) where a greedy algorithm does not work.

In computer science it is important to create test data for your algorithms. This means you create some data to input to your algorithm to check if your algorithm is correct. Crafting good test data is a very important skill. With the wrong test data, your algorithm may appear to be correct, but that would only be on that specific instance.

Note that 'greedy algorithm' is not an algorithm; it is a technique used in an algorithm.

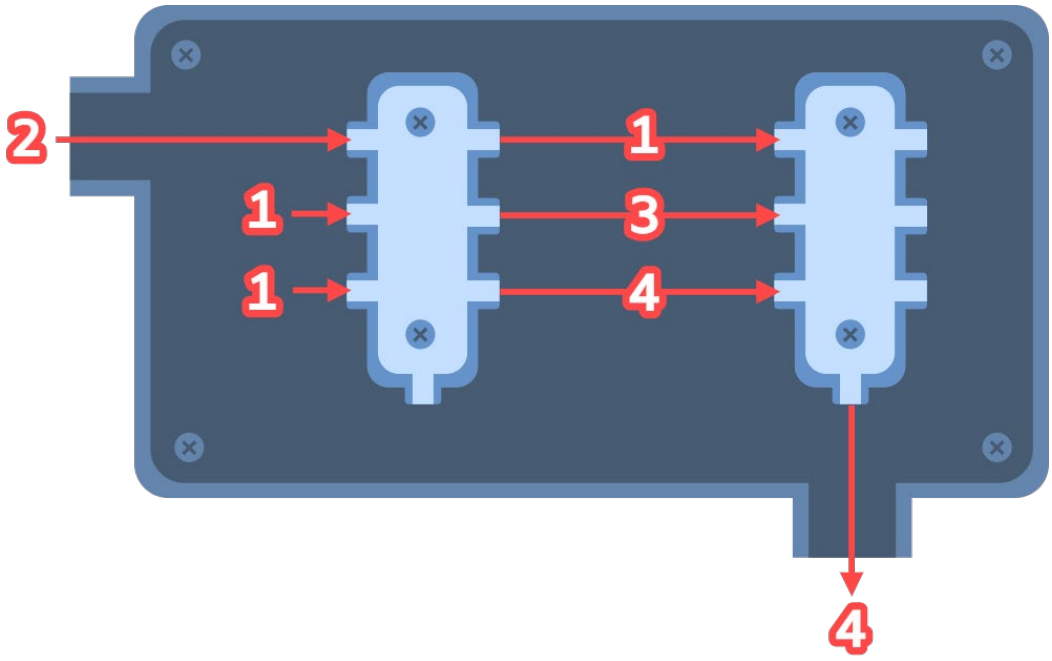
MathMachine

A group of beavers have created a MathMachine. It takes a number as input and returns another number as output. Inside, the MathMachine uses components. All components work in the same way. Each component takes three numbers as input, and processes them following these rules:

- If the first number is 1, return the third number to the MathMachine as output.
- Else:
  - Decrease the first number by 1. The result is the new first number.
  - Increase the second number by 2. The result is the new second number.
  - Add the new second number and the third number. The result is the new third number.
  - Pass the new numbers to the next component, in the same order.

When the MathMachine receives an input, it passes this number to the first component as the first number.
The other second and third numbers entered into this component are 1.
As soon as the MathMachine receives the output of any component, it returns this number as result.

The image below shows how the MathMachine processes the input 2, using two components in this case.



Question

The MathMachine processes the input 4. Which number does the MathMachine return as output?

7 | 10 | 16 | 64

Continued on next page

MathMachine – continued

Answer

16 is the correct answer, as shown by the process in the table below:

Component 1 Input	Component 2 Input	Component 3 Input	Component 4 Input
4	3	2	1
1	3	5	7
1	4	9	16

When one of the components recieves 1 as the input for the first number, the third number (in this case, 16) is used as the output of the MathMachine.

7 is incorrect: The “last” component returns the third number, not the second.

