Project name: Mining Social Network

In this project, we would analyze social network data. The opportunities for learning are practically endless in a social network. Who are the “influential” members of the network? What are the sub-communities in the network? Who is connected to whom, and by how many links? To answer these questions, you would be asked to analyze a real-world social network data set by implementing a couple of graph algorithms.

About the data

The data set we use contains the Facebook friendships between 2000 students at an university and is represented by a graph. Each number represent a student node in the graph. Each pair of numbers is a directed and unweighted edge representing the friendships between two students. The file name is facebook_2000.txt, and the edges in this file are directed; however, each edge also appears in reverse order, making the final result undirected. The graph in this file is not connected. Another file named facebook_1000.txt represents a connected graph with 1000 students’ friendship. If you have an algorithm with a large runtime, it may be helpful to use file TinySample.txt with much smaller graph size instead of the larger ones. All of the files can be found in the starter code’s data folder.

About the starter code

In the starter code, the Graph.java implements an adjacency list graph representation. Each of its API is described in the comments, and can be used in your graph processing algorithms. Also, please feel free to add your own APIs if needed. The GraphLoader.java is provided to read the file and construct the graph accordingly. The GraphTester.java is used to test all the algorithms you implement by printing out the analysis results. Your output should be the same as the expected results shown in the end of this instruction.

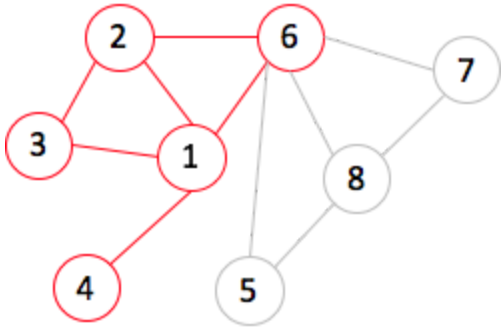
All other files including Ego.java, Center.java, CC.java, Influence.java and Diameter.java are the skeleton codes where you can apply your code to implement specific APIs which would be called in the GraphTester.java to analyze the social network data from differents aspect. Each class is used by clients to answer one question about the data.

If you are using eclipse, you can import the project into eclipse: Import the starter files: File -> Import -> Select "Existing Projects into Workspace" -> Next -> Browse and set root directory to folder contents of zip were extracted to -> Finish. Feel free to use another IDE or manually compile and run your programs. If you need help, Google is your friend.

Questions to answer:

1. How powerful your personal network is?

To answer this question, we first need to build the personal network. Ego network is used to represent the personal network, and it consists of a focal node ("ego") and the nodes to whom ego is directly connected to (these are called "alters") plus the ties, if any, among the alters. Of course, each alter in an ego network has his/her own ego network, and all ego networks interlock to form the human social network. For example, in the following graph, the red subgraph is the ego network of node1. The strength of the ego network depends on both its size (e.g, the number of nodes) and its density (e.g, the number of edges / the number of all possible pair). To simplify it, we use the number of edges in an ego network to measure the power of your personal network.

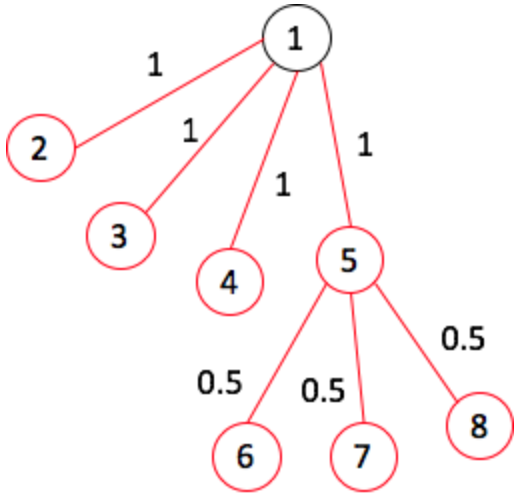


Your task is to implement the Ego class in Ego.java file to return the more k powerful personal networks in the facebook data set by writing the following APIs.

- **public** Ego(Graph g): this is the constructor to build the data structure that stores all ego networks from graph g in a sorted order
- **public** List<egonet> top(**int** k): this is the method to return the top k ego networks with the largest number of edges. The nested class egonet is provided, which contains the egonet graph and its center.

2. How influential you are in the whole network?

Your influence or impact on the network depends on how many people you can reach and how far for you to connect those people. For example, marketers would get their message spread by influencers who have a large and shallow following to maximize the reach. To quantify your impact, we calculate the reachability by adding up your influence on each reachable people, which is 1/(2^(distance-1). For example, in the following graph, node 1's influence is 5.5.



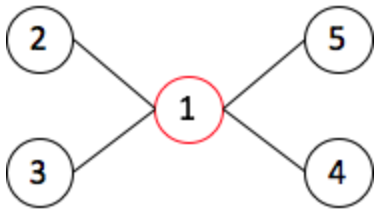
Your task is to implement the Influence class in Influence.java file to return the more k influential personal in the facebook data set by writing the following APIs.

