Instructions:

This lab must be submitted to the Lab 11 drop-box in D2L before the end of today’s lab. Answer the following questions for the Solozol problem. Please type your answers with blue color in this word file.

Questions:

1. What type of problem is this (i.e.: shortest path, topological sort, spanning tree, union find, etc.)?

Minimum Spanning tree problem

1. Which algorithm (that you know) can be used (with enhancements) to solve this problem? (if more than one algorithm can be used, explain why you selected the one that you did)

There’s two algorithm that can be used to solve this problem:

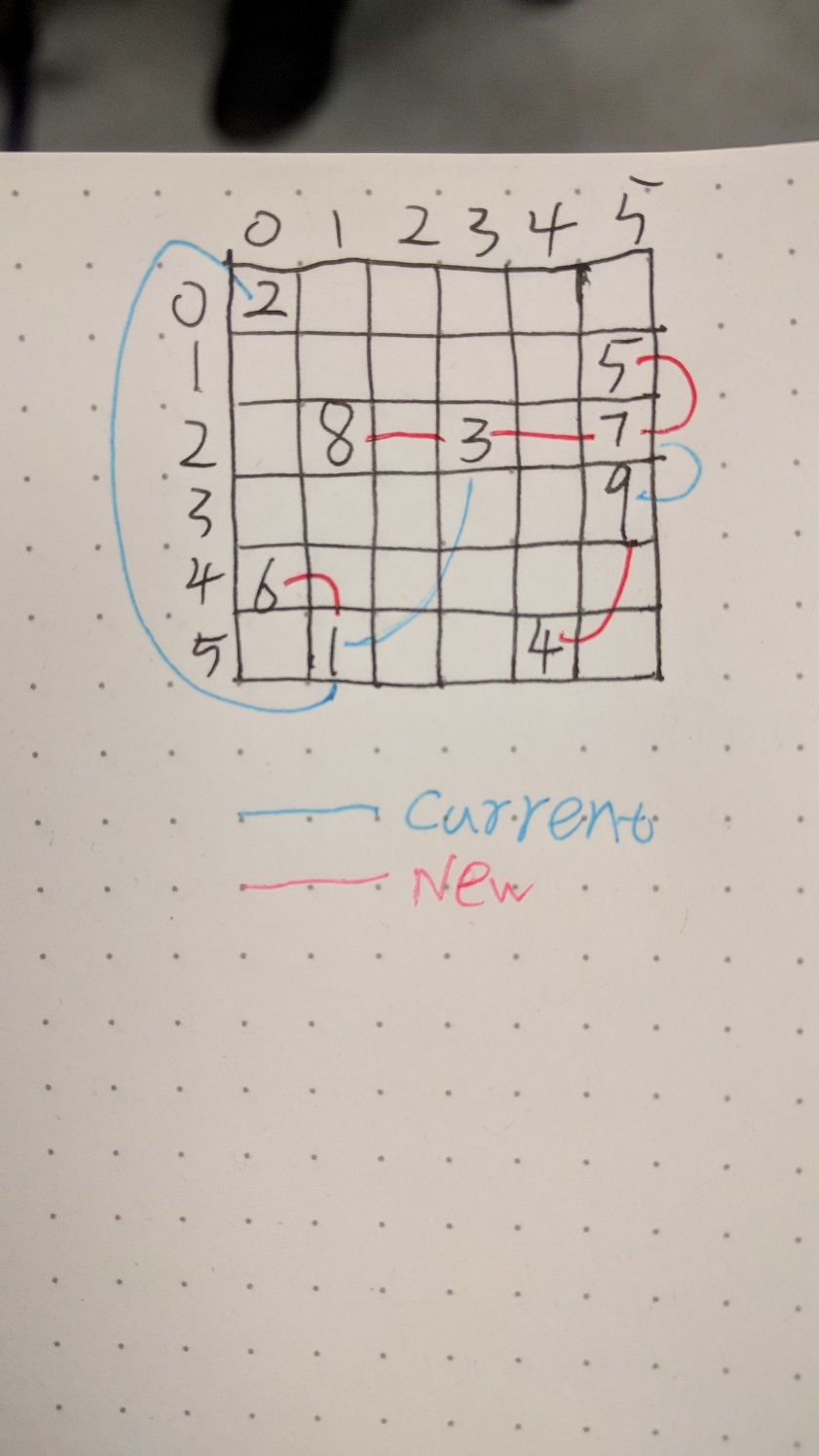
Prim -> (O(vlogv + elogv))

