CS325: Analysis of Algorithms, Winter 2020

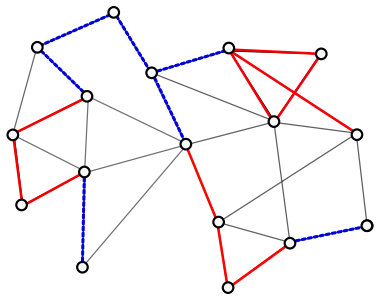
Group Assignment 3

Due: Tue, 3/3/20

Homework Policy:

1. Students should work on group assignments in groups of at most and 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. ☺

Let $G = (V, E)$ be a graph on V, and let $w : E \to \mathbb{R}$ be edge weights. Also, let $E' \subseteq E$ be a subset of edges that are already selected. We would like to compute the minimum weight subset of edges $E^* \subseteq E$ such that $G = (V, E^* \cup E')$ is connected. For example, in the following figure the red (solid) edges compose E' and the blue (dashed) edges compose a candidate for E^* so that $G = (V, E^* \cup E')$ is connected.



Design and analyze an algorithm to compute E^* with minimum total weight. The running time of your algorithm should be similar to the minimum spanning tree's running time.

Report (60%). In your report, include the description of your algorithms, running time analysis, and proof of correctness. 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%). Implement a geometric version of your algorithm. Your input is a set of points on the plane, with a subset of edges E'. The distance between any pair of points is their so called Manhatan distance: the distance between (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$. Your output will be a single number, the total (Manhatan) length of E^*.

Submission. Submit a python program for the geometric problem. 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.

Specifically, you must implement the function "minimum_cost_connecting_edges" in the following code. The code you submit will be an implementation of this procedure in a file named "assignment3.py".

```
'''
    This file contains the template for Assignment3.  For testing it, I will place it
    in a different directory, call the function <minimum_cost_connecting_edges>, 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 minimum_cost_connecting_edges(input_file_path, output_file_path):
    pass

'''
    To test your function, you can uncomment the following command with the the input/output
    files paths that you want to read from/write to.
'''
# minimum_cost_connecting_edges('input1.in', 'input1.out')
```

Input/Output The input file is composed of two lines. The first line specifies our set of points $P[1 \dots n]$, each point has integer coordinates between 0 and $1000,000$. Also, $0 \leq n \leq 1000$. The second line specified E'. Each edge of E' is a pair from $\{1, \dots, n\} \times \{1, \dots, n\}$. You can assume that $|E'| \leq 10000$. If E' is the empty set, the second line is "none".

The output file must composed of one line, which is a single integer: the total weight of E^*.

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

Sample Output (1):
2

Sample Input (2):
(0,0),(2,2),(3,3),(2,3),(3,2),(5,5)
(2,3),(2,4),(3,4),(2,5)

Sample Output (2):
8

Sample Input (3):
(0,0),(0,1),(0,2),(0,12),(0,20)
(1,2),(1,3),(4,5)

Sample Output (3):
10

Sample Input (4):
(0,0),(0,1),(0,2),(0,12),(0,20)
none

Sample Output (3):
20

Extra Credit – Test cases (∼10%). I will test your code for large test cases with $n >> 1000$ coming from uniformly distributed points in the rectangle $[0, 1000, 000] \times [0, 1000, 000]$. You receive extra points if your code is meaningfully faster than the other codes. You can assume that the second line of the input for all these extra credit tests is "none".