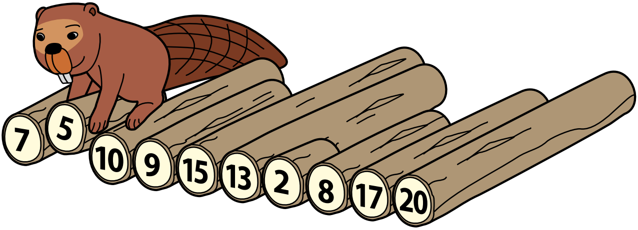
1 SORTING LOGS

A beaver is sorting the log set by comparing two adjacent logs. He compares the first log with the second one, and if they are out of order, he exchanges these two logs. Then he compares the second log with the third one, etc. If an exchange is made, then he looks to find the correct position for the swapped log within the sequence of the first sorted logs and places it there.
This can be repeated until all logs are sorted (let's say, from the shortest one to the longest)..

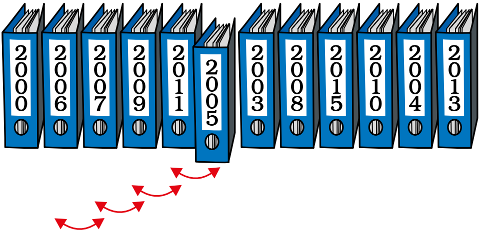


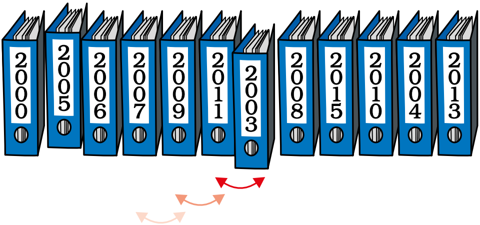
How many exchanges does the beaver need to do?

Insertion sort is used when the number of objects is small. It can also be useful when input data (array) is almost sorted, i.e., when only few elements are misplaced in a complete, big array.

2 SORTING FILES

An office has many files labeled by year. Office Beaver needs to sort them in ascending order so that each file can be accessed immediately if required.





Office Beaver compares two adjacent files one by one: if they are out of order, he exchanges these two files. If an exchange is made, then he looks to find the correct position for the swapped file within the sequence of the first sorted files and places it there.

This can be repeated until all files are sorted.

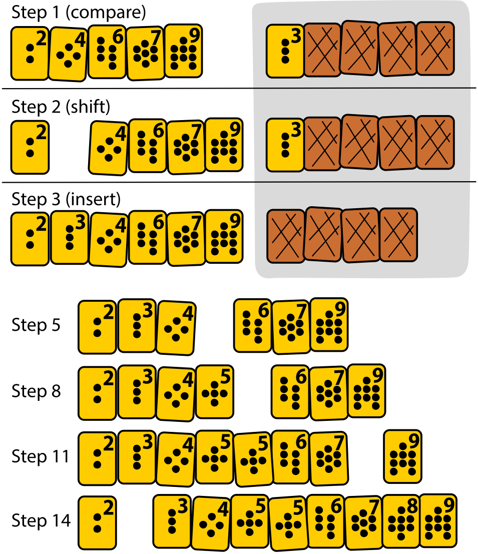
If the exchange of two files takes 5 seconds, how long will the insertion sorting of the given example take?

The algorithm is called insertion sort because you're inserting one item at a time into the correct position in a new array.

DRAFT

3 SORTING AND SHIFTING CARDS

Beaver is holding playing cards, and these cards are sorted. He takes a new card and puts it into the correct place so that the cards remain sorted.
The new card could be smaller than some of the sorted cards, and Beaver needs to go down the line, comparing the new card against each card in the line, until the place is found to put it in. Beaver inserts the new card in the right place, and once again, he has fully sorted cards. Then another card is taken, and the same procedure is repeated.
This is the idea behind this sorting algorithm.



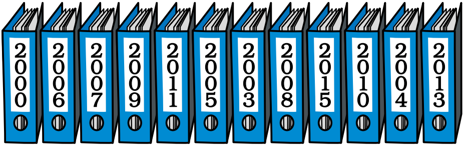
Which cards are inserted (in Steps 6, 9, 12, and 15)?

When people manually sort cards in a bridge hand, most use a method that is similar to insertion sort with shifting.

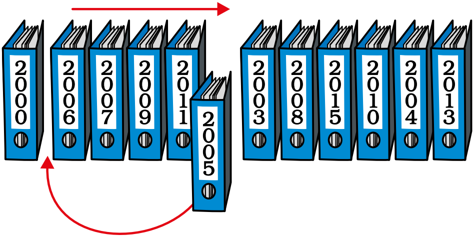
DRAFT

4 SORTING AND SHIFTING FILES

An office has many files labeled by year. Office Beaver needs to sort them so that each file can be accessed immediately if required.



Office Beaver would like to sort the files using an algorithm called the insertion sort with shifting.
Instead of exchanging two subsequent files, he shifts all files to the right until the space for the file to be inserted is free.



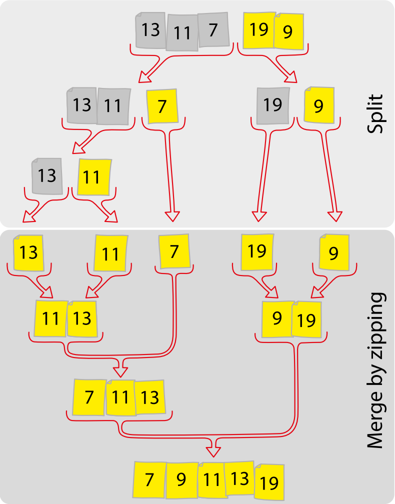
If it takes 5 second to take out a file and to shift the other files, how long will the sorting with this algorithm take?

Swapping means to exchange the places. Shifting means simply shift from the current location.