Kruskal -> O(|E| log |E|)

I would choose Kruskal because since there’s some Towns already connected by highway.

1. Explain how you can model the problem as a graph. What do vertices, edges etc represent? Draw the graph that you corresponds to the sample input that is provided with the problem.

Vertices -> Town

 Edge -> highway (existed or will build)

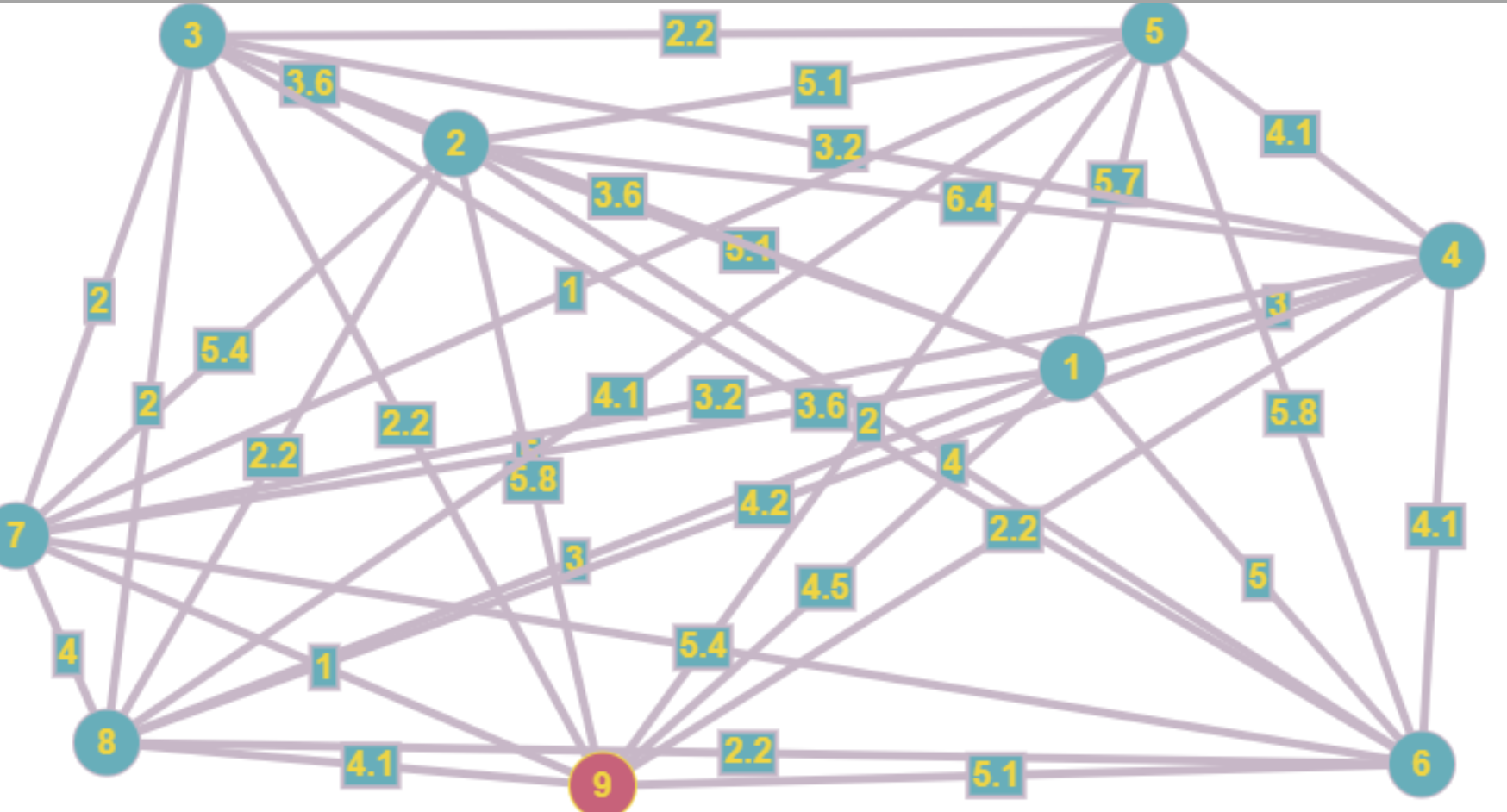
1. Describe any modifications/enhancements/changes required to make the algorithm you selected work for this problem. Describe your enhancements in plain English, and provide pseudocode if you feel it will help explain what you are doing.

