Name: Vidulash Rajaratnam **Number:** 8190398

Assignment 4 Report

Brief Description

Classes:

- **Graph:** This class was kept unchanged. It is a class used to create directed graph objects that each contain a list of integers called ***nodes***, representing all the nodes in the graph, and a map of an integer to a list of integers called ***edges,*** linking nodes to their adjacent nodes. It has two methods, where one returns the list of integer nodes, called ***getGraphNodes()*** and one to the map edges called ***getGraphEdges().***

- **CSI2110:** This is the main class in this assignment that will create the a graph object using the given email-dnc.edges file and print the top 10 most influential nodes in the graph.

    - **Methods:**

-

    - **readGraph(edgesFilename: String): Graph**

      This method was kept unchanged. It is responsible for converting the edges file into a readable file. It is also responsible for converting this data into a Graph object by getting data from each line in the file. Each line is split into an array of strings by the symbol ",". The first string in the array is added into the Graph object's nodes integer list while the second string in the array is added into both the edges integer list and the nodes integer list, indicating they are adjacent. Once every line has been analyzed, it returns the Graph object.

    - **getOutgoingLinks(A: int, g: Graph): returns ArrayList<Integer>**

      This method was added and is responsible for returning the arraylist of integers (nodes) that are outgoing from the node A in the graph g. It will use the given Graph object and get its nodes and edges. From that it will find the node A, get the corresponding list of integer in edges, and add all nodes to an arraylist. It then returns this arraylist.

    - **getTiToA(A: int,g:Graph): returns ArrayList<Integer>**

      This method was added and is responsible for returning the arraylist of integers (nodes) that lead to the node A in the graph g. It will use the given Graph object and get its nodes and edges. It iterates through the list of nodes and adds it to an arraylist if it contains the target node in its corresponding list of integers in edges. It returns this arraylist.

    - **makeList(g:Graph, value:double): returns ArrayList<Pair<Integer,Double>>**

      This method was added and is responsible for making an arraylist of pairs. For each node found in the graph, it creates a corresponding pair that has the node (integer) and an initial PR value(double) given by value. It then adds this pair to the arraylist and returns this arraylist.

    - **retrieve(key: Integer, a: ArrayList<Pair<Integer,Double>>): returns int**

      This method was added and is a method that will return the index value, if the key value is equal to a Pair's node value. If a match is found, it will return the index value and otherwise it will return -1 indicating it was not found.

- **getPRforNode(a: ArrayList<Pair<Integer,Double>>, g: Graph, count: int): returns ArrayList<Pair<Integer,Double>>**

This method was added and is responsible for updating the PR value of a node in a arraylist of pairs after calculating its new value. It takes the initial arraylist a, and gets the pair at index count. Once it finds the pair, it checks the number of nodes that lead to this node (getTiToA). If that is empty, the new PR value is set as 1-d where d is the damping factor of 0.85, and updates the pair's PR value. If it is not empty, it gets number of outgoinglinks and the PR value of those nodes to calculate the new PR value through the equation given in the assignment, and updates the PR value of the pair. It returns the updated arraylist once complete.

- **PRiteration(cons: ArrayList<Pair<Integer,Double>>, g: Graph, count: int, size: int, upd: ArrayList<Pair<Integer,Double>>): returns ArrayList<Pair<Integer,Double>>**

This method was added and is a recursive method that iterates through the arraylist of pairs and updates all PR values using PR values from a secondary list. Count value is incremented by 1 every iteration and if it reaches the size value of the number of nodes, it will stop updating the arraylist and return it. Otherwise it will update the PR value, using the method getPRforNode, for the pair at index count in the upd arraylist using PRvalues from the cons arraylist, increments count and enters itself with added count value and updated upd arraylist.

- **getTopTen(a: ArrayList<Pair<Integer,Double>>): returns ArrayList<Pair<Integer,Double>>**

This method was added and returns an arraylist of pairs of the nodes with the highest PR values from another arraylist of pairs. It iterates through every pair and finds the highest values till the new arraylist reaches a size of 10. Once it is full, it will return this new arraylist. It will return the same arraylist if the given arraylist was 10 or less in size.

How to Obtain Solution?

To get the top 10 most influential nodes in the Graph, I used a combination of the methods described above found in the CSI2110 class above. Here is the order in which it was determined:

1. Use the ***makelist*** method to make TWO arraylists of pairs with a **PR value of 1.0**. One that is constantly updated called updatedPR and one that is used as a reference called initialPR.
    a. The initial arraylists with PR values set at 1.0 is printed.
2. Use a **for loop**, to iterate the function ***PRiteration***, a method that will update the PR values in updatedPR, and then set the initialPR equal to the updatedPR. In every iteration, the initialPR is set as the updatedPR while the updatedPR is left unchanged.
    a. This for loop was iterated **10 times** for more precise PR values.
    b. The final updatedPR array list is printed.
3. Once the arraylist updatedPR is updated 10 times, the method ***getTopTen*** is used to make a new arraylist of pairs with the highest 10 PR valued nodes and printed in order using a for loop

RESULTS: **Computation time ~ 8 seconds**

1. Node 1669, PR = 51.98603189202687
2. Node 1874, PR =42.3643827338342
3. Node 453, PR =35.01149980621457
4. Node 1287, PR =18.774318850467242
5. Node 1258, PR =14.825454681729813
6. Node 895, PR =13.198856123232307
7. Node 1706, PR =11.938225425625978
8. Node 56, PR =8.866690516360963
9. Node 999, PR =8.702585924082937
10. Node 1144, PR =8.604447456284564