Submission Instructions

You will be uploading your submission to Gradescope. The Gradescope server may be updated (to address oversights or errors in the marking) up until Wednesday August 7th . After August 7th, the server will only be updated if absolutely necessary, and you will be notified if this happens. You must adhere to the following rules (as your submission will be subjected to automatic marking system):

1. Download the compressed file "comp2402a5.zip" from cuLearn.
2. Retain the directory structure (i.e., if you find a file in subfolder "comp2402a5", you must not remove it).
3. Retain the package directives (i.e., if you see "package comp2402a5;" in a file, you must not remove it).
4. Do not rename any of the methods already present in the files provided.
5. Do not change the visibility of any of the methods provided (e.g., do not change private to public).
6. Upload a compressed file "comp2402a5.zip" to the assignment server to submit your assignment as receive your mark immediately. By default, your last submission is your mark. However, if you click on `Submission History` you may select an earlier submission as your mark.

Please also note that your code may be marked for efficiency as well as correctness – this is accomplished by placing a hard limit on the amount of time your code wil be permitted for execution. If you select/apply your data structures correctly, your code will easily execute within the time limit, but if your choice or usage of a data structure is incorrect, your code may be judged to be too slow and it may receive a grade of zero.

You are expected to demonstrate good programming practices at all times (e.g., choosing appropriate variable names, provide comments in your code, etc.) and your code may be penalized if it is poorly written. The server won't judge this, but in case of discrepancies, this will be evaluated.

Instructions

Start by downloading the comp2402a5.zip file from Brightspace, which contains a comp2402a5 folder. This folder has one file, `IslandTrip.java`, that you will modify. In addition you may add helper functions or classes as you require. You may wish to take a look at the `ods.Graph`, `ods.AdjacencyList` or `ods.Algorithms` (where you will find breadth-first and depth-first search algorithms) classes, though these are NOT necessary to use in your solution.

Island Trip

You are given a rectangular grid of 0's and 1's, where the 0's represent water and the '1s represent land. A land-cell is connected to another if the two are adjacent (horizontally, vertically and diagonally). All the land-cells that are connected in this manner form an island and the number of land-cells in the island define the island size. In the following example, there are a total of 6 islands, their sizes being 2, 4, 5, 7, 11 and 12.

Every island is connected via bridges to the smallest island larger than itself and to the largest island smaller than itself. In addition, there is a bridge between two islands if their sizes have the same sum-of-digits. In the example, there are bridges between the islands

$$2 \leftrightarrow 4, 2 \leftrightarrow 11, 4 \leftrightarrow 6, 6 \leftrightarrow 7, 7 \leftrightarrow 11, 11 \leftrightarrow 12$$

Note that there is a bridge between 2 and 11. The sum of 2's digits is simply 2. The sum of 11's digits is $11 \to 1 + 1 = 2$. Thus there is a bridge between them.

This is not a recursive definition. That is, for an island with size 99, the sum of digits is $99 \to 9 + 9 = 18$. You would not do sum-of-digits a second time, i.e., $99 \to 9+9 = 18 \to 1+8 = 9$ is **wrong**. 18 is the correct sum-of-digits for an island with size 99.

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

Figure 1: One possible input with islands of size 2, 4, 6, 7, 11 and 12.

Your task is to output the sizes of all the islands in sorted order, followed by the minimum number of bridges that must be crossed in order to go from the smallest island to the largest. For the example, the output in the above example should be

```
2
4
6
7
11
12
2
```

The first six lines are the island sizes in sorted order. The last line represents the length of the shortest path from island 2 to island 12, which is 2, where the path is 2 ↔ 11 ↔ 12. Note: you do not need to give the actual path in your solution - just the length.

You can make the following assumptions:

- You can read the input one line at a time, each line being a string of 0's and 1's.
- All the lines have the same length.
- There is at least one island.
- All the islands have unique sizes.

Hint: Most of the marks are for correctness. However, the tests on the server are heavily memory constrained. Storing the grid in a straightforward manner may not be the best idea. Think how you can optimize storage when you are storing the grid in memory. Speed will also be a factor in the later tests.

The final two tests are for bonus marks. Your solution will have to be *very* memory efficient and *very* fast to achieve these marks.