CS 325- Homework Assignment 4

Problem 1: *(5 points)* **Road Trip**:

Suppose you are going on a road trip with friends. Unfortunately, your headlights are broken, so you can only drive in the daytime. Therefore, on any given day you can drive no more than d miles. You have a map with n different hotels and the distances from your start point to each hotel $x_1 < x_2 < \ldots < x_n$. Your final destination is the last hotel. Describe an efficient greedy algorithm that determines which hotels you should stay in if you want to minimize the number of days it takes you to get to your destination. What is the running time of your algorithm?

Problem 2: *(4 points)* **CLRS 16-1-2 Activity Selection Last-to-Start**

Suppose that instead of always selecting the first activity to finish, we instead select the last activity to start that is compatible will all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution.

Problem 3: *(6 points)* **MST**

Consider an undirected graph G=(V,E) with nonnegative edge weights $w(u,v) \geq 0$. Suppose that you have computed a minimum spanning tree T, and that you have also computed shortest paths to all vertices from vertex $s \in V$. Suppose each edge weight is increased by 1: the new weights $w'(u,v) = w(u,v) + 1$.

(a) Does the minimum spanning tree change? Give an example if it changes or prove it cannot change.

(b) Do the shortest paths change? Give an example of a graph where the paths change or prove that the paths cannot change.

Problem 4: *(15 points* **Euclidean MST Implementation**
Implement an algorithm in the language of your choice (C, C++ or Python) that determines the MST in a graph $G=(V,E)$ where the vertices are points (x,y) in the rectangular coordinate system and the weight of the edge between each pair of points is the Euclidean distance between those points. To avoid floating point precision problems in computing the square-root, we will always round the distance to the nearest integer. For all $u, v \in V$, $u = (x_1,y_1)$, $v = (x_2,y_2)$ and

$$w(u,v) = d(u, v) = nearestint\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

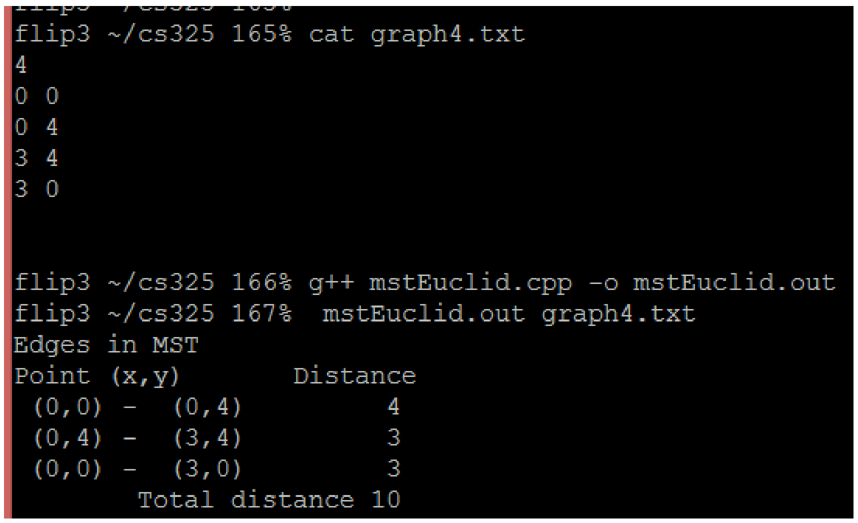
For example, if u = (0, 0), v = (1, 3),

$$\begin{aligned} d(u,v) &= \text{nearestint}\left(\sqrt{(0-1)^2+(0-3)^2}\right) \\ &= \text{nearestint}\left(\sqrt{(-1)^2+(-3)^2}\right) \\ &= \text{nearestint}(\sqrt{1+9}) \\ &= \text{nearestint}(\sqrt{10}) \\ &= \text{nearestint}(3.1622\ldots) \\ &= 3 \end{aligned}$$

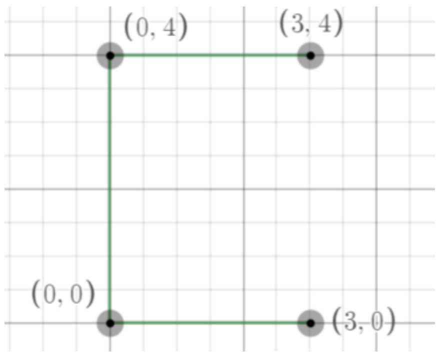
Assume that the graph $G=(V,E)$ is complete so that there is an edge between every two vertices.