DRAFT

5 SORTING CARDS

A family is sorting the Bebras cards using the Mergesort algorithm.
First, divide the card set into two parts of (almost) equal size, repeat until you get the smallest unit (1 card). Then compare each card with the adjacent card set to sort and merge the two adjacent card sets by a kind of zipper principle. The algorithm is explained by an example:



For the given card set

5 17 11 1 3 7 8 20

which answer illustrates the first two card sets to be merged?

A. 5 17 and 11 1

B. 1 5 11 17 and 3 7 8 20

C. 5 and 17

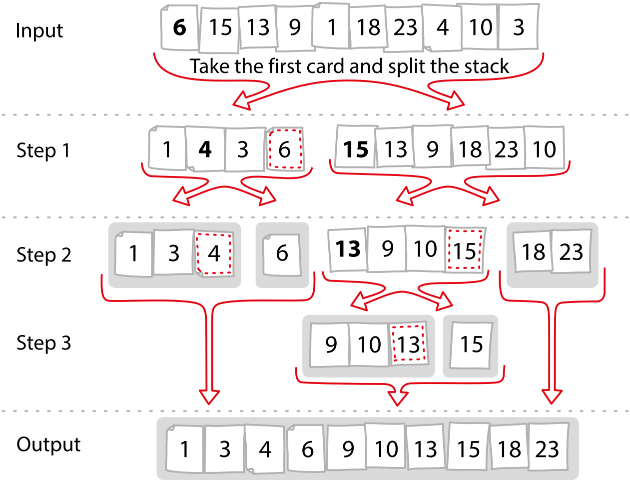
D 3 and 7

Mergesort is a divide and conquer algorithm that was invented by John von Neumann in 1945.

DRAFT

6 AS FAST AS POSSIBLE

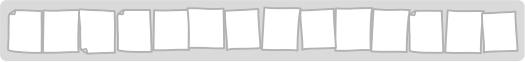
Quicksort is an efficient sorting algorithm, serving as a systematic method for placing the elements in order. With the following example we demonstrate how this algorithm proceeds:



The following card set is given:

Input 11 7 6 15 17 10 23 14 3 5 9 13 20 2

How does the card set look like at Step 3?



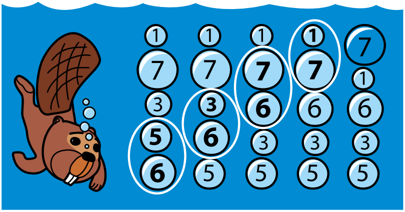
Quicksort algorithm was developed by the famous British computer scientist C.A.R. Hoare in 1961 and it is still a commonly used algorithm for sorting.

DRAFT

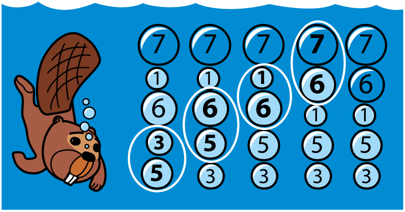
7 THE LARGEST BUBBLE RISES UP

Beaver plays with water bubbles: he sorts them from smallest to biggest using the Bubble sort algorithm. He repeatedly steps through the bubble-line to be sorted, compares each pair of adjacent bubbles and swaps them if they are in wrong order. The pass through the bubble line is repeated many times.

Pass 1



Pass 2



Beaver needs 5 passes to sort the bubbles.

How does the fourth pass look like?

The Bubble sort algorithm is simple, but it is slow and impractical for most problems. Generally, it needs to complete all passes even when no swaps happen. Due to its simplicity, bubble sort is often used to introduce the concept of a sorting algorithm.

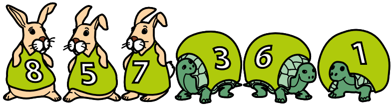
8 TORTOISES AND RABBITS

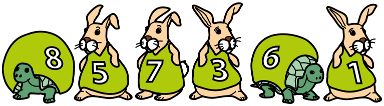
Bubble sort is an algorithm based on the idea of repeatedly comparing pairs of adjacent elements and then swapping their positions if they exist in the wrong order. In Bubble Sort, elements move in different directions at different speeds:

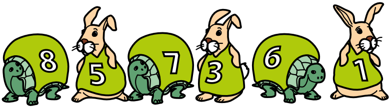
- An element that must move toward the end can move quickly because it can take part in successive swaps.
- An element that must move toward the beginning of the list cannot move faster than one step per pass, so elements move toward the beginning very slowly.

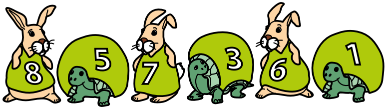
If the smallest element is at the end, it will take n−1 passes to move it to the beginning. This has led to these types of elements being named rabbits and turtles, respectively (after the characters in Aesop's fable The Tortoise and the Hare).

Which picture represents most precisely the Bubble sort?

A.


B.


C.


D.


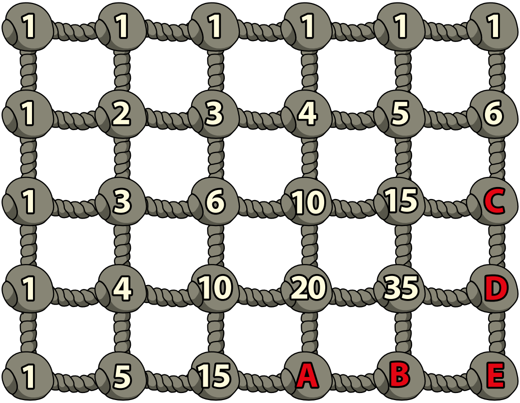
It is called Bubble sort, because with each iteration the largest element in the list bubbles up towards the last place, just like a water bubble rises up to the water surface.

DRAFT

9 COUNTING SHORTEST PATHS

Dynamic programming is a method for solving a complex problem by breaking it down into a collection of simpler subproblems. It suggests solving each of the subproblems only once and recording the results in a table from which a solution to the original problem can be obtained.

