CS325: Analysis of Algorithms, Fall 2017

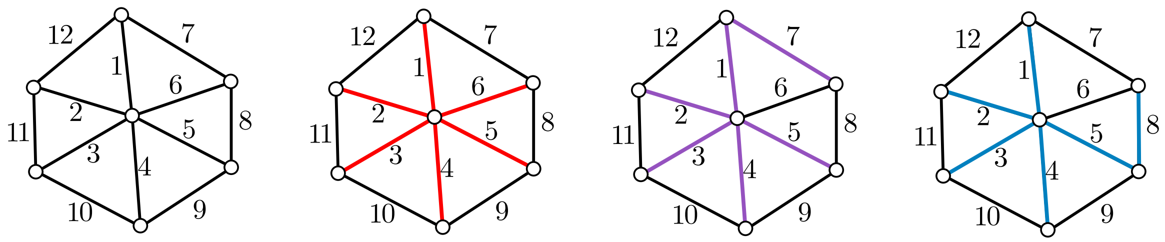
Group Assignment 3[*]

Due: Tue, 11/14/17

Homework Policy:

1. Students should work on group assignments in groups of preferably three people. Each group submits to TEACH a zip file that includes their source code and their *typeset* report. Each group, also, hands in a printed hard copy of the report in class or slides the hard copy under my door before the midnight of the due day. The hard copy will be graded, and the codes submitted to teach will be tested.
2. The goal of the homework assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
3. You are allowed to discuss the problems with other groups, and you are allowed to use other resources, but you *must* cite them. Also, you must write everything in your own words, copying verbatim is plagiarism.
4. *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a completely wrong answer will receive zero.
5. Algorithms should be explained in plain english. You can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.
6. More items might be added to this list. ☺

In class, we have seen multiple algorithms for computing minimum spanning trees. In this assignment, we design algorithms for computing the second and third minimum spanning trees. For example, the following figure shows a graph, and its first, second, and third minimum spanning trees.



Report (70%). In your report, describe and analyze an algorithm to compute the first, second, and third minimum spanning trees. Prove that your algorithm is correct. (For full credit, the algorithm must run in $O(V^2 E \log V)$. However, obtaining an $O(VE)$ time algorithm is not too hard.)

[*]The problem is from Jeff Erickson's lecture notes. Looking into similar problems from his lecture notes on recursion is recommended.

Code (30%). Write a program to compute first, second, and third minimum spanning trees of a given graph G. Your program will be tested against several test cases, for correctness and efficiency. For each test case, the program will be automatically stopped after 20 seconds if it is not done in that time. In this case, the group will miss the points of that test case. Your program must be written in one of the following languages: Python, C++, or Java.

Input/Output Your program reads from one text file, "input.txt". The first line is one integer $1 \leq n \leq 30$. The following n lines each is a row of the adjacency matrix of G, that is the ith number of the jth row is the weight of the the (i, j) edge (You can assume that the graph is complete, and all weights are non-negative integers). (There will be test cases with 100 vertices for extra credit.)

Your output must be written in "output.txt". The output is composed of three numbers, each written in a separate line: the weights of the first, second, and third minimum spanning trees in order.

Sample Input (1):
3
0,1,2
1,0,3
2,3,0

Sample Output (1):
3
4
5

Sample Input (2):
5
0,1,9,9,1
1,0,1,9,9
9,1,0,1,9
9,9,1,0,1
1,9,9,1,0

Sample Output (2):
4
4
4

Submission Each group submits to TEACH a zip file that includes their source code (*which must be just one file with name "mst.cpp", "mst.java", or "mst.py"*) and their report in pdf format. This file can be submitted by any member of the group, but all names must be listed in the submitted report. Each group, also, hands in a printed hard copy of the report in class or slides the hard copy under the door of my office before the midnight of the due day. The hard copy will be graded, and the submitted code to teach will be tested.

Your codes will be tested *automatically.* So, you need to carefully follow all formatting requirements mentioned above. To summarize:

(1) Your source code file should be named "mst.cpp", "mst.java", or "mst.py".

(2) It reads from files “input.txt” in the current folder.

(3) It writes to the file “output.txt” in the current folder. It should write exactly one number into “output.txt”, with no extra symbol.

Test your code with the sample test files (`http://web.engr.oregonstate.edu/~nayyeria/CS325/Fall17/hws/test3.zip`) before submitting them, to make sure there is no formatting error.