10 is incorrect: The components do not increase the result by 3 starting with 1.

64 is incorrect: The components do not multiply the result by 4 starting with 1.

It’s Computational Thinking

Computational Thinking Skills: Decomposition, Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Data Interpretation, Specification, Algorithms, Implementation, Digital Systems

A common method of simplification is to divide a problem into subproblems of the same type. As a computer programming technique, this is called divide and conquer and is key to the design of many important algorithms. Recursion in computer science is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem. Recursion solves problems by using functions that call themselves from within their own code. The approach can be applied to many types of problems, and recursion is one of the central ideas of computer science.

The power of recursion evidently lies in the possibility of defining an infinite set of objects by a finite statement. In the same manner, an infinite number of computations can be described by a finite recursive program, even if this program contains no explicit repetitions. Tail recursion is particularly useful, and often is easy to handle in implementations. Tail calls can be implemented without adding a new stack frame to the call stack.

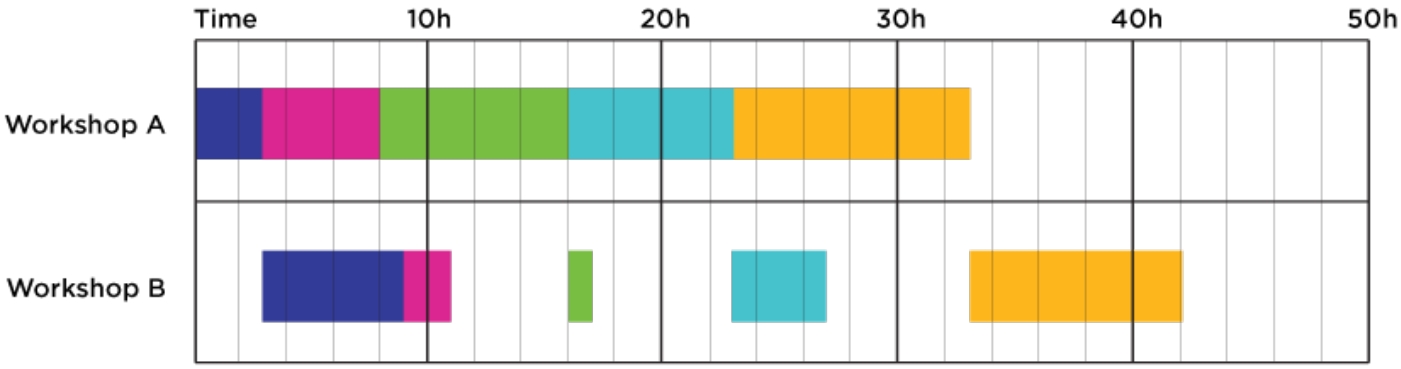
This question comes from China

Optimal Processing Flow

A car factory has received five part orders. These five parts need to be processed, first being produced in workshop A before being painted and polished in workshop B. You need to order the parts so that the work is completed as quickly as possible.

The processing time for the five parts in the two workshops is as follows:

Code	Part	Workshop A	Workshop B
1	rear	3h	6h
2	front	5h	2h
3	wheels	8h	1h
4	door	7h	4h
5	frame	10h	9h



In the example above, if parts are processed in the sequence 1-2-3-4-5, the total time will be 42 hours.

Question

What is the minimum time needed to fully process all 5 parts?

33 | 34 | 36 | 40

Continued on next page

This question comes from China

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Hard

Optimal Processing Flow – cont.

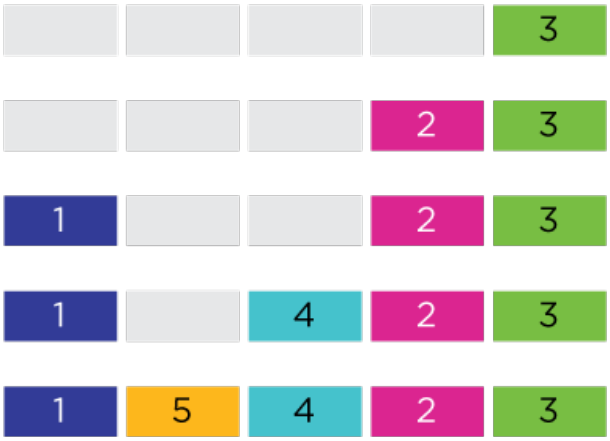
Answer

The correct answer is 34.

