University of Ottawa Computer Science Club

Presents

CS Games 2020 Tryouts

Algorithms Tryout

Time: 3 Hours

Language: Your Choice

Difficulty:

Instructions

Provide solutions for each problem in separate files. Submit all solution files to the git repository provided. Partial solutions are accepted and partial points are awarded. Commented code is typically easier to understand when things aren’t working. Include the string “Q{Question #}” in the name of each solution file. For example, your solution file for question 1 must Have the string “Q1” in it.

As these questions only test your algorithmic capabilities, you can write Your solutions in any object oriented language. It is preferred that you use a common, main-stream language (Python, Go, Java, Ruby etc.) and stay away from functional languages for this test. Please include a README file indicating how to compile and run your solutions.

To submit, push your code to the git repository that you were provided. For multiple submissions, only the final is marked so don’t be afraid to push multiple times. Sample input and output for each question (except for question 1) can be found in the files “Q#-examples.txt” where # is the question number.

Challenges (100 Pts. Total)

1. Sorting (5 pts.)

Consider an array of numeric strings where each string is a positive number with anywhere from ***1*** to ***10^5*** digits. Sort the array's elements in ascending order of their integer values and print each element of the sorted array on a new line.

Input Format

- The first line contains a single integer, ***n***, denoting the number of strings in ***unsorted***.
- Each of the subsequent ***n*** lines contains an integer string ***unsorted[i]***.

Constraints

- ***$1 \le n \le 10^4$***
- Each string provided will be a positive, unsigned integer
- The total number of digits across all strings in ***unsorted*** is between ***1*** and ***10^5*** inclusive.

Output Format

Print each element of the sorted array on a separate line

Sample Input

```
8
6
15
2
23478750267234654378234
156
10
9
10
```

Sample Output

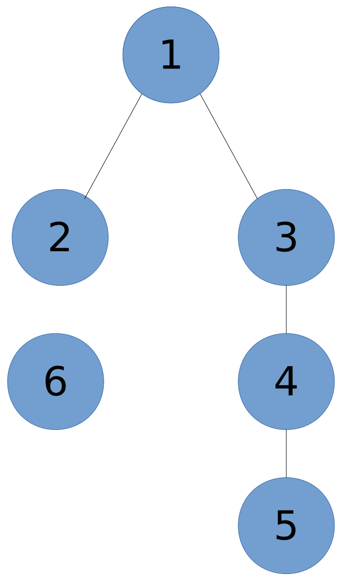
```
23478750267234654378234
156
15
10
10
9
6
2
```

2. Tree Searching (25 Pts.)

Consider an undirected graph where each edge is the same weight. Each of the nodes is labeled consecutively.

You will be given a number of queries. For each query, you will be given a list of edges describing an undirected graph. After you create a representation of the graph, you must determine and report the shortest distance to each of the other nodes from a given starting position using the breadth-first search algorithm (BFS). Distances are to be reported in node number order, ascending. If a node is unreachable, print -1 for that node. Each of the edges weighs 4 units of distance.

For example, given a graph with 6 nodes and 4 edges: ***[1,2], [1,3], [3,4] [4,5]***



The start node is the root (***1***). Outputs are the calculated distances to nodes ***2*** through ***6***: ***4***, ***4***, ***8***, ***12***, ***-1***. Each edge is 4 units, and the unreachable Node ***6*** gets the required return value of -1. If a node is unreachable, its distance is always -1.

Input Format
The first line contains an integer ***q***, the number of queries. Each of the following ***q*** sets of lines has the following format:

- The first line contains two space-separated integers, ***n*** and ***m***, the number of nodes and edges in the graph.
- Each line ***i*** of the ***m*** subsequent lines contains two space-separated integers, ***u*** and ***v***, describing an edge connecting node ***u*** to node ***v***
- The last line contains a single integer, ***s***, denoting the index of the starting node.

Constraints

- ***1≤q≤10***
- ***2≤n≤1000***
- ***1≤u,v,s≤n***

Output Format

For each of ***q*** queries, print a single line of ***n-1*** space-seperated integers Denoting the shortest distances to each of the n-1 other nodes from starting position ***s***. These distances should be listed sequentially by node number, But should not include ***s***. If some node is unreachable, print -1 as the Distance.

Examples

Found in: Q2-examples.txt

3. Cipher (30 Pts.)

An English text needs to be encrypted using the following encryption scheme. First, the spaces are removed from the text. Let ***L*** be the length of this text. Then, characters are written into a grid, whose rows and columns have the following constraints:

floor(√L) ≤ rows ≤ columns ≤ ceil(√L)

For example, the sentence *"cryptography is the basis of modern cyber warfare"* is ***42*** characters long, after removing spaces. √***42*** is between ***6*** and **7**, so it is written in a grid with 6 rows and 7 columns.

```
cryptog
raphyis
thebasi
sofmode
rncyber
warfare
```

- Ensure that ***rows X columns ≥ L***
- If multiple grids satisfy the above conditions, choose the one with the minimum area (***rows X columns***)

The encoded message is obtained by displaying the characters in a column, inserting a space, and then displaying the next column and inserting a space, and so on. For example, the encoded message for the above rectangle is:

```
crtsrw rahona ypefcr phbmyf tyaoba oisder gsiere
```

Input Format
The input consists of one line of text, the plaintext string ***s***

Constraints
- ***1 ≤ |s| ≤ 81***
- s is comprised of lowercase ascii characters [a-z] only.

Output Format
Print the encoded message on one line as described.

Examples
Found in: Q3-examples.txt

4. Blocked (40 pts.)

You are given a square grid with some cells open (.) and some blocked (X).
You can move along any row or column until you reach the
edge of the grid or a blocked cell. Given a grid, a start and an end
position, determine the number of moves it will take to get to the end
position.

For example, you are given a grid with sides ***n=3*** described as follows:

```
...
.X.
...
```

Your starting position is ***(0,0)*** so you start in the top left corner. The end
Position is ***(2,1)*** so the path is ***(0,0) → (2,0) → (2,1)***. It takes 2 moves to get
to the end position.

Input Format

The first line contains an integer ***n***, the size of the array grid.
Each of the next ***n*** lines contains a string of length ***n***, the row.
The last line contains 4 space separated integers: ***StartX, StartY, EndX, EndY***

Constraints
- ***1 ≤ n ≤ 100***
- ***0 ≤ StartX, StartY, EndX, EndY ≤ n***

Output Format
Print an integer stating the minimum number of steps required to move from
the start position to the finish position.

Sample Input

```
3
...
.X.
..X
2 1 1 2
```

Here is a path that one could follow to reach the destination in ***4*** steps:
(2,1) → (2,0) → (0,0) → (0,2) → (1,2)

Examples
Found in: Q4-examples.txt