AA. Final Project – Social Network Edgelist II

This project will extend the midterm project to further analyze the Twitter data. For this application *responsiveness* is very important, so runtime should be minimized even at the expense of increased memory usage. This is a common situation in real world applications.

Specifically, you need to create two new methods:

```
Collection<TwitterUser> getFollowing(TwitterUser user)
TwitterUser getByPopularity(int x)
```

In the A. Midterm Project – Social Network Edgelist I, you read in a data file (social_network.edgelist) that indicated whom a given user is following. The getFollowing method should do the reverse -- it should return a collection of the people who are following a given user being followed. The object that you return can be any type of Java Collection that we have studied: list, queue, set, map, etc. Choose whichever is most appropriate. The getByPopularity method should return the xth TwitterUser (starting from zero) when the users have been sorted by the following criteria:

1. Number of users who are following a given user being followed (largest to smallest)
2. If two different users have the same number of users who are following the given user being followed, sort by the number of people the user being followed is following (largest to smallest)
3. If two different users have the same number of users who are following a given user being followed, and two different users have the same number of people the user being followed is following, sort by user id (smallest to largest) of the user being followed. Note that since user id is unique, these criteria impose a total rather than partial ordering on the TwitterUser objects (in other words, there will never be any "ties")

The time complexity of both of these methods must be constant, i.e. O(1). In order to achieve this, you will likely need to create additional data structures in the TwitterUser class, the driver program, or both. Keep in mind that it is ok if reading in the data files and initializing all of the TwitterUser objects is slow (within reason). It is also fine if these objects take up a lot of space in memory, as long as the program can execute on a standard desktop computer. For each method, write a comment (in your code, right before the method implementations) that describes the changes you have made to your project since the A. Midterm Project – Social Network Edgelist I in order to implement the method. Additionally, modify your driver program to test the operation of the two

Copyright © 2017, 2018 Sinclair Community College. All Rights Reserved.

new methods.

You will be graded according to the following requirements:

- The getFollowing method returns the correct results
- The getFollowing method has a constant time complexity
- The comment explaining the getFollowing method is complete and clearly written
- The getByPopularity method returns the correct results
- The getByPopularity method has a constant time complexity
- The comment explaining the getByPopularity method is complete and clearly written
- The driver program tests the execution of both methods
- The program compiles
- The program runs
- The program is clearly written and follows standard coding conventions
- ***Note:*** *If your program does not compile, you will receive a score of 0 on the entire assignment*
- ***Note:*** *If you program compiles but does not run, you will receive a score of 0 on the entire assignment*
- ***Note:*** *If your Eclipse project is not exported and uploaded to the eLearn drop box correctly, you will receive a score of 0 on the entire assignment*
- ***Note:*** *If you do not submit code that solves the problem for this particular project, you will not receive any points for the program's compiling, the program's running, or following standard coding conventions.*

Copyright © 2017, 2018 Sinclair Community College. All Rights Reserved.