The part processing queue can be formed as follows:

First, select the part with the least processing time in either workshop. If this part's shortest time is in workshop A, we put the part in the first available space in the queue. If the part's shortest time is in workshop B, put the part in the last available space in the queue.

We repeat this operation until the entire queue is formed. The sequence of queuing will be as follows:



The start time of a part in workshop B is the maximum of the completion time of that part in workshop A and the completion time of the previous part in workshop B.

Product code	1	5	4	2	3
Completion time in workshop A	3	13	20	25	33
Completion time in workshop B	9	22	26	28	34

It's Computational Thinking

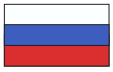
Computational Thinking Skills: Decomposition, Abstraction, Modelling & Simulation, Algorithms, Evaluation

Concepts: Abstraction, Data Representation, Data Interpretation, Specification, Algorithms, Implementation, Digital Systems, Interactions

It is a typical case of informatics task scheduling and greedy algorithm to design the optimal processing scheme for different parts of the workshop with different processing procedures and different time. The optimal solution of each stage will correspond to the global optimal solution.

A *greedy algorithm* is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage with the intent of finding a global optimum. In many problems, a greedy strategy does not usually produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time.

For example, a greedy strategy for the travelling salesman problem (which is of a high computational complexity) is the following heuristic: "At each step of the journey, visit the nearest unvisited city." This heuristic does not intend to find a best solution, but it terminates in a reasonable number of steps; finding an optimal solution to such a complex problem typically requires unreasonably many steps. In mathematical optimisation, greedy algorithms optimally solve combinatorial problems having the properties of matroids, and give constant-factor approximations to optimisation problems with submodular structure.



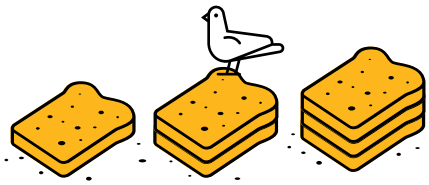
This question comes from Russia

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Hard

Squirrel Prison

Two brothers live at a monastery where they are trying to practice silence, but still wish to communicate. They worked out a way to "talk" to each other using the 6 slices of bread they each receive at lunch time. Each brother arranges his bread on the table in anywhere from 1 to 6 piles. They always use all 6 slices of bread. The arrangement of piles represents the word they wants to communicate.

For example, the word "hello" might be communicated by arranging the bread as shown:



Question

How many different words can the brothers communicate using this system?

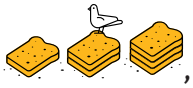
Answer

The answer is 32.

Each brother has 6 slices of bread. Consider the following method of arranging them:

- Place the first slice on the table.
- Flip a coin.
- If "heads", place the next slice on top.
- If "tails", start a new pile.
- Continue to flip coins until all the slices are arranged.

Every possible arrangement of bread can be achieved by following this method.

The example message,  , would be the result of flipping **tails, heads, tails, heads, heads**.

Since five coin flips are needed to arrange all 6 slices of bread, and each coin flip has two possible outcomes, there are $2^5 = 32$ different ways to arrange the bread.

The brothers can use their bread to communicate 32 different words.

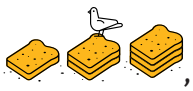
It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Abstraction, Algorithms, Evaluation

Concepts: Abstraction, Data Collection, Data Representation, Data Interpretation, Specification, Algorithms, Implementation, Digital Systems, Interactions, Impacts

In this task the brothers found a way to encode the words they wish to communicate. Encoding is the process of applying a code to data in order to represent the same data in a different (equivalent) way. The brothers used a code to represent their words as arrangements of bread.

