Assignment 4

CSI2110

Brian Makos - 300194563

Method 1: BFS

The method BFS I wrote takes in a HashMap adjacencyMatrix (it's the edges parameter taken in computeClosenessCentrality(Graph graph), List of integers nodes, an array on integers of the previous nodes, and an integer distanceFromStartNode that is just the distance required to hit the node desired from the node we started at.

The goal of this method is to perform a Breadth-first search on the given graph and return the minimum distance required to reach each node from a specific starting node. The starting node is then swapped, and each instance is stored in matrix list.

It contains a quadruple nested loop which is terrible for the run time, but it allows me to create and instance where we can start at every node, loop through every case of an end node. I also created a list of Booleans named visited nodes, that will notify the program when a node has already been visited. There is also a list that notifies the program what nodes it still needs to visit (nodes get added as the program runs through every case). At then end of the code, all the shortest distances are added to a list of lists named allDistancesFromStartNode. That information is then passed back to ComputeClosenessCentrality.

Method 2: CC

This method computes the CC of the minimum distances of each node pulled in from computeClosenessCentrality. It uses the equation given in the assignment and then returns a hashmap ccHasMap that contains a integer key and a double value for each key. The integers in this case are the given nodes and the doubles are the cc values this method computes.

This method takes in an List of Integer lists and a list of the nodes. Inside I instantiate the ccHashMap where I store the values. I then fill that HashMap to simply the replacement of the values later on. I then create some temp values for the N-1 part of the equation. After that, I loop through the List of Integer list and sum up all the values of the shortest distances. I then do the temp value divided by the sum of all the shortest distances to find the cc value for that specific node. I then do the same for each other node and store it into the ccHashMap which I return to computeClosenessCentrality.

Method 3: computeClosenessCentrality

In this method I run the other methods and return the final answer HashMap. This method takes in the given graph given by the professor. I then get the values of edges and nodes and set them to a value. I instantiate some of the variables used in the other methods and finally after running it through all my other methods, this method returns the final HashMap desired by the professor.

Run time:

Because of my lack of time with my exams coming up and my part time job, I couldn't fully complete this assignment. Therefore, the runtime for my code can't be determined properly. However, with a quadruple nested loop I can assume that the run time wasn't very efficient and rather slow.

Sorry for the incomplete assignment.