Homework 5 Programming

Problem 1 - Dijkstra’s Algorithm

In this problem you will implement Dijkstra’s algorithm to find shortest paths between pairs of cities on a map. We are providing a GUI that lets you visualize the map and the path your algorithm finds between two cities.

Before you start:

Examine the Codio directory. You will find the file `cityxy.txt` which contains a list of cities and their (X,Y) coordinates on the map. These are the vertices in the graph. The file `citypairs.txt` lists direct, connections between pairs of cities. These links are bidirectional, if you can go from NewYork to Boston, you can go from Boston to NewYork. These are the edges of the graph.

Compile all Java sources and run the program `Display`. At this point, click on the “Box URL” button along the Codio tool bar at the top of your screen. This should bring up a view of the computer’s actual desktop screen. Inside, a window will have appeared with your running program that displays the map. You will notice that clicking on “Compute All Euclidean Distances” does nothing and that “Draw Dijkstra’s Path” will throw a null pointer exception on the terminal tab. You will have to implement these methods yourself.

You can switch between this Virtual Desktop tab and any of your other tabs freely. Make sure you quit the Java program before running it again. You can do this by either clicking the small x at the top of the java application in the virtual desktop window or by ctrl^c’ing the program from terminal.

Carefully read through the classes `Vertex.java` and `Edge.java`, which represent the vertices and edges of a graph. You will not have to modify these classes for the assignment, but you need to understand how the graph is represented and what information is associated with each vertex and edge.
You will only have to modify `Dijkstra.java`, which represents a graph and will contain your implementation of Dijkstra’s algorithm. You will need to use the instance variable `vertexNames`, which contains a mapping from city names to `Vertex` objects after the graph is read from the data files. The main method of Dijkstra illustrates how the class is used by the GUI and might be useful for testing your implementation on the command line.

- Implement the method `computeAllEuclideanDistances()` which should compute the euclidean distance between all cities that have a direct link and set the weights for the corresponding edges in the graph. Once this works correctly, the correct distances should be displayed in the GUI when clicking on "Compute All Euclidean Distances".
- In the method `doDijkstra(String s)`, implement Dijkstra’s algorithm starting at the city with name s. Use the distances associated with the edges. The method should update the `distance` and `prev` instance variables of the `Vertex` objects. You should not use a priority queue to store vertices that still need to be visited. Instead, keep these vertices on a List and scan through the entire list to find the minimum. We are making this simplification (at the expense of runtime) because `java.util.PriorityQueue` does not support the `decreaseKey` operation.
- Implement the method `getDijkstraPath(String s, String t)`, which uses the `distance` and `prev` instance variables of the Vertex objects to find the shortest path between `s` and `t`. This method should be called only after `doDijkstra(String s)` has been called with the start city. The resulting path should be returned as a list of `Edge` objects. Once this works correctly, you should be able to compute and display paths in the GUI.

Note: Your program must work for repeated runs of dijkstra’s algorithms on different city pairs. To make sure that your program does this, be sure that you reinitialize the bookkeeping fields (`known`, `prev`, and `distance`) in the vertices stored in `vertexNames` at the beginning of each run of `doDijkstra`.