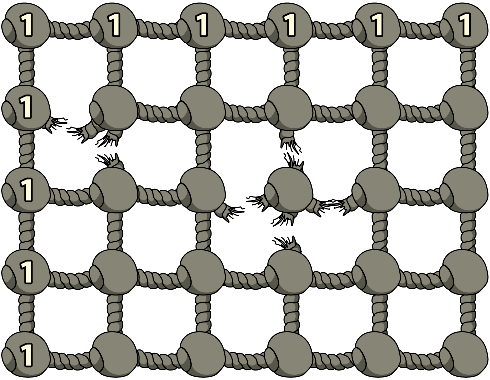
As a clear example of the dynamic programming, consider a problem of counting shortest paths.
Count the number of the shortest paths from Start to each intersection in a grid of squares. Compute the values row by row down and moving left to right along each row. Find valutes of A, B, C, D, and E.



Dynamic programming is both a mathematical optimization method and a computer programming method (invented by U.S. mathematician Richard Bellman, in the 1950s). In both contexts it refers to simplifying a complicated problem by breaking it down into simpler sub-problems in a recursive manner.

10 BREAKED PATHS

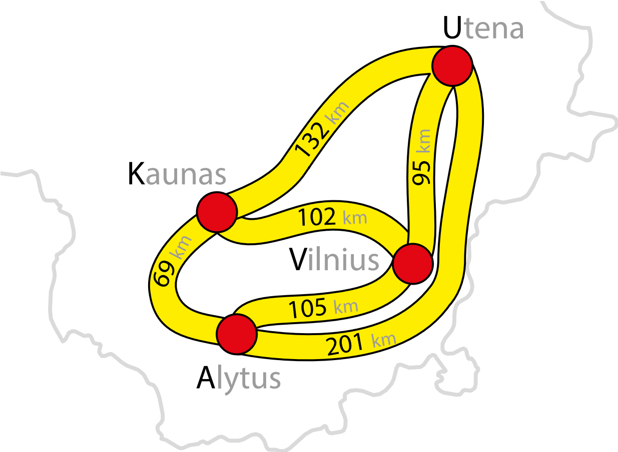
Bever would like to count the number of the shortest paths from Start to each intersection in a grid of squares. The values should be computed row by row down and moving left to right along each row. However some paths are broken.
Bever would use the dynamic programming method which suggests solving each subproblem only once and recording the results in a table from which a solution to the original problem can be obtained.
Help the beaver to fill the empty rounds.



Dynamic programming is used for example, in the coin change problem of finding the minimum number of coins of given denominations needed to make a given amount. Dynamic programming would find an optimal solution for each amount by first finding an optimal solution for each smaller amount and then using these solutions to construct an optimal solution for the larger amount.

11 MISSING ROUTES

Lina must travel from Vilnius (**V**) visiting 3 towns and coming back to **V**.



Applying the brute force algorithm we started to write down the routes that start and end at **V**.

1. V K U A V	102	+	132	+	201	+	105	=	538
2. V A U K V	105	+	201	+	132	+	102	=	538
3. V U A K V	95	+	201	+	69	+	102	=	467
4. V K A U V	102	+	69	+	201	+	95	=	467

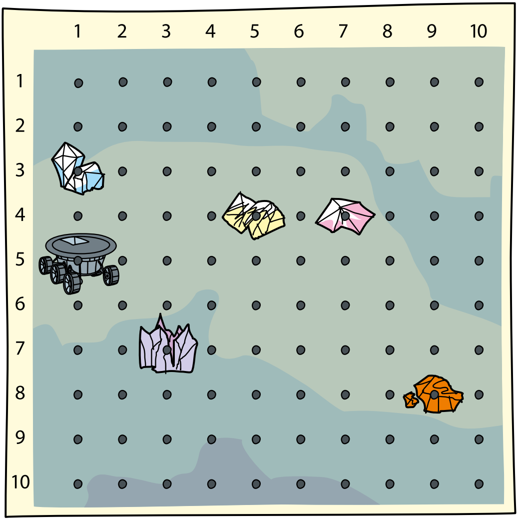
How many routes are missing?
How long is the shortest route?

The travelling salesman problem (TSP) is one of the most famous and most studied problems in theoretical computer science. The TSP has several applications, such as in planning, logistics, the manufacture of microchips, and even DNA sequencing.

DRAFT

12 LUNATIC ROVER ROAD

A lunatic rover road was sent to the moon to collect minerals. The marked errands in the given map must be run (in no particular order).



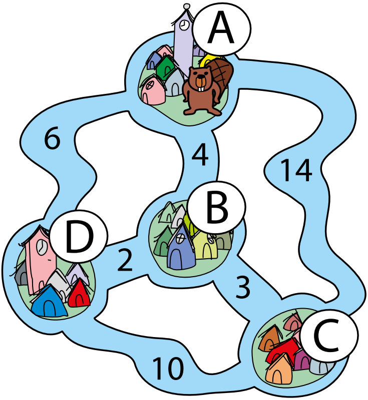
The lunatic rover must start and end at home position where it is now. Each block on the map is exactly 1 km.

Draw a graph correcponding to this problem.
Guide the lunatic roler to find the optimal (shortest) route to collect all minerals..

Actually the Traveling Salesman Problem (TSP) is the problem of finding an optimal Hamilton circuit in a complete weighted graph. The name is chosen by old tradition regardless of the context in which it comes up.

13 NEIGHBOURS

Beaver swims from city A visiting each of 3 cities (B, C, D) only once and coming back to A. Paths are estimated by costs and indicated by numbers.