To Apply my algorithm, I will make sure only connect the vertices that haven’t connected to each other.

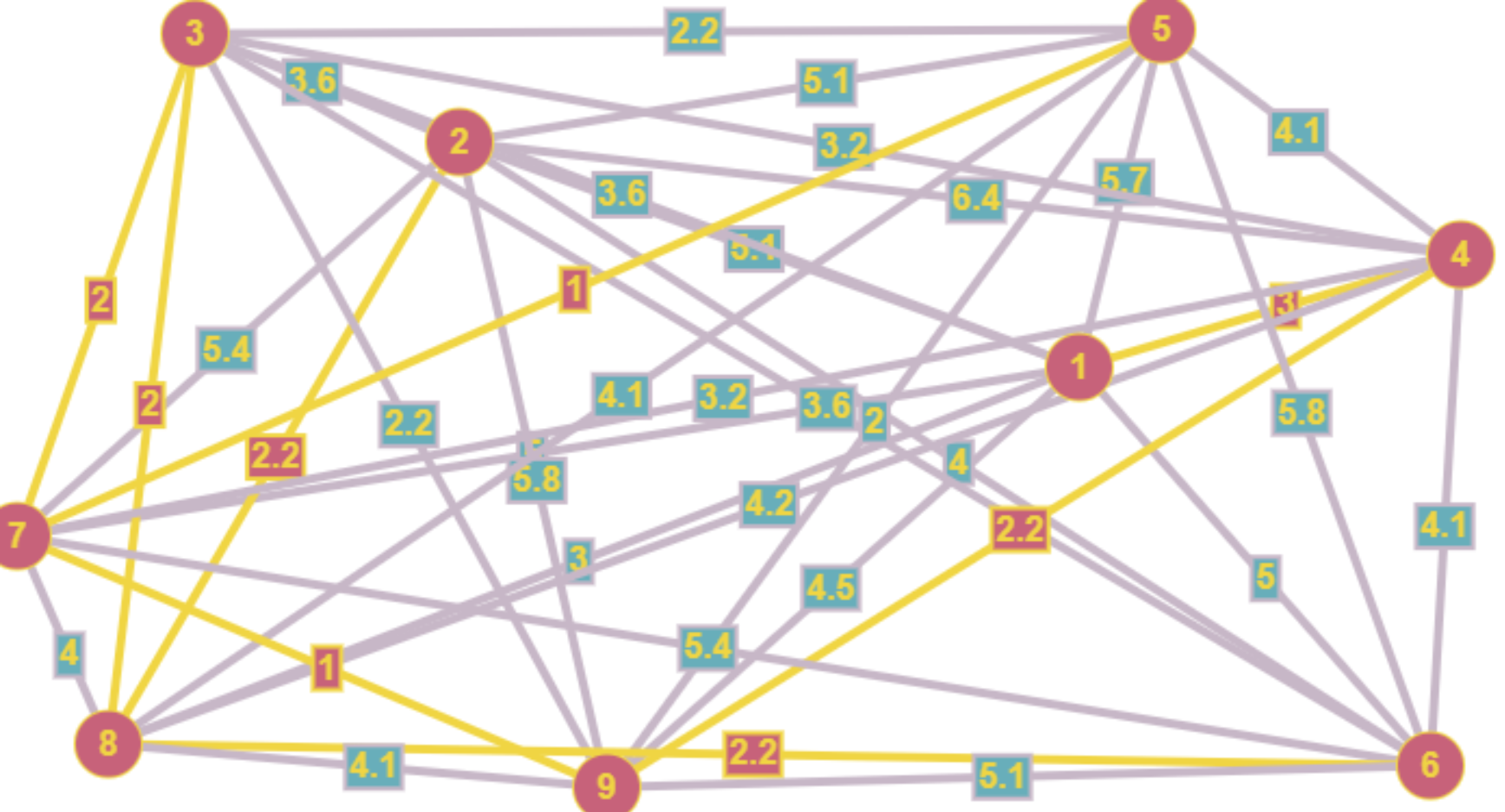
Also, since we don’t have the distances between two vertices, we will need to calculate the distance first.

