



Algorithms and Data Structures in C


Practical V - "Maze"

Deadline: Sunday, March 30, 2025, at 23:59

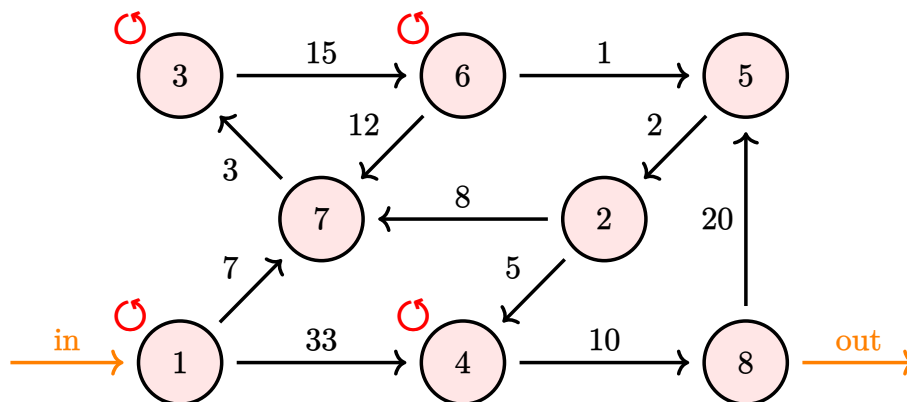
In this last assignment, you will use Dijkstra's algorithm to determine the shortest path to the exit of a special maze. You should also write a programming report. You have two weeks time for this.

Problem statement



After having implemented a filesystem in the previous assignment, you went to the amusement park to take a break and enjoy yourself. However, upon arriving, you found out the park has a new attraction: the *Inverso-Maze* — which you cannot wait to write a program for.

The *Inverso-Maze* is a maze consisting of various underground *chambers*, which are connected to each other via *tunnels*. The tunnels have special gates which make sure they can only be traversed in one direction, although this direction changes every time a -button is pressed somewhere in the maze.

The *Inverso-Maze* could look as follows:




The numbered vertices of this graph represent the chambers of the maze, and the edges represent the tunnels between them. The labels near the edges denote the lengths of the corresponding tunnels, and the direction of each edge represents the direction in which that tunnel may initially be traversed.

The chambers 1, 3, 4 and 6 are marked with  in the picture. This means these chambers have a -button, which – upon pressing – causes all tunnel gates to reverse the direction in which they allow traveling. In other words, these buttons let you switch the direction of all tunnels at the same time.

Your goal is to write a program that takes an *Inverso-Maze* as input, and outputs the shortest path from the entrance to the exit. For example, in the maze given by the picture above, the shortest path is

$$1 \rightarrow 7 \rightarrow 3 \xrightarrow{\text{button}} 7 \rightarrow 2 \rightarrow 5 \rightarrow 6 \xrightarrow{\text{button}} 5 \rightarrow 2 \rightarrow 4 \rightarrow 8,$$


and it has the length $7 + 3 + 3 + 8 + 2 + 1 + 1 + 2 + 5 + 10 = 42$. Here, the -button is pressed in chambers 3 and 6, but not in chambers 1 and 4.



Input and output

Input

The first line of the input contains the integer n , the number of chambers, and the integer m , the number of tunnels (where $n \geq 1$ and $m \geq 0$). The chambers are numbered 1 to n . Chamber 1 is the entrance, and chamber n is the exit.


The second line of the input lists all the chambers that have a -button, terminated by -1.

Then, m more lines follow. Each of these lines contains the integers a , b and ℓ (where $a, b \in \{1, 2, \dots, n\}$ and $\ell \geq 0$), indicating that there is a tunnel between chambers a and b with length ℓ , which may initially be traversed in the direction $a \rightarrow b$.

All numbers in the input fit in an **int** (on Themis, these are 32 bits).

Output

Your output should start with a line containing a single integer: the length of the shortest path from chamber 1 to chamber n . It is guaranteed that this number is less than `INT_MAX`.

After this, you should print the numbers of the chambers of your shortest path in order of traversal, each chamber on a separate line. When the -button is pressed in a chamber, you should print 'R' after that chamber number. You should never print 'R' after chamber n (the final chamber).

If there is no path from chamber 1 to chamber n , you should output `IMPOSSIBLE`.



Note.

The test cases are such that the shortest path, if it exists, is always unique.

Example

For the example maze given on the first page, the input and output could look as follows:

Input:

```
8 11
6 3 4 1 -1
7 3 3
1 4 33
5 2 2
2 4 5
2 7 8
3 6 15
1 7 7
6 7 12
6 5 1
4 8 10
8 5 20
```

Output:

```
42
1
7
3 R
7
2
5
6 R
5
2
4
8
```

Bonus: A* algorithm

In the bonus exercise, you will make your program more efficient by extending it to use the A* algorithm.

The input for the bonus exercise consists of two parts. The first part of the input is identical to the input for the regular assignment. The second part of the input consists of n more lines, each containing the integers x_i and y_i . These lines specify the coordinates of all the n chambers, given in the order of increasing chamber number.

You should implement the A* algorithm to – again – find the shortest path from chamber 1 to n . You can make use of the Pythagorean theorem to determine the geometric distance between two chambers: it is guaranteed that the actual shortest distance (via the tunnels) is lower bounded by this distance.

The output format is identical to the output format of the regular assignment.

For brevity, we do not include a full input/output example for the bonus assignment, but you can find a number of visible test cases on Themis.

Note: the Teaching Assistants will check that you are actually using the A* algorithm.

Grading

For this assignment, you can get 11 points in total:

- The test cases for the regular assignment are worth 3 points.
- The test cases for the bonus assignment are worth 1 point.
- The report is worth 5 points.
- You can get up to an additional 2 points for code quality.

Note that **not all test cases are worth the same amount of points**; Themis indicates how much points each passed test case is worth.

For the report, you are required to use the template from Brightspace and follow Appendix E of the *Lecture notes*. Furthermore, recall that the *Grading document* on Brightspace lists a number of things you can be subtracted points for, both regarding the report and regarding code quality.



Alert!

Dijkstra's algorithm is well-known, and many implementations exist online. In order to learn the most, we ask you to write your own solution without looking up code online.

Any cases of plagiarism will be forwarded to the Board of Examiners.



Note.

For this assignment, efficiency is very important. Consider this from the beginning onwards when designing your program. Inefficient programs will not pass all the test cases.