Additionally, if we further encode flipping "heads" as 0 and flipping "tails" as 1, then we can represent arrangements of bread as binary strings.

The example message,  , would be the binary string **10100**.

Encoding has many uses in computer science. Encoding data can be a way to hide messages. This idea is called cryptography. Encoding data can also be a way to reduce the size of messages. This idea is called data compression.

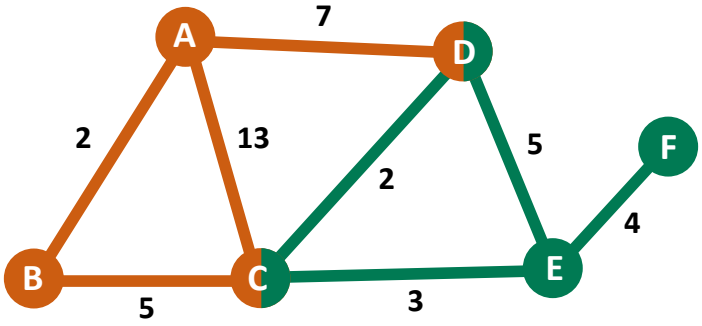
Telegraph Networks

An international data network is shown in the figure below. Each network has nodes (shown as circles) of the same colour. Nodes with two different colours indicate connections between different networks. There is a roaming charge in the connections, so if a message is transmitted from one network to another network, the cost of the transmission is:

(the cost of the path inside the first network) + (the cost of the path through all other networks x2)

Each branch of the network has its own cost, as shown in the graph.

Example:



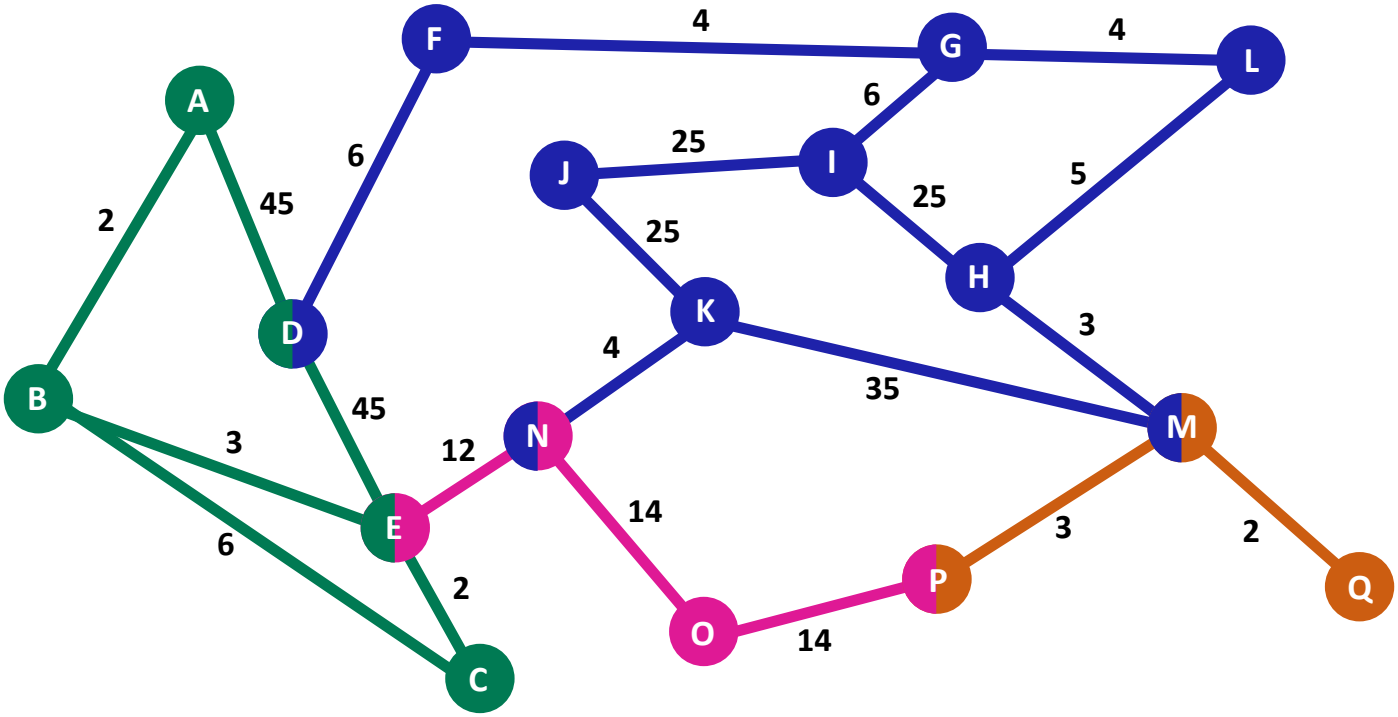
The cheapest path to transfer information from A to F is 21:

