ALGORITHMICS - Bachelor of Software Engineering, 2023

LAB GUIDE. SESSION 4

GOALS:

- **Greedy algorithms: PRIM**

1. Introduction

Prim's algorithm is one of the clearest examples (perhaps the best) of a greedy algorithm. Given a graph of **n** nodes, it chooses safely and quickly, edge by edge, until **n-1** edges are selected, obtaining the way to have the **n** nodes connected with the lowest cost (optimal cost).

As always, when implementing an algorithm, the data structures chosen will determine greater or lesser complexity. For the subsequent implementation of this algorithm that is required of us, any complexity will be accepted, although we'll positively assess lower complexities.

2. Algorithm of Prim

The **helper.py** module is provided, which has functions that allow you to create a complete random graph in a matrix and load it from a text file. Of course, you can modify and add to that module whatever you consider appropriate.

The files **graph4.txt**, **graph8.txt**, … **graph256.txt** contain complete graphs with bidirectional edges (with weights in the interval [100…999]). The format of these files involves placing in the first line the number **n** of nodes, in the second line the weights of the edges from the first node (node 0) to all the others (in order), in the third line the weights of the edges from the second node (node 1) to all others with a higher index, and so on.

The files **result4.txt**, **result8.txt**, … **result256.txt** contain the solution, calculated by Prim, for each of the input graphs seen before. If there are several optimal solutions with the same minimum cost, any of them will work, so there could be other valid solutions to those proposed, but obviously they must have the same optimal minimum cost.

YOU ARE REQUESTED TO:

Implement a **Prim.py** module, so that after entering the name of a file, it calculates and writes the optimal solution on the screen.

Explain the time complexity of the implemented algorithm.

Implement a **PrimTimes.py** module that creates random graphs of various sizes and calculates the time it takes for the algorithm made in the previous section to solve the problem, so that the following table can be filled in.

Does the previously calculated complexity follow the times in the table?

TABLE FOR PYTHON ALGORITHM OF PRIM

(times in milliseconds):

We can type "OoT" for times over 10 minutes.

n	***t Prim***
256	
512	
1024	
2048	
4096	
16384	
… (*until OoT*)	

3. Work to be done

- An `algstudent.s4` **package** in your course project. The content of the package should be the files used and created during this session.
- A `session4.pdf` **document** using the course template (the document should be included in the same package as the code files). You should create one activity each time you find a "YOU ARE REQUESTED TO" instruction.

Deadline: The deadline is one day before the next lab session of your group.