CSCI 3101, Individual Project, Seowon Jung

Prim

1. My program works?

Yes, and it has not showed any error messages for all 12 test cases.

2. What I learned or difficulties I had with the program?

Actually, I've spent more than 100 hours for this Prim algorithm. To finish this, I've never gone to the outside and was doing this from morning to midnight on Saturday and Sunday. Nevertheless, I couldn't solve it for 3 weeks and I experienced frustration, because I'm not smart.

First time, I tried to trace nodes that need to be drawn (linked). However, this didn't work for every test case because, all test case has different situation. The most important thing was, I misunderstood Prim algorithm. I surfed on many websites about Prim, and therefore I understood what Prim algorithm works. After that, I started coding and I made it after only 3 hours.

It was very very fun, and helpful for me and my programming skill even though it was a time-consuming job. And besides, I drew and traced all test cases on graph papers which I made

3. Basic approach

I've done this with Python because, I think, Python is the best language for implementation of algorithms. It's very easy to use data structures, such as List, Set, and Dictionary. In my opinion, this project's key point is how to implement a checking cycle function.

My program gets input from the keyboard, a number of the test case, and then X and Y coordinates. All X and Y coordinates, and any other information for each node stores in class objects. Freckle Class creates object instances which has X, and Y coordinates, visited list, and distances between current node and all distances. For example, Node #1 has,

node[1].x = X coordinate

node[1].y = Y coordinate

node[1].visitedList = All node list that visited from this node.

And then, the program calculate all distances for the first node, and extract the shortest path from the distance list. After that, the program runs again with current node and current's closest node.

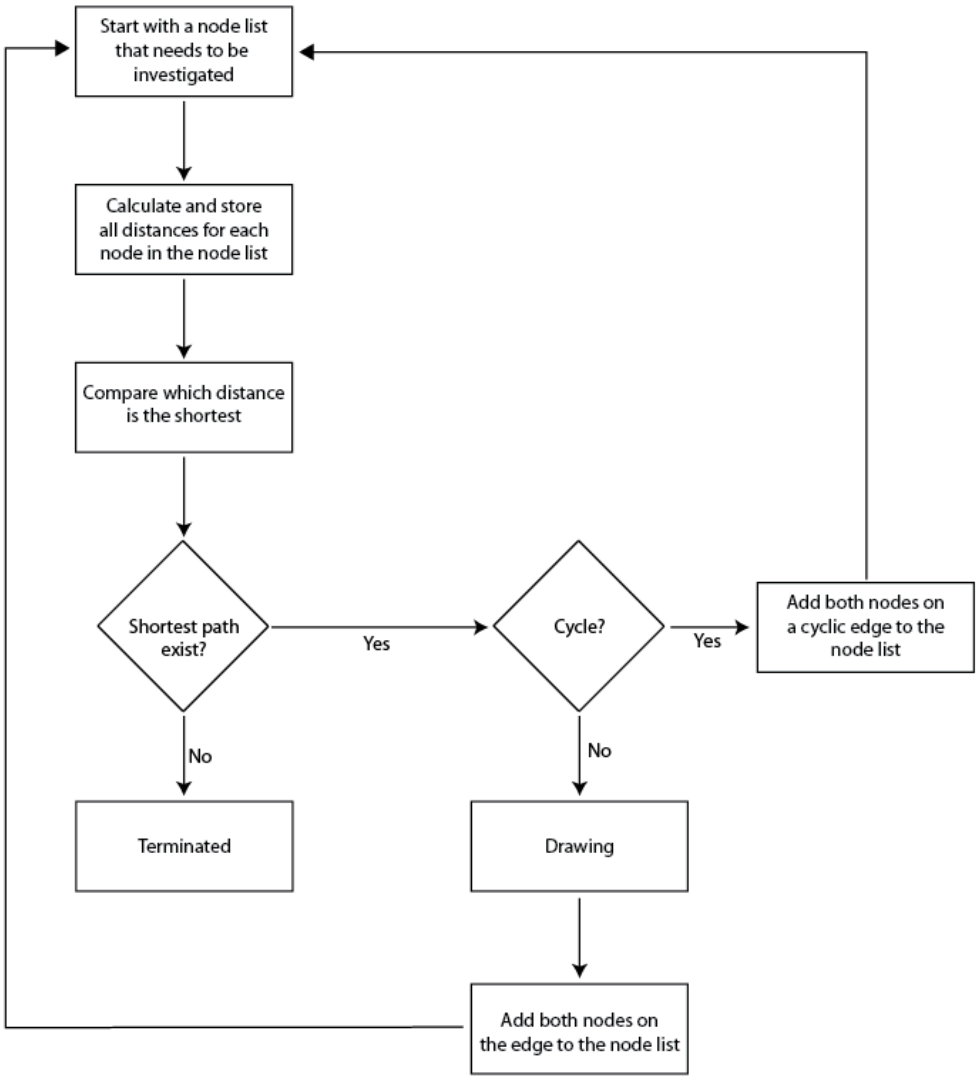
And then, the program calculate all distances for the first node and other nodes in the list in order to investigate a weight, and which edge is the shortest. After that, the program runs again with this shortest node and other nodes from before in order to compare the shortest path.

For checking cycle, I used a visited list that each node object has. If node[1].visitedList = [1, 3, 6], CycleTest function investigates all visited nodes for 1, 3, 6, and their all sub-nodes. If they have at least one common node in the visited list, this edge is cyclic.

4. Output for test cases

1) 3: 3.41
2) 100: 6658.39
3) 2: 0.41
4) 1: 0
5) 10:23.95
6) 1: 0
7) 9: 20.06
8) 9: 18.22
9) 2: 9.62
10) 6: 12.42
11) 1: 0
12) 4: 9.52

5. Work Flow =========>



CSCI 3101, Individual Project, Seowon Jung

Kruskal

1. My program works?

Yes, and it has not showed any error messages for all 12 test cases.

2. What I learned or difficulties I had with the program?

Kruskal was easier than Prim for me. I started coding Kruskal in the morning, and I could finish it in the evening on the same day. And also, it was so much fun, and helpful for me and my programming skill.

3. Basic approach

This program also uses Class object to store X and Y coordinates, and other information for each node.

My program get inputs X and Y coordinates from the keyboard, and then calculate shortest distances for all each node. And then, it finds out 2-way nodes, and remove them. For example, if the shortest path of node[3] is node[6], and vice versa, the shortest path of node[6] will be removed by ascending order. After this work, the program sorts the distance list by shortest order.
And then it draws shortest paths.

And then it calls the function named, findAnotherClosestNode to find out unlinked nodes. This function does same the above, calculating shortest distances for all each node, and remove 2-way nodes, and then sorting them. But, CycleTest function is added. Even if any shortest distance is cyclic, the function re-calculate all distances except this node. I used the same function with Prim's CycleTest.
However, Kruskal's running time was a little more longer because, the program re-calculates all shortest distances for all nodes when it reaches a cycle. It took 3 min 53 seconds for the test case with 100 nodes (Prim version took only 40 seconds. Intel Core i7 2GHz).

4. Output for test cases

1) 3: 3.41
2) 100: 6658.39
3) 2: 0.41
4) 1: 0
5) 10:23.95
6) 1: 0
7) 9: 20.06

8) 9: 18.22

9) 2: 9.62

10) 6: 12.42

11) 1: 0

12) 4: 9.52

5. Work Flow