TÉCNICO LISBOA

DEI
Departamento de Engenharia Informática

Advanced Algorithms

2023/2024

Project 2

Connect a network

The first delivery deadline for the project code is the 19th of March, at 12:00. The deadline for the recovery season is 9th July at 10:00, the system opens the 7th of July at 10:00. The deadline for the special season is 23th July at 10:00, the system opens the 21th at 10:00.

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. Your are given an allotted budget that can not be exceeded and a list of possible road constructions. Each road connects two cities in such a way that transit can flow in both directions. As a systems designer you need to balance opposing goals. On the one hand your goal is to try to make sure that as many cities as possible have access to the transportation network, i.e., that each city has a route to as many other cities as possible. A route

does not need to be direct connection. Any sequence of connections between two cities is equally effective. On the other hand there is a fixed budget that must be respected. Hence your proposal must have an overall cost bellow this value, even if there ends up existing disconnected portions of the network, i.e., a pair of cities for which there is no viable route. Furthermore even within this scenario you proposal should be as small as possible. More precisely if it is not possible to connect the whole network your proposal should be such that any other road that could connect disconnected portions is over budget and the proposed configuration requires the minimum cost among such configurations.

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 budget B ($B \geq 0$).
- One line containing the number E of possible roads.
- 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 road 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:

- One line with the total cost of the network that is possible to build.

- The number of components in resulting network.

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

output 1

```
2
2
```

input 2

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

output 2

```
1
3
```

input 3

```
6
975
5
2 4 384
3 1 247
2 1 377
2 6 339
2 5 171
```

output 3

```
757
3
```

input 4

```
6
10
15
1 2 6
1 3 6
1 4 6
1 5 6
1 6 6
2 3 5
2 4 5
2 5 5
2 6 5
3 4 4
3 5 4
3 6 4
4 5 3
4 6 3
5 6 2
```

output 4

```
9
3
```

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 submit 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.

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

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

References

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