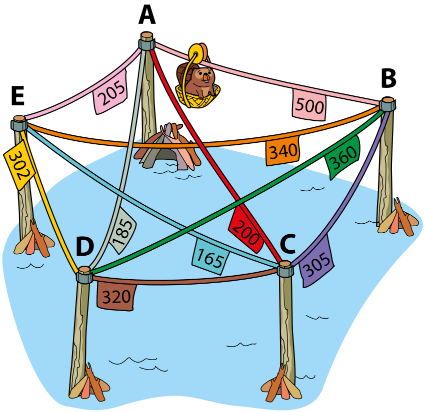


Beaver wants to apply the Nearest Neighbour Algorithm, which is almost completely described by its title:
Start at home.
Whenever you are in a city, travel to the cheapest neighboring city that you haven't visited yet.
Keep doing step 2 until you have visited all the cities.
Here are 3 possible routes:
1. A B D C A = 4+2+10+14
2. A B C D A = 4+3+10+6
3. A C B D A = 14+3+2+6
Which one is obtained by the Nearest Neighbour Algorithm?

This task is the TSP (Travelling Salesman Problem) and it cannot be solved by an efficient algorithm. The Nearest Neighbour Algorithm tends to give a solution reasonably close to the optimal solution.

14 CHANNEL NETWORK

Let's apply the Nearest Neighbour Algorithm to a bever tour visiting his friends.



He starts at A (that's home). Each path is estimated by euros. From A the cheapest trip is to D (185 €), so he goes there. From D the cheapest place to go to next (other than A) is E. From E the cheapest place to go to is C. From C the cheapest new place to go to is B (Beaver doesn't want to go back to E or A!). Once he is at B, he visited all the friends. The last leg of the trip is back to A, Thus, the nearest neighbour algoritm produces the Hamilton circuit A, D, E, C, B, A with a total cost of 1457€.

Is this optimal solution?

If not, apply an improved algorithm so called the repetitive nearest neighbour algoritm. The idea is that the nearest neighbour algoritm can be started at any vertex of the graph (not just the home vertex). The Hamilton circuit can then be rewritten with the home vertex as the starting point.

Calculate the rest four Hamilton circuits by applying the repetitive nearest neighbour algoritm.

Both the Nearest Neighbour Algorithm and its modification produce solutions that are not optimal in general, however we get an algorithm that is easily understood and quickly carried out.

15 EASY WAY TO FIND PRIMES

The first algorithm for the computation of primes was introduced by Eratosthenes using a very simple idea:

1. Create a list of consecutive integers from 2 through n (n = 100).
2. Initially take 2, the smallest prime number, and strike through all the multipliers of 2, e.g. 4, 6, 8 . . .
3. Take the next number that is not striked, e.g. 3 and strike all its multipliers. If there is no such a next number, stop.
4. Repeat Step 3 until the algorithm terminates.

The numbers remaining not striked through in the list are all the primes below n.

How many primes are there between 2 and 100?

	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~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	49	~~50~~
~~51~~	~~52~~	53	~~54~~	~~55~~	~~56~~	~~57~~	~~58~~	59	~~60~~
61	~~62~~	~~63~~	~~64~~	~~65~~	~~66~~	67	~~68~~	~~69~~	~~70~~
71	~~72~~	73	~~74~~	~~75~~	~~76~~	77	~~78~~	79	~~80~~
~~81~~	~~82~~	83	~~84~~	~~85~~	~~86~~	~~87~~	~~88~~	89	~~90~~
91	~~92~~	~~93~~	~~94~~	~~95~~	~~96~~	97	~~98~~	~~99~~	~~100~~

Eratosthenes was the third librarian of the famous library in Alexandria and developer of an algorithm for collecting prime numbers - it is known as the Sieve of Eratosthenes. This method is over 2200 years old. But it's very simple and elegant and you can teach it to any child.

16 FASTER COMPUTATION OF PRIMES

The improved algorithm for the computation of primes works like this:

1. Make a current list of numbers from 2 to as large as you wish; call the maximum number n.
2. Append the first number from the current list into a list of primes, make a new list in which each element of the current list, including the first, is multiplied by the appended first number.
3. "Subtract" the new list from the current list, keeping in the resulting list only those numbers in the current list that do not appear in the new list. Make this resulting list into the current list. Exclude from this current list the first element.
4. Repeat Steps 2, 3, and 4 on the current list until the current list is empty.

How many times is the "subtraction" of the list needed for getting the first 10 primes?

The ancient Sieve of Eratosthenes is inefficient in the sense that some composite numbers are struck out more than once; for instance, 21 is struck out by both 3 and 7. The great Swiss mathematician Leonhard Euler invented a sieve that strikes out each composite number exactly once, at the cost of some additional bookkeeping.

17 CATCH A LION

Informatics knows how to catch a lion in Africa!
Here is a legendary algorithm based on the binary search technique.

Step 1. Divide the area in half.

Step 2. Check the first half - if the lion is there, explore this one further. If not, shift focus to the other half.

Step 3. Check if the explored area is less than 0.5 km2, then assume that the lion is caught, and the algorithm stops. Otherwise go to Step 1.

The area of Africa is about 30 370 000 km2.

How many areas do you need to check in order to catch the lion?

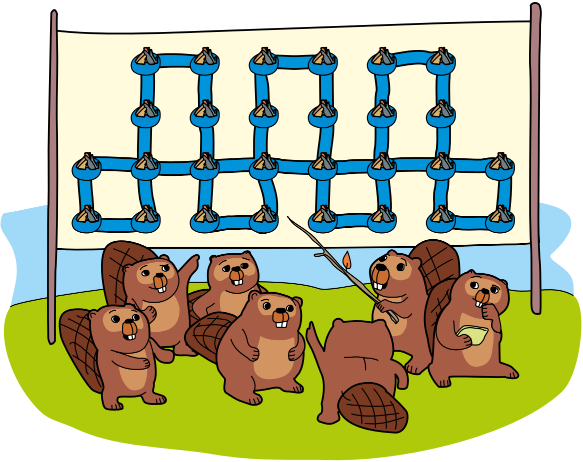
18 COUNTERFEIT COIN

In computer science, binary search, also known as half-interval search, logarithmic search, or binary chop, is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array; if they are unequal, the half in which the target cannot lie is eliminated and the search continues on the remaining half until it is successful. If the search ends with the remaining half being empty, the target is not in the array.