A > B	B > C	C > E	E > F
2	5	3	4

$$2+5+(3+4)*2=21$$

Question

Using these rules, what is the minimum cost to send a message from point A to point Q, as shown below? *Enter your answer as an integer.*

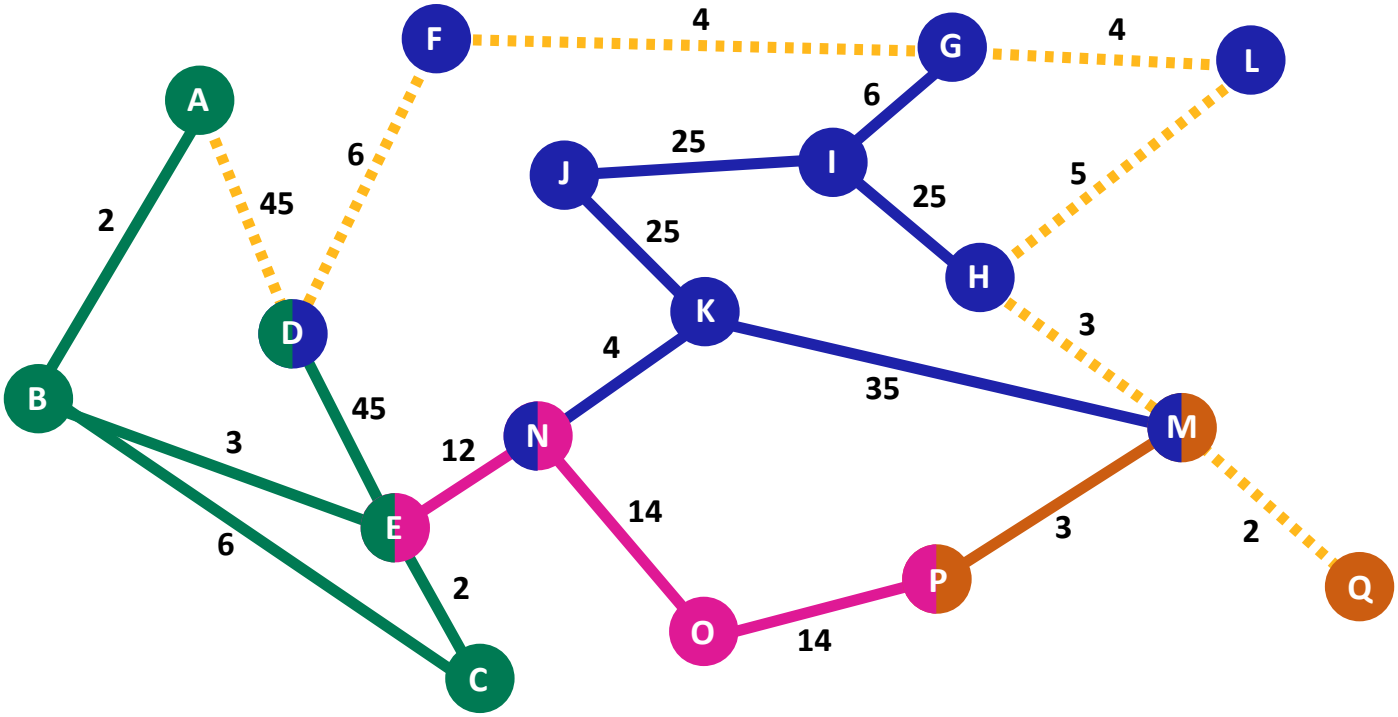


Continued on next page

Telegraph Networks – continued

Answer

The correct answer is: 93



The shortest route is not always the minimum cost path. In order to cross the green network, going to the violet network the minimum cost is 5, But from node E, crossing to N,K,M to Q, or N,O,P,M, to Q, the costs are higher than crossing from D,F,G,L,H,M to Q.

Then:
Path: A,D,F,G,L,H,M,Q=45 + (6 + 4 + 4 + 5 + 3 + 2) * 2 = 93
Path: A,B,E,N,K,M,Q=(2 + 3) + (12 + 4 + 35 + 2) * 2 = 111
Path: A,B,E,N,O,P,M,Q=(2 + 3) + (12 + 14 + 14 + 3 + 2) * 2 = 95

It's Computational Thinking

Computational Thinking Skills: Decomposition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Data Representation, Data Interpretation, Specification, Algorithms, Implementation, Digital Systems, Interactions, Impacts

The task consists in finding the minimum cost paths from one node to the other nodes. We are interested in the link points between subgraphs. To find the minimum cost paths, you can use, for example, Dijkstra's algorithm. Dijkstra's algorithm basically finds the shortest path in a graph, hence used in many fields including computer networking (Routing systems). It also has application in Google maps to find the shortest possible path from one location to other. In Biology it is used to find the network model in spreading of a infectious disease. Unfortunately, due to the way the costs are calculated in this task, Dijkstra's algorithm does not apply.

This question comes from
Belgium

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Hard

Flipping Cards

We play the following 'game'. A row of cards is laid out in front of you.

Some cards lie face up  , some lie face down  .

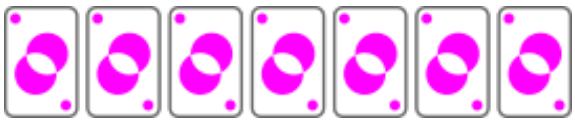
For each step in the game you do the following:
Starting at the right and moving left, turn over the cards one at a time, and stop as soon as you turn a card face up (or when you have turned over the leftmost card).

The picture below shows the effect of such a step (the sequence on the left shows the cards as they were before the step, the sequence on the right shows the cards after the step): you first turn over the rightmost card, then the card to the left of it and then the card to the left of that. At that point the step must stop, because the third card is turned face up.

Question

If you start the game with 7 cards lying face down:

How many steps will it take before you first encounter the sequence given below (with all 7 cards facing up)?



