Exercise 3,4

COMP 6651 – Winter 2018

Dr. Tiberiu Popa

Introduction

Due to time constraints and the due date of the final examination we compressed E3 and E4 into one exercise that has a theoretical component and a programming component.

Theoretical Exercises

For the theory part you have to solve the following exercises from the textbook (3rd, edition): 22.2-2, 22.2-4, 22.2-8, 22.3-7, 22.4-2.

Additionally, you have to show that for any connected, positively weighted, undirected graph whose weights obey the triangle inequality, the shortest distance between two vertices in terms of sum of weights of the path is equal to the distance of the path induces by the BFS tree with one of the vertices as root.

Your solution has to be typed and submitted as a PDF together with your programming part, see below for further details. (no complete or partially handwritten solution will be accepted).

Programming Exercise

For this exercise you are asked to solve the "*Traveling Salesman Problem*": given a complete, undirected weighted graph of N>2 vertices you have to find the minimum Hamiltonian cycle. A Hamiltonian cycle is a cycle that visits all vertices exactly once (with the exception of the first and last vertex). You can assume that the weights are positive, but you cannot assume that the weights obey the triangle inequality.

Specifications

The input is specified in a file whose name is the first argument of the program. The first line contains an integer M specifying how many datasets are in the file. The reminder of the file encodes the datasets. Each dataset consists of a definition of a graph as follows: It starts with two space separated positive integers V and E on the first line that indicates the number of vertices and edges respectively. The following lines specify the weights on the edges of the graph as following: each edge is specified on one line as 3 numbers: u v w. u is the index of the first vertex, v is the index of the second vertex and w is a positive integer weight of the edge (u,v). Note that all edges of the graph appear in this list exactly once. (i.e. if we have an entry for (u,v) we will not have one for (v, u)).

Here is an example:

```
2
3 3
0 1 1
0 2 1
1 2 1
4 6
0 1 2
0 2 1
0 3 2
1 2 2
1 3 1
2 3 2
```

The output is a file called whose name is the second argument of the program. Each line encodes the results of each test case. The algorithm should output the length of the minimal Hamiltonian cycle.

For example, the output corresponding to the input above is as follows:

```
3
6
```

What is given

No code is given for this exercise.

Submission

The theoretical part you have to solve in a file called ***theory.pdf***. The solution must be typed (i.e. scans of any kinds are not acceptable).

The programming part you have to implement your program in plain C/C++ in a file called *main.cpp* that has no dependency other than the standard C/C++ libraries available in a standard Linux system such as the one in the graduate labs. The code should compile using the command ***g++ main.cpp*** in any machine in the graduate labs. **You are not allowed to use any other flags for this homework.**

You need to combine the **theory.pdf** file and **main.cpp** file into one zip file called **E4.zip**. If the zip file does not contain ONLY these two files the solution will receive a grade of 0.

Submit the exercise under "**Assignment 3**". The due-date is **April 17th, 5pm.** This deadline is a hard deadline so any late submission will receive a grade of 0.

This exercise is individual.

Originality and Plagiarism

Some of the problems proposed in the course are “classical” in the algorithm design literature and, therefore, solutions may be available on-line. Please note that you are expected to do this problem individually and you are expected to produce an original solution. We run plagiarism tests and if your submission is red-flagged you are expected to explain it in detail to the instructor. Failure to comply with this request or to adequately explain your own code may result in filing a plagiarism case with the Dean of the Faculty of Engineering.

Evaluation and Testing

The theoretical exercise will be evaluated by a human.

The programming component will be evaluated automatically, but we do check if you implemented the right algorithm and if the code is original. You will receive 0 if the code does not compile. We run 3 test cases, you receive 1/10 marks if your code compiles, but it does not return the correct results on any of the test cases, 4/10 if it compiles and runs correctly on one of the test cases, etc.

You may be asked to explain your code to the teaching assistant or the instructor. This may lead to a decrease in your overall grade for this exercise.