1. Apply your algorithm to the sample input provided with the problem. Show all your work so the instructor is convinced that your algorithm works and that you know what you are doing.

So first, I calculated the distances between each vertex. Then I pin them in the graph below.



After that, I find the smallest weight edge, and from there I started to connect each vertex. Every time before I add edge between two vertices. I will check three things, is the weight of the edge min of the whole graph, is the vertex already connected, will I cause a closed graph by adding this edge. And if they all false. I will add the vertex.



**Solozol**

The independent nation of Solozol is perfectly flat. Unfortunately, Solozol has a very poor system of public highways. The Solozolian government is aware of this problem and has already constructed a number of highways connecting some of the most important towns. However, there are still some towns that you can't reach via a highway. It is necessary to build more highways so that it will be possible to drive between any pair of towns without leaving the highway system.

Solozolian towns are numbered from 1 to N and town i has a position given by the Cartesian coordinates (x i, y i). Each highway connects exactly two towns. All highways (both the original ones and the ones that are to be built) follow straight lines, and thus their length is equal to Cartesian distance between towns. All highways can be used in both directions.

Highways can freely cross each other, but a driver can only switch between highways at a town that is located at the end of both highways. The Solozolian government wants to minimize the cost of building new highways. However, they want to guarantee that every town is highway-reachable from every other town. Since Solozol is so flat, the cost of a highway is always proportional to its length. Thus, the least expensive highway system will be the one that minimizes the total highways length.

**Input**

The first line of the input consists of an integer indicating the number of test cases in the input. Then there's a blank line and the datasets separated by a blank line.

Each test case consists of two parts. The first part describes all towns in the country, and the second part describes all of the highways that have already been built.

The first line of the test case contains a single integer N (1 ≤ N ≤ 750), representing the number of towns. The next N lines each contain two integers, x i and y i separated by a space. These values give the coordinates of i Th town (for i from 1 to N). Coordinates will have an absolute value no greater than 10000. Every town has a unique location.

The next line contains a single integer M (0 ≤ M ≤ 1000), representing the number of existing highways. The next M lines each contain a pair of integers separated by a space. These two integers give a pair of town numbers which are already connected by a highway. Each pair of towns is connected by at most one highway.

**Output**

Write to the output file a single line for each new highway that should be built in order to connect all towns with minimal possible total length of new highways. Each highway should be presented by printing town numbers that this highway connects, separated by a space.

If no new highways need to be built (all towns are already connected), then the output file should contain a line with the sentence "No new highways needed". Print a blank line between test cases.

**Sample input**

1

9

1 5

0 0

3 2

4 5

5 1

0 4

5 2

1 2

5 3

3

1 3

9 7

1 2

**Sample output for the sample input**

1 6

3 7

4 9

5 7

8 3