It will take 10 steps or less

It will take more than 10 steps but at most 100

It will take more than 100 steps but at most 1000

It will take more than 1000 steps

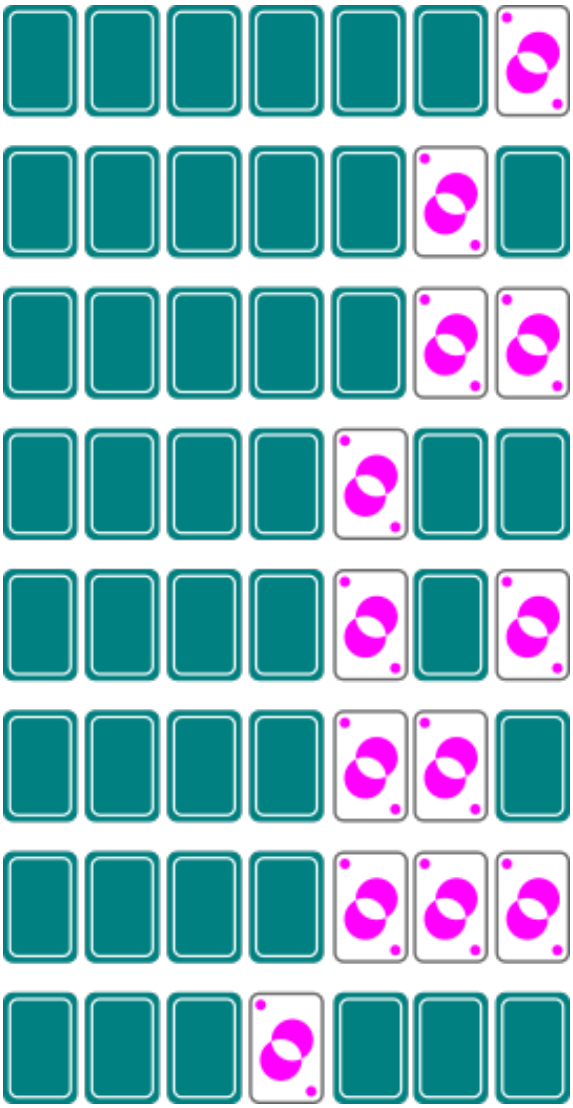
You can never reach the situation with all 7 cards facing up

Continued on next page

Flipping Cards – continued

Answer

There are various way to see (or guess) that this is the correct answer. If you perform the first few steps you will see the following patterns.



Notice that the rightmost card is first turned face-up at the 1st step, the second card from the right is first turned face-up at the 2nd step, the third card from the right is first turned over at the 4th step and the fourth card from the right is first turned face-up at the 8th step.

Looking at this sequence of numbers, we can see that every number in the sequence is double the number that preceeds it. So you can guess that it will take 16 steps before the 5th card from the right is turned over, 32 steps for the 6th card and 64 steps for the 7th card. (And the 7th card from the right is the first from the left.) So we need *at least* 64 steps.

It is a little bit more difficult to see that we need **127 steps** to turn *all* cards facing up. For that, we need to realise that just one step before the 4th card is turned over (on step 7) the pattern contains 3 consecutive cards facing up. Similarly, one step before the 5th card is first turned over (on step 15) the pattern contains 4 consecutive cards facing up. So, 7 cards facing up happens one step before the 8th card would be turned over (if there were 8 cards), which is at step 2x64-1 = **127**.

It's Computational Thinking

Computational Thinking Skills: Decomposition, Pattern Recognition, Modelling & Simulation, Algorithms, Evaluation

Concepts: Data Representation, Data Interpretation, Specification, Algorithms, Implementation, Digital Systems

Inside a computer numbers are represented in so-called *binary notation* where instead of the digits 0-9, a number is 'written' using only zeroes and ones (called *bits*). The representations of the first few numbers are as follows: 1 is represented as 0000001, 2 as 0000010, 3 as 0000011, 4 as 0000100, 5 as 0000101, 6 as 0000110, etc. (We use 7 bits for a number here, but on modern computers the total number of bits used is usually 32 or 64.)

Do you recognise these patterns? Indeed, if you use 0 for a card that is placed face down, and 1 for a card that is lying face up, than you obtain the same patterns as for the first 6 steps of our game. And if you know that 1111111 represents the number 127, then you see that indeed 127 steps are needed to end up with the requested pattern.

The 'step' which we use in our game is what is used by the actual electronics inside a computer to increase a binary number by 1.

We would like to thank the International Bebras Committee and community for their ongoing assistance, resources and collaborative efforts. Special thanks to Eljakim Schrijvers, Alieke Stijf and Dave Oostendorp for their support and technical expertise.

If you would like to contribute a question to the International Bebras community, please contact us via the details below.

Contact us

CSIRO Digital Careers
digitalcareers@csiro.au
digitalcareers.csiro.au