A well-known example has nine coins, that are identical in weight save for one, which in this example is lighter than the others—a counterfeit. The difference is only perceptible by weighing them on balance scale—but only the coins themselves can be weighed.
Is it possible to isolate the counterfeit coin with only two weighings?

19 IS IT POSSIBLE?

There is a grid of canals connecting villages. Such abstraction is called a graph and serves for many practical tasks. A graph consists of two types of objects: vertices and edges. The vertices usually are represented as dots and the edges as lines.
There are are many paths in the graph. When a path contains each and every edge of a connected graph exactly once, it is called an Euler path.
Beavers want to swim each canal exactly once. One of the beavers knows an useful Euler's theorem: If a graph is connected and has exactly two vertices of odd degree, then it has at least one Eulerian path (usually more).
Some beavers don't know what the degree of a vertex means. Could you help by guessing?
Could you determine if the graph below (the canal system) has an Eulerian path?



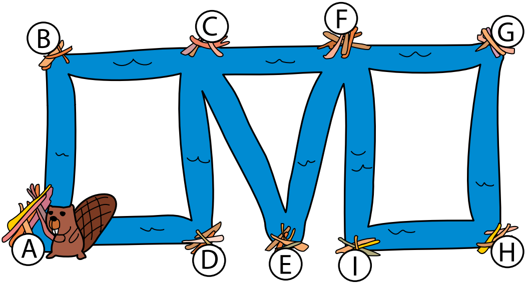
Eulerian path was first discussed by Swiss mathematician, physicist, astronomer, logician and engineer Leonhard Euler while solving the famous Seven Bridges of Königsberg problem in 1736.

20 FINDING AN EULERIAN CIRCUIT

Beaver wants to swim each and every canal exactly once starting and ending at his lodge A. (When an Eulerian path starts and ends at the same vertex it is called an Eulerian circuit.)

Beaver uses the Fleur's algorithm whose main idea is:

At each step, if he can, he chooses the next canal to not be a "a bridge" for the untraveled part (a bridge is defined as a vertex whose removal divides the graph into two disconnected parts - burn a bridge behind you, and you'll never be able to get back to where you were). If there is no such "non-bridge" canal, pick the remaining canal.



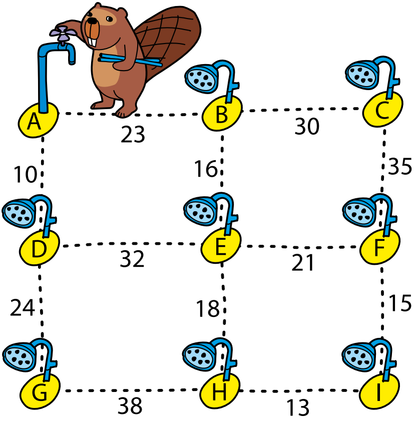
Step 1: A to B
Step 2: B to C
Step 3: C to E (C to D is a bridge!)
Step 4: E to F
Step 5: Which one of the canals F to C, F to G or F to I should be chosen?

How many steps are left to complete the Eulerian circuit?

To find an Eulerian circuit or an Eulerian path, bridges are the last edges you want to cross.

21 WATER PIPE SYSTEM

Beaver wants to lay down a network of pipes so that water from the main source A can get to each of the sprinkler-heads (A, B, C, D, E, F, G, H, I). He also wants to do the job as cheaply as possible. A number on a pipe (edge) indicates the cost for the installation of this pipe.



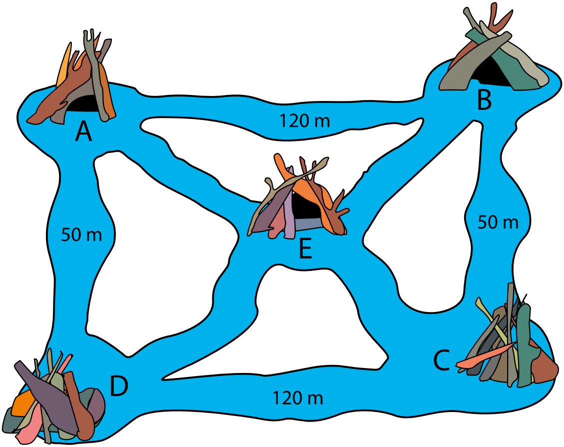
Beaver knows the Kruskal's algorithm:
Find the edge with the smallest cost (if there is more than one, pick one at random). Mark it.
Find the next smallest unmarked edge. If it forms a circuit with already marked edges, discard it, and remove it from further selection. If it doesn't, then mark it.
Repeat step 2 until all sprinkler-heads are connected.
He applies the algorithm:
Mark AD.
Mark HI.
Mark FI.
Mark BE.
Mark EH.
Mark EF. Circuit. Remove EF.
...

Continue and find the cheapest network of pipes. How does this network (a tree) look like?

Kruskal's algorithm always finds a minimum spanning tree (with the least total weight) for a connected weighted graph.

22 BUILDING A CHANNEL

Five beaver lodges (A, B, C, D, E) are loccated as shown on the folloving map.

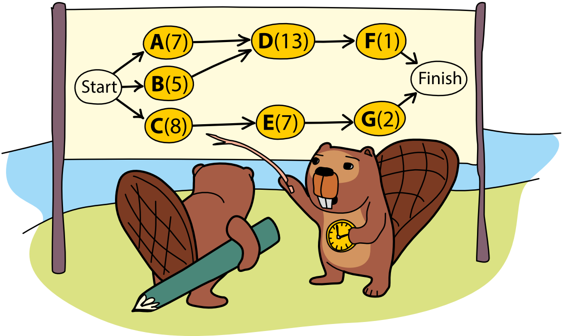


Beaver family would like to connected these five lodges by a water channel. The work of building the channel system connecting any two lodges is proportional to the distance between the lodges. Help bever family to find the lengh of the channel network of easiest work.