- **public** Influence(Graph g): this is the constructor to build the data structure that stores all influencers from graph g in a sorted order
- **public** List<influencer> top(**int** k): this is the method to return the top k influencers. The nested class influencer is provided, which contains the id of the influencer and its power calculated by its reachability.

3. How important you are within the network?

One main factor for determining the most important members of a network is an individual's location between different sections of a network, which is called betweenness centrality. In graph theory, betweenness centrality is based on shortest paths. The betweenness centrality for each vertex is the number of shortest paths that pass through the vertex. Calculating the betweenness centralities of all the vertices in a

graph involves calculating the shortest paths between all pairs of vertices on a graph. For example, in the following graph, the node 1 exists in the flowing 20 shortest paths: 1->2, 1->3, 1->4, 1->5, 2->1, 2->3, 2->1->4, 2->1->5, 3->1, 3->1->2, 3->1->4, 3->1->5, 4->1, 4->1->2, 4->1->3, 4->1->5, 5->1, 5->1->2, 5->1->3, 5->1->4. This node is considered as the most important node in the graph with the highest betweenness centrality.



Vertices with higher betweenness is more important within a network by virtue of their control over information passing between others. They are also the ones whose removal from the network will most disrupt communications between other vertices because they lie on the largest number of paths taken by messages. The closer a person is to the center of the graph, the more visibility they have across multiple verticals.

Your task is to implement the Center class in Center.java file to return the more k important person in the facebook data set by writing the following APIs.

- **public** Center(Graph g): this is the constructor to build the data structure that stores all students and their betweenness centralities from graph g in a sorted order
- **public** List top(**int** k): this is the method to return the most important k person. The nested class center is provided, which contains the id of the person and its betweenness centrality.

Bonus questions:

By successfully answering the following bonus questions, you can get up to 10% bonus credits towards your final grade.

4. How many communities in the network?

The simplest way to detect communities in a network represented as a graph is to find all connected components, which is a maximal set of vertices such that every vertex is reachable from every other vertex. Based on this idea, we can define a community as a

group of elements where each element can communicate with other elements: this reflects the fact that there is always a bidirectional flow that connects two elements of a community.

Your task is to implement the CC class in CC.java file to return the biggest k communities in the facebook data set by writing the following APIs.

- **public** CC(Graph g): this is the constructor to build the data structure that stores all connected components that represent the communities from graph g in a sorted order. Here, we only need to store the id and the size of each connected component.
- **public int** count(): this is the method to return the number of connected component.
- **public** List<cc> top(**int** k): this is the method to return the biggest k communities. The nested class cc is provided, which contains the id of the community and its size.

5. How many links you need to connect to another person in the network?

To discuss this question, we first look at the six degrees of separation theory, which is the idea that all living things and everything else in the world are six or fewer steps away from each other so that a chain of "a friend of a friend" statements can be made to connect any two people in a maximum of six steps. It is generalized to the average social distance being logarithmic in the size of the population. Is this theory true if we look at our facebook data set? To find it out, we need to calculate the graph diameter, which is the longest shortest path among all pairs of vertices in the graph. The graph diameter tells the most numbers of links you need to connect to another person in the social network.

Your task is to implement the Diameter class in Diameter.java file to return the graph diameter in the facebook data set by writing the following APIs.

- **public** Diameter(Graph g): this is the constructor to compute the graph diameter. Here, we assume the graph g is connected. To assure that, we load facebook_1000.txt to construct the graph g for testing.
- **public int** getDiameter(): this is the method to return the graph diameter.

The expected output

According to the test code GraphTester.java in the starter code, the following output is expected.

```
======== Top 5 ego networks ========
[0] center: 1957 strength: 350
[1] center: 1360 strength: 308
[2] center:  644 strength: 288
[3] center:  546 strength: 256
[4] center:  371 strength: 252
======== Top 5 Influencers ========
[0] source: 1382 influence: 541.03
[1] source:  908 influence: 517.44
[2] source: 1617 influence: 513.84
[3] source:  758 influence: 509.19
[4] source:  842 influence: 508.72
======== Top 5 Centers ========
[0] Id: 1382 centrality: 104077
[1] Id:  177 centrality: 69842
[2] Id: 1506 centrality: 68806
[3] Id:  908 centrality: 64948
[4] Id: 1998 centrality: 61580
You used 8256.18 ms
Bonus project:
======== Top 5 Connected Component ========
Total number of CC: 13
[0] Id:  1 Size: 1755
[1] Id:  2 Size: 3
[2] Id:  3 Size: 2
[3] Id:  4 Size: 2
[4] Id:  5 Size: 2
======== Social Diameter ========
Largest Diameter: 12
```

More bonus

You may notice that the execution time of task 1,2 and 3 are measured in the GraphTester.java. If your execution time is ranked at top 5 in the class, you can get extra 3% bonus credits towards your final grade. So feel free to choose any data structures you want, and pick the best one for optimal performance if you want this extra bonus credits. All codes would be run at my personal laptop for fairness.

Project submission

Finally, please submit your completed work including all java files and a screenshot of your output to Blackboard in a zip file.