Your program (mstEuclid) should read information about the graph from a file given in the command line. Each input file contains information about one graph: the first line in the file is the number of vertices in the graph (n ≤ 100) and the following lines contain the x,y coordinates of the vertices in the graph. The output should be displayed to the terminal and contain the edges (x1,y2) to (x2,y2) in the MST along with the edge distances and the total distance of the MST.

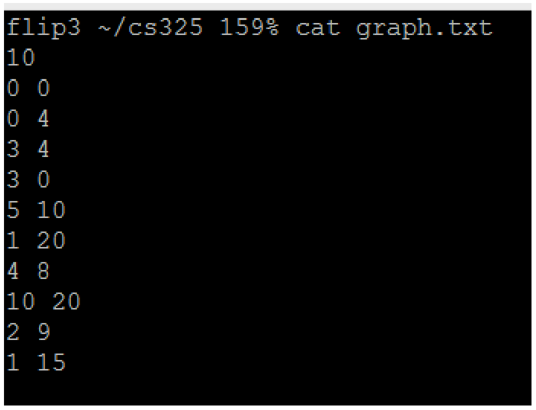
Below is an example of graph4.txt which contains four points (0,0), (0,4), (3,4) and (3,0). The MST has a total distance of 10.



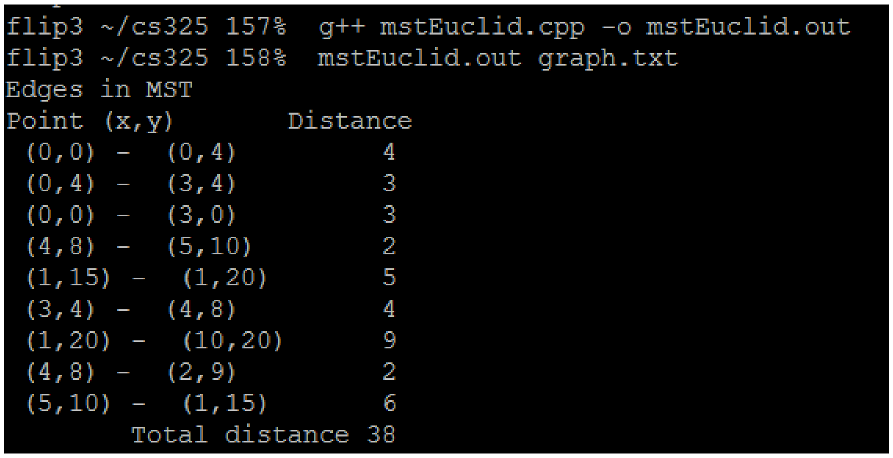
```
flip3 ~/cs325 165% cat graph4.txt
4
0 0
0 4
3 4
3 0


flip3 ~/cs325 166% g++ mstEuclid.cpp -o mstEuclid.out
flip3 ~/cs325 167%  mstEuclid.out graph4.txt
Edges in MST
Point (x,y)        Distance
 (0,0) -   (0,4)          4
 (0,4) -   (3,4)          3
 (0,0) -   (3,0)          3
        Total distance 10
```



Below is an example of the program executed with the input file graph.txt.



```
flip3 ~/cs325 159% cat graph.txt
10
0 0
0 4
3 4
3 0
5 10
1 20
4 8
10 20
2 9
1 15
```



```
flip3 ~/cs325 157%  g++ mstEuclid.cpp -o mstEuclid.out
flip3 ~/cs325 158%  mstEuclid.out graph.txt
Edges in MST
Point (x,y)       Distance
 (0,0) -  (0,4)         4
 (0,4) -  (3,4)         3
 (0,0) -  (3,0)         3
 (4,8) -  (5,10)        2
 (1,15) -  (1,20)       5
 (3,4) -  (4,8)         4
 (1,20) -  (10,20)      9
 (4,8) -  (2,9)         2
 (5,10) -  (1,15)       6
        Total distance 38
```

Submit to Canvas:

- A verbal description of your algorithm and data structures
- The pseudo-code for your algorithm.
- Analysis of the theoretical running time of your algorithm.

Submit to TEACH

- README file with instructions on how to compile and run your code
- All code files