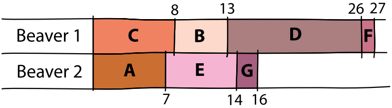
DRAFT

23 TWO BEAVERS ARE WORKING

Two beavers build a dam and need to do 7 tasks: A(7), B(5), C(8), D(13), E(6), F(1), G(2). The numbers in the brackets indicate working hours. Some tasks must be completed before others can be started - the procedence relations are represented by the arrows.



Beavers use the Decreasing Time Algorithm (DTA): from among all the tasks that are available at that moment they should choose the longest one (see the time Table).



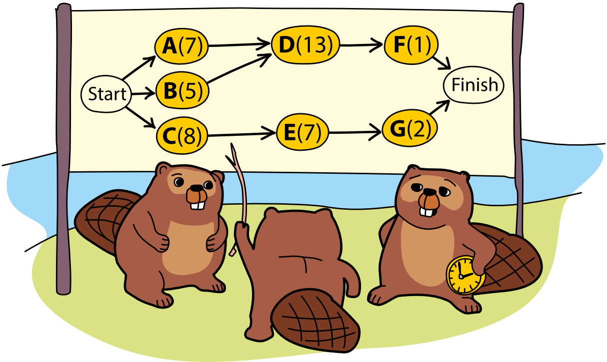
Following this algorithm the beavers can complete a dam in 27 hours.

Is it possible to build the dam in a shorter time? Validate the answer.

The Decreasing-Time Algorithm is based on a seemingly simple strategy:
Do the longer jobs first and save the shorter jobs for last.

24 THREE BEAVERS ARE WORKING

Now thre beavers would like build a dam and need to do 7 tasks: A(7), B(5), C(8), D(13), E(6), F(1), G(2). The numbers in the brackets indicate working hours. Some tasks must be completed before others can be started - the procedence relations are represented by the arrows.



Beavers use the Decreasing Time Algorithm (DTA): from among all the tasks that are available at that moment they should choose the longest.

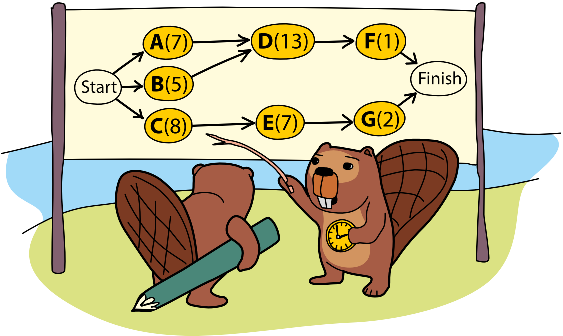
Make a time table. How much time they will need to complete a dam?

Although the strategy of scheduling the longer tasks first sounds good, it does have a major flaw. The DTA ignores any information in the project digraph that might indicate that one or more tasks should be done early rather than late.

DRAFT

25 TASK SCHEDULING

Two beavers build a dam and need to do to do 7 tasks: A(7), B(5), C(8), D(13), E(6), F(1), G(2). The numbers in the brackets indicate working hours. Some tasks must be completed before others can be started - the precedence relations are represented by the arrows.



We added two fictitious tasks, START and END with processing time 0. This is just a convenience that allows us to visualise and get a directed graph called a shorted digraph.

Bevers know this about a Critical Path and want to find it: The Critical Path can be thought of as the longest path (in terms of total completion time) in the entire digraph.

The Backflow Algorithm helps to find the length of the critical paths from each vertex in a digraph. The basic idea is to start at END and move backward toward START. At each vertex we record (in square brackets) the length of the critical path from that vertex.

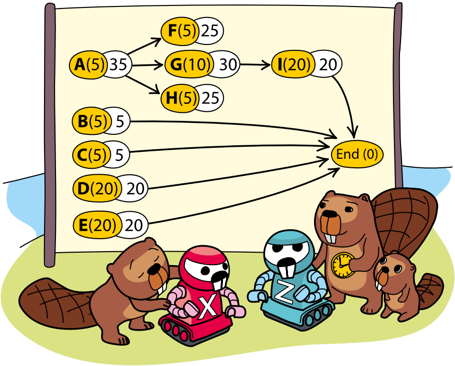
Complete and record critical paths of the vertex A, D, …

Although the strategy of scheduling the longer tasks first sounds good, it does have a major flaw. The DTA ignores any information in the project digraph that might indicate that one or more tasks should be done early rather than late.

DRAFT

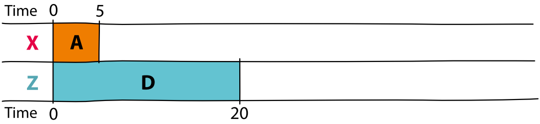
26 SCHEDULING JOBS TO TWO ROBOTS

Beaver family uses two robots X and Z and they need to do 9 tasks: A(5), B(5), C(5), D(30), E(30), F(10), G(10), H(10), I(30). The numbers in the brackets indicate working hours. Some tasks must be completed before others can be started - the precedence relations are represented by the arrows.



Beavers know the critical path algorithm and would like to use it. In the algorithm, whenever a robot is free, he should look at all the tasks that are available at that moment and choose the one that is on a path which has the longest total sum of completion time from start to end - called the critical path. In case of a tie, choose task randomly. For each task, the length of critical path is recorded in square brackets.

Continue using the Critical Path Algorithm to schedule the tasks for the two robots.



