DEI
Departamento de Engenharia Informática

Advanced Algorithms

2022/2023

Project 2

Build a Navy Network

The first delivery deadline for the project code is the 24th of March, at 12:00. The deadline for the recovery season is the 14th July at 12:00, the system opens the 12th July at 12:00. The deadline for the special season is 26th July at 12:00, the system opens the 24th July at 12:00.

Students should not use existing code for the algorithms described in the project, either from software libraries or other electronic sources. They should also not share their implementation with colleagues, specially not before the special season deadline.

It is important to read the full description of the project before starting to design and implementing solutions.

The delivery of the project is done in the mooshak system.

1 Network Modeling

1.1 Overview

In this project you need to model the following problem using an algorithm taught in class. Your role is to design a transportation network connecting several cities. The objective is to connect several cities by highway connections, or by shipping ports. You are given the map of the cities and the costs of creating connections between them.

Your goal is to guarantee that there is a route between any pair of cities. This route might be a direct connection or it may pass through other cities.

In the construction it is assumed that at least one city must build a shipping port. Otherwise the configuration is considered impossible. Moreover any city that may construct a shipping port must do so as this is considered a strategic investment. Cities that have shipping ports are considered connected to any other city that also possesses an shipping port. Cities that do not possess a shipping port must be connected by highways. We assume that all connections work in both directions, both highways and shipping port.

Your program should decide which highways to build and where to build shipping ports so that all cities are part of the network and the overall construction cost is minimized.

1.2 Specification

To automatically validate the index we use the following conventions. The binary is executed with the following command:

```
./project < in > out
```

The file `in` contains the input commands that we will describe next. The output is stored in a file named `out`. The input and output must respect the specification below precisely. Note that your program should **not** open or close files, instead it should read information from `stdin` and write to `stdout`. The output file will be validated against an expected result, stored in a file named `check`, with the following command:

```
diff out check
```

This command should produce no output, thus indicating that both files are identical.

The format of the input file is the following:

- One line containing the number N ($N \geq 2$) of cities.
- One line containing the number A ($0 \leq A \leq N$) of potential shipping ports.
- A sequence of A lines. In each line there are two integers a and c (separated by a white space), where c represents the cost of building a shipping port in city a.
- One line containing the number E of potential highways.
- A sequence of E lines. In each line there are three integers a, b and c (separated by a white space), where c represents the cost of building a highway connection between cities a and b.

Assume that the cities are numbered from 1 to N.
The input contains no more data.
The format of the output file should be the following:

- In case it is impossible to build the network the output should consist of a single line with the word `Impossible`.
- One line with the total cost of the network.
- One line with two integers separate by a white space. The first number indicates how many shipping ports to build and the second number how many railway connections.

All lines must be terminated by and end of line character `'\n'`.

1.3 Sample Behaviour

The following examples show the expected `output` for the given `input`. These files are available on the course web page.

input 1

```
4
3
1 1
2 5
3 1
4
1 2 1
1 3 6
2 4 2
3 4 3
```

output 1

```
9
3 1
```

input 2

```
4
3
1 1
```

```
2 5
3 1
2
1 2 1
1 3 6
```

output 2

```
Impossible
```

input 3

```
4
4
1 1
2 5
3 2
4 10
2
1 2 1
1 3 6
```

output 3

```
18
4 0
```

input 4

```
4
3
1 1
2 5
3 1
4
1 2 1
1 3 6
2 4 2
3 4 2
```

output 4

```
9
3 1
```

2 Grading

The mooshak system is configured to a total 40 points. The project accounts for 4.0 values of the final grade. Hence to obtain the contribution of the project to the final grade divide the number of points by 10. To obtain a grading in an absolute scale to 20 divide the number of points by 2.

Each test has a specific set of points. The first four tests correspond to the input output examples given in this script. These tests are public and will be returned back by the system. The tests numbered from 5 to 12 correspond to increasingly harder test cases, brief descriptions are given by the system. Tests 13 and 14 are verified by the valgrind[1] tool. Test 13 checks for the condition `ERROR SUMMARY: 0 errors from 0 contexts` and test 14 for the condition `All heap blocks were freed -- no leaks are possible`. Test 15 to 17 are verified by the lizzard[2] tool, the test passes if the `No thresholds exceeded` message is given. Test 15 uses the arguments `-T cyclomatic_complexity=15`; test 16 the argument `-T length=150`; test 17 the argument `-T parameter_count=9 -T token_count=500`. To obtain the score of tests from 13 to 17 must it is necessary obtain the correct output, besides the conditions just described.

The mooshak system accepts the C programming language, click on `Help` button for the respective compiler. Projects that do not compile in the mooshak system will be graded 0. Only the code that compiles in the mooshak system will be considered, commented code will not be considered for evaluation.

Submissions to the mooshak system should consist of a single file. The system identifies the language through the file extension, an extension `.c` means the C language. The compilation process should produce absolutely no errors or warnings, otherwise the file will not compile. The resulting binary should behave exactly as explained in the specification section. Be mindful that `diff` will produce output even if a single character is different, such as a space or a newline.

Notice that you can submit to mooshak several times, but there is a 10 minute waiting period before submissions. You are strongly advised to sub-

[1]`https://www.valgrind.org/`

[2]`https://github.com/terryyin/lizard`

mit several times and as early as possible. Only the last version is considered for grading purposes, all other submissions are ignored. There will be **no** deadline extensions. Submissions by email will **not** be accepted.

References

Cormen, T. and Leiserson, C. and Rivest, R. and Stein, C. *Introduction to algorithms*. The Massachusetts Institute of Technology, 2nd Edition, 2001.