CSCI203 ASSIGNMENT 3 Due 23:55 26/10/2018

Dijkstra's algorithm finds the shortest path from a given node to all other nodes.

1) We observe that we can modify this algorithm to stop as soon as a particular node is reached; thus producing an algorithm to find the shortest path between a specific pair of points. However, this algorithm may involve the consideration of a number of points which do not lie on the final shortest path.

We now consider 2 alternatives:

2) We can modify the algorithm to add nodes to the solution based on an A* criterion derived from the Euclidean (straight line) distance from each candidate node to the desired end node.
3) We can attempt to improve our efficiency by modifying Dijkstra's algorithm to start at both the source and destination nodes and to construct two partial solution trees in parallel until one node is in both partial solution trees.

Your task is to:

1. Code the modified Dijkstra's algorithm to search from the start node out.
2. Code the A* variant.
3. Code the proposed improved algorithm.

Input consists of the following data:

1) The number of nodes in the graph.
2) A set of triples containing the node number, its X-coordinate and its Y coordinate – one triple for each node in the graph.
3) The number of edges in the graph.
4) A set of triples consisting of two node numbers and a cost – one triple for each edge in the graph.
5) A pair of node numbers representing the start and end nodes between which a path must be found.

Output consists of the following data:

- The length of the shortest path from solution 1:
- The path (ordered list of nodes) from solution 1:
- The number of additional nodes in the solution tree for solution 1 (those not in the shortest path that were added to the selected set):
- The length of the shortest path from solution 2:
- The path (ordered list of nodes) from solution 2:
- The number of additional nodes in the solution tree for solution2 (those not in the shortest path that were added to the selected set):
- The length of the shortest path from solution 3:
- The path (ordered list of nodes) from solution 3:
- The number of additional nodes in the solution tree for solution 3 (those not in the shortest path that were added to the selected set).

Notes: The graph is undirected, so each edge connects the pair of nodes specified in both directions.
Do not use the STL.
The graph will not have more than 100 nodes.
Your program should print an appropriate error message if no path exists between the specified nodes.

Programs must compile and run under gcc (C programs), g++ (C++ programs), java or python.

Programs which do not compile and run will receive no marks. Programs should be appropriately documented with comments.

In addition to the source file ass3.c or ass3.cpp you should also submit a text file containing a brief discussion of your results. You should talk about the relative efficiency of each of the three proposed approaches and note any problems that may arise with each of them

submit -u *user* -c csci203 -a 3 ass3.c ass3.txt
or
submit -u user -c csci203 -a 3 ass3.cpp ass3.txt

Where your unix user name should appear instead of *user*.

The following image shows the graph, with node numbers, for which test data is provided.