CS325: Analysis of Algorithms, Fall 2020

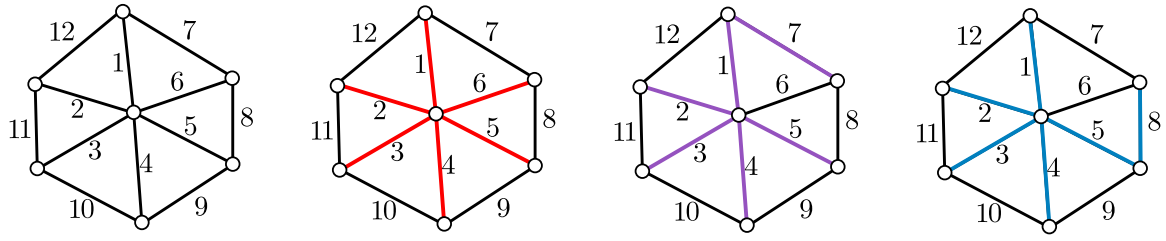
Group Assignment 4*

Due: Tue, 11/24/20

Homework Policy:

1. Students should work on group assignments in groups of preferably three people. Each group submits to CANVAS a zip file that includes their source code and their *typeset* report. Specifically, for this assignment your zipped folder should contain two files named `assignment4.pdf`, and `assignment4.py`. One submission from each group is sufficient.
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.



(**Hint:** How different are the first minimum spanning tree and the second minimum spanning tree? How about the first and third minimum spanning trees?)

*Looking into MST chapter of Jeff's book is recommended.

Report (60%). In your report, include the description of your algorithm, and provide running time analysis and proof of correctness. 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. Proof of correctness is more important for this assignment. To get an idea on how to solve this problem, and how to write the proof of correctness, I recommend reviewing the main claim of the MST lecture. 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.

Code (40%). You will complete and submit the `assignment4.py` file to compute best, second and third minimum spanning trees. The following template is provided. You need to implement the function *first_second_third_mst*. This function has two parameters: *input_file_path* and *output_file_path*, specifying the full path of the input and output files, respectively. When called, the function *first_second_third_mst* should read the game from the input file, and writes three numbers in output file, in three separate lines, which are the total weight of the first, second and third minimum spanning trees. Note that the total weight of these trees might be equal; see the second sample input for an example.

```
'''
    This file contains the template for Assignment4.  For testing it, I will place it
    in a different directory, call the function <first_second_third_mst>, and check its output.
    So, you can add/remove  whatever you want to/from this file.  But, don't change the name
    of the file or the name/signature of the following function.

    Also, I will use <python3> to run this code.
'''

def first_second_third_mst(input_file_path, output_file_path):
    '''
    This function will contain your code, it will read the input from the file
    <input_file_path> and write to the file <output_file_path>.

    Params:
        input_file_path (str): full path of the input file.
        output_file_path (str): full path of the output file.
    '''
    pass

# first_second_third_mst('input0.in', 'input0.out')
```

Tests. 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. **Note:** it is important that your output is formatted as described below, since your codes will be tested automatically.

Input/Output. The input file is formatted as follows. The first line is one integer $1 \leq V \leq 30$, which is the number of vertices of the input weighted undirected graph G. (There will be test cases with ≥ 100 vertices for extra credit.)

The following V lines each is a row of the *weighted* 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 positive integers.

The output file is composed of three different numbers each written in a separate line, which are the weights of the lightest, second lightest, and third lightest 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

Practice Problems

Do *not* submit solutions to the following problem. However, I recommend to work on them as good practice problems for graphs and graph algorithms.

Minimum Spanning Trees. The following problems from Chapter 7 of the book (`https://jeffe.cs.illinois.edu/teaching/algorithms/book/07-mst.pdf`): 1, 5, 6, 9