Data Mining in the Social Network

Goals:

- Learn how to solve real-world problems with graph theory.
- Understand how to convert large data sets to graph representations.
- Practice with multiple classic graph algorithms.

This project can be done with a single partner.

Tasks:

The opportunities for learning are practically endless in a social network. In this project, you are asked to analyze a real-world social network data set to answer the following questions by implementing a couple of graph algorithms:

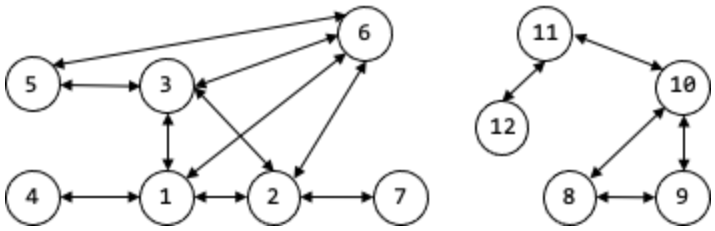
- ☐ How powerful is one's personal network?
- ☐ How influential is one in the whole network?
- ☐ How many communities are in the network?

Data Sets:

- `facebook_2000.txt` contains a graph representation of the Facebook friendships among 2000 students.
- `TinySample.txt` contains a graph representation of the Facebook friendships among 16 students. This is mainly used for quick testing.

In those files, each number represents a student node in the graph. In each line, a pair of numbers represents a *`directed`* and *`unweighted`* edge starting from the first number and going to the second number. Each edge represents the friendship between two students. This friendship is mutual, meaning there are two edges between two numbers in opposite directions. Note that the nodes in this graph are not all connected.

The following is the graph represented by TinySample.txt. Here, two edges between two nodes are merged into one with two arrows for simplicity.



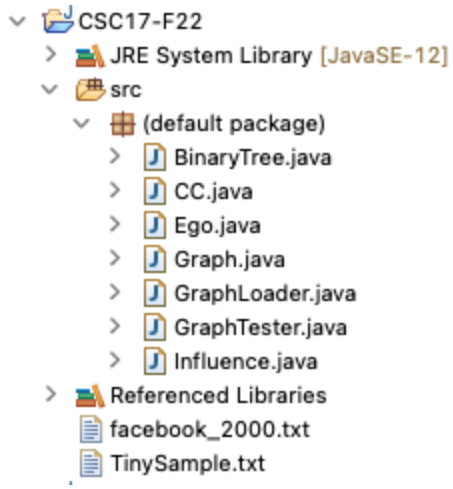
Starter Code:

Besides the data sets, several java files are provided to set up the graph, the to-be-implemented methods, and the testing. Here is a brief explanation. More details can be found in the comments.

- ★ `Graph.java` implements an adjacency list graph representation that provides several functions (e.g., adding an edge, returning a neighbor list) that can be used by your graph processing algorithms. Feel free to add your own if needed.
- ★ `GraphLoader.java` initializes a graph by reading the data set file.
- ★ `CC.java, Ego.java, and Influence.java` are the skeleton codes where you can implement graph processing algorithms that would be called in the GraphTester.java to analyze the social network data.
- ★ `GraphTester.java` prints the analysis results generated by the algorithms you implement. The output should be the same as the following results.

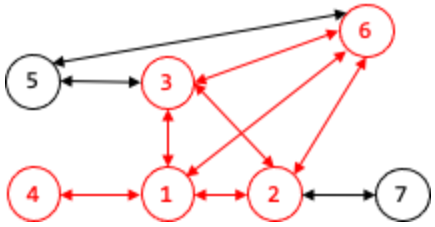
```
======== Top 5 ego networks ========
[0] center: 1957 strength: 350
[1] center: 1360 strength: 308
[2] center:  644 strength: 288
[3] center:  546 strength: 256
[4] center:  371 strength: 252
======== Top 5 Influencers ========
[0] source: 1382 influence: 541.03
[1] source:  908 influence: 517.44
[2] source: 1617 influence: 513.84
[3] source:  758 influence: 509.19
[4] source:  842 influence: 508.72
======== Top 5 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
```

You can drag-and-drop files in the starter code folder to the eclipse java project. The layout should look like this:



Question 1. How powerful is one's personal network?

The personal network, also called the ego network, consists of a focal node ("ego") and the nodes to whom the ego is directly connected (these are called "alters") plus the ties, if any, among the alters. The power of an ego network is measured by the number of edges in that ego network. For example, in the graph below, the red subgraph is the ego network of node 1, and its power is 14.

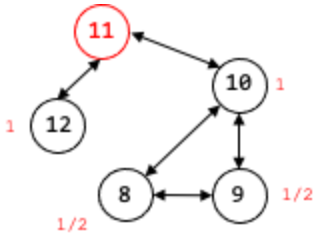


Your task is to implement the Ego class in the Ego.java file to return the k most powerful personal networks in the social network graph by implementing the following methods.

- `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 based on their powers.
- `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 to define an ego network.

Question 2. How influential is one in the whole network?

One's influence on the network depends on how many people one can reach and how far one connects with those people. For example, marketers would get their message spread by influencers who have a large and shallow follower network to maximize their reach. One's influence is calculated by adding up one's influence on each reachable person, which is 1/(2^(distance-1) where the distance is the number of links between two people. For example, in the following graph, node 11's influence is 3 since its influence on nodes 10, and 12 is 1, and its influence on nodes 8, and 9 is 1/2.

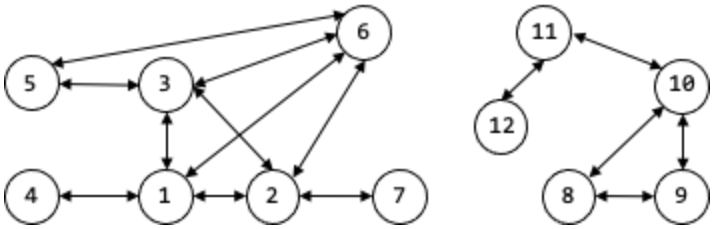


Your task is to implement the Influence class in the Influence.java file to return the k most influential person in the social network graph by implementing the following methods.

- `public Influence(Graph g)`: this is the constructor to build the data structure that stores all influencers from graph g in a sorted order based on their influences.
- `public List<influencer> top(int k)`: this is the method to return the top k influencers. The nested class influencer is provided to define an influencer.

Question 3. How many communities are 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. For example, there are two communities in the following graph, and their sizes are 7 and 5.



Your task is to implement the CC class in the CC.java file to return the k biggest communities in the social network graph by implementing the following methods.

- `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 based on the size (i.e., the number of nodes). 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 components.
- `public List<cc> top(int k)`: this is the method to return the k biggest communities., the nested class cc is provided to represent a community, which contains the id of the community and its size.

Submission:

Please submit your completed CC.java, Ego.java, and Influence.java and a screenshot of your output to Blackboard. You won't receive full credits if you fail to submit the screenshot.