27 WHAT IS WRITTEN?

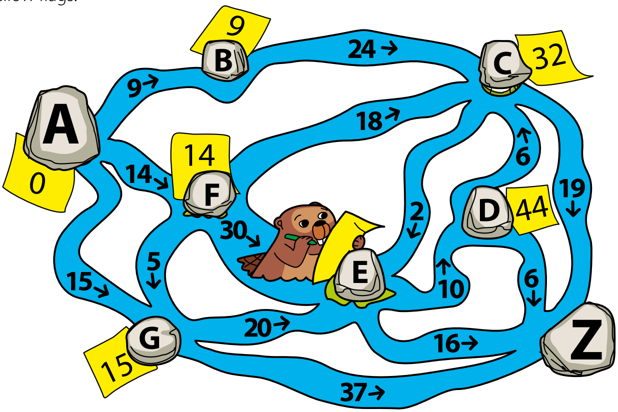
In cryptography, a Caesar cipher is one of the simplest encryption techniques. It is a type of substitution cipher in which each letter in the plaintext message is replaced by a letter some fixed number (key) of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. Beaver doesn't know the key. **Could you help him to read the inscription?**

Hfjxfw hnumw nx fqxt pstbs fx ymj xmnky hnumw, Hfjxfw htij tw Hfjxfw xmnky.

The Caesar cipher is named after Julius Caesar, who used it in his private correspondence. Caesar cipher is also known as the shift cipher, Caesar code or Caesar shift.

28 EXPLORING PATHS

Beaver explores one-way paths through the Bebras lodges labeled from A to Z. All the rivers have a length and a direction. The length and direction are given in the yellow flags.



Over the course of many different trips, Beaver leaves blue notes with a number under a log in each lodge.

What is the meaning of the numbers he has left under the logs?

A. the length of the shortest path from A to a particular lodge going through the least number of lodges;

B. the length of the shortest path from A to a particular lodge;

C. the length of the shortest path from A to a particular lodge by taking a left turn at crossings if possible;

D. the length of the shortest path from A to a particular lodge by taking a right turn at crossings if possible.

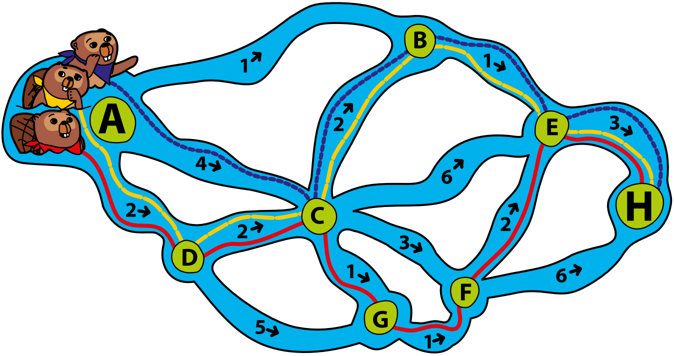
29 FINDING THE SHORTEST PATH

Four beavers have travelled from lodge A to lodge Z by swimming only downstream. The beavers' paths are marked by particular colours.

One of the beavers knows the Dijkstra's algorithm for the shortest path:

1. Assign a tentative distance value to every crossing by setting it to zero for A (our initial crossing) and to infinity for all other crossings.
2. Set A as the current crossing.
3. For the current crossing, consider all of its unvisited neighbours and calculate their tentative distances. For each neighbour, compare the newly calculated tentative distance with the current assigned value and assign to it the smaller one.
4. When we are done considering all of the neighbours of the current crossing, mark the current crossing as visited. A visited crossing will never be checked again.
5. Select the unvisited crossing that is marked with the smallest tentative distance, set it as the new "current crossing", and go back to step 3.
6. If the destination crossing (Z) has been marked visited, then stop. The algorithm has finished.

Which one of the beavers has used the Dijkstra's algorithm?

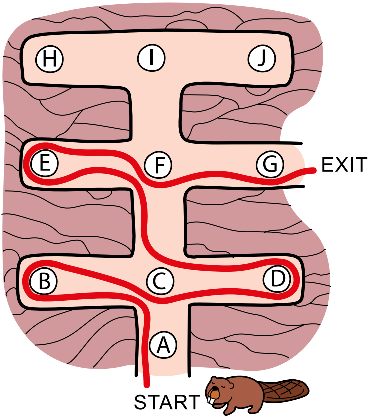


In some fields, like artificial intelligence in particular, Dijkstra's algorithm or a variant of it is known as uninformed-search and formulated as an instance of the more general idea of best-first search

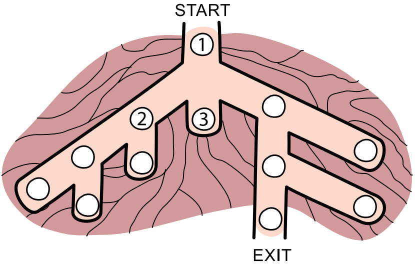
30 LABYRINTH

Beaver entered an underground labyrinth and seeks to exit it. He knows the so-called breadth-first search algorithm: it searches the labyrinth layer-wise from the start point, that is, first all directed neighbours (distance 1) , then all corridors at distance 2, etc.

The breadh-first search is described by an example. Starting from A, a path to G is sought and the way is shown.



Help Beaver to visit the second labyrinth using the breadth-first search. Mark with numbers in circles the to show in which order the junctions or ends are visited.



SOLUTIONS

1A. Sorting logs

The following table shows the steps for sorting the logs {7, 5, 10, 9, 15, 13, 2, 8, 17, 20}.
In each step, the key log under consideration is underlined. The key log that was moved (or left in place because it was longest yet considered) in the previous step is shown in bold.

7, 5, 10, 9, 15, 13, 2, 8, 17, 20
5, 7, 10, 9, 15, 13, 2, 8, 17, 20 - 1 exchange
5, 7, 10, 9, 15, 13, 2, 8, 17, 20
5, 7, 9, 10, 15, 13, 2, 8, 17, 20 - 1 exchange
5, 7, 9, 10, 15, 13, 2, 8, 17, 20
5, 7, 9, 10, 13, 15, 2, 8, 17, 20 - 1 exchange
2, 5, 7, 9, 10, 13, 15, 8, 17, 20 - 6 exchanges
2, 5, 7, 8, 9, 10, 13, 15, 17, 20 - 4 exchanges
2, 5, 7, 8, 9, 10, 13, 15, 17, 20
2, 5, 7, 8, 9, 10, 13, 15, 17, 20

In total 13 exchanges.
In the worst case where all files are in reverse order, the number of exchanges is
$1 + 2 + 3 + \ldots + (n-1) = n \times (n-1) / 2$

1B. Sorting files

The following table shows the steps for sorting the files {2000, 2006, 2007, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013}. In each step, the key file under consideration is underlined. The key that was moved in the previous step is shown in bold.

2000, 2006, 2007, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, 2006, 2007, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, 2006, 2007, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, 2006, 2007, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, 2006, 2007, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, 2005, 2006, 2007, 2009, 2011, 2003, 2008, 2015, 2010, 2004, 2013 - 4 exchanges
2000, 2003, 2005, 2006, 2007, 2009, 2011, 2008, 2015, 2010, 2004, 2013 - 5 exchanges
2000, 2003, 2005, 2006, 2007, 2008, 2009, 2011, 2015, 2010, 2004, 2013 - 2 exchanges
2000, 2003, 2005, 2006, 2007, 2008, 2009, 2011, 2015, 2010, 2004, 2013
2000, 2003, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2015, 2004, 2013 - 2 exchanges
2000, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2015, 2013 - 8 exchanges
2000, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2013, 2015 - 1 exchange
2000, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2013, 2015
In total 22 exchanges or $22 \times 5 = 110$ seconds.
In the worst case where all files are in reverse order, the number of exchanges is
$1 + 2 + 3 + \ldots + (n-1) = n \times (n-1) / 2$

2A. Sorting and shifting cards

5 5 8 2

2B. Sorting and shifting files

If it takes 5 second to take out a file and to shift the other files, how long will the sorting with this algorithm take?
In each step, the key the key file that is taken out is underlined.

2000, 2006, 2007, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, 2005, 2006, 2007, 2009, 2011, 2003, 2008, 2015, 2010, 2004, 2013 - 5 seconds
2000, 2003, 2005, 2006, 2007, 2009, 2011, 2008, 2015, 2010, 2004, 2013 - 5 seconds
2000, 2003, 2005, 2006, 2007, 2008, 2009, 2011, 2015, 2010, 2004, 2013 - 5 seconds
2000, 2003, 2005, 2006, 2007, 2008, 2009, 2011, 2015, 2010, 2004, 2013
2000, 2003, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2015, 2004, 2013 - 5 seconds
2000, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2015, 2013 - 5 seconds
2000, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2015, 2013
2000, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2013, 2015 - 5 seconds
The sorting with this algoritm takes 30 seconds, e.i. 6 x 5 = 30 seconds.

3A.

Incorrect. The first two card sets merged will be base case sets, we have not yet reached a base case.
Incorrect. These will be the last two lists merged
Correct! The card 5 and 17 are the first two base cases encountered by mergesort and will therefore be the first two card sets merged.
Incorrect. Although cards 3 and 9 are next to each other they are in different halves of the card set starting with the first split.

3B.

4A. The largest bubble rises up

The fourth pass is represented in the picture below.
7 7 7 7 7
6 6 6 6 6
5 5 5 5 5
1 3 3 3 3
3 1 1 1 1

N3 is in order and the last bubble N1 should be in order as well. Now, the array is already sorted, but the algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.
To analyze the bubble sort, we should note that regardless of how the items are arranged in the initial list, n−1 passes will be made to sort a list of size n.

4B. Tortoises and rabbits

D is correct.
We should pay attention to
- An element that must move toward the end can move quickly - they should be rabits (8, 7, 9).
- An element that must move toward the beginning of the list cannot move faster than one step per pass, so elements move toward the beginning very slowly - they should be turtles (1, 3, 5).

5A. Counting shortest paths

Solution can be proved by reasoning. Let P[i, j] be the number of shortest paths from Start intersection to the intersection i (1<=i<=5) and j ((1<=j<=6). Any shortest path here is composed of horizontal segments going right along the ropes and vertical segments going down the ropes. Therefore, the number of shortest paths from Start to the intersection i and j can be found as the sum:

P[i, j] = P[i-1, j] + P[i, j-1] for every 1<i<=5, 1<j<=6

where P[1, j] = 1 for every 1<=j<=6, P[i, 1]=1 for every 1<=i<=5

Using these formulas we can compute the values of P[i, j] row by row starting with row 1 and moving left to right along each row.

The problem can be also solved by simple combinatorial argument.

A = 35, B = 70, C=21,D=56 ir E=126.

5B. Breaked paths

Counting paths is a well-known application of dynamic programming. This approach finds the number of shortest paths from Start to every intersection in the grid. The breaken segments are not counted. If an intersection has both the left and upper neighbours, its number is computed as the sum of the neighbouring numbers; if an intersection has just one of such neighbours, it gets the same number as that of the neighbour.

10A. Is it possible?

Yes, it is possible.

14B. Scheduling jobs to two robots

15A. What is written?

The key is 5.

16A. Exploring paths

In order to find the correct answer, the distances for each lodge according to the different specifications A through D have to be computed:

A. is wrong because otherwise D = 45, Z = 52;

C. is wrong because otherwise C = 33, D = 45, Z = 52;

D. is wrong because otherwise C = 51, D = 45, Z = 52.

So the blue number shows the length of the shortest route from A to a particular lodge (B).

The problem of finding the shortest route is called "shortest path problem" and it is one of the fundamental computing (informatics) tasks with everyday applications. Shortest path algorithms are applied to automatically find directions between locations, such as driving directions on websites like MapQuest or Google Maps. Dijkstra's algorithm is one of the most popular algorithms used to find the shortest route. It assigns some initial distance values and improves them step by step.