Lab 2: Social Network

Professor: Ronaldo Menezes

TA: Ivan Bogun

Department of Computer Science
Florida Institute of Technology

August 27, 2014

1 Problem statement

Assume you are working on a social network. You decide to add a nice feature called "invite to a party" which would simplify the process of inviting people to an event (party in this case). It should do the following: given a friend who organizes the party it should send invites to all it's friends, friends of friends, friends of friends of friends and so on. In this way only people for which there is a link of friends would receive an invitation.

1.1 Communities

Assume that if same people could be invited to the same party they are part of the same community. Relation "being in the same community" is an equivalence relation, thus it partitions social network into disjoint sets (communities). Implement a function which would partition all people in the network into their respective communities.

2 Implementation

Implement the class *SocialNetwork.java*[1] which should implement the interface *Network.java* .

```
// SocialNetwork.java
import java.util.ArrayList;

public class SocialNetwork implements Network{

        //maximum number of people in the network
        int n;

        // names of the people in the network
        ArrayList<String> names;

        // constructor
        public SocialNetwork(int n_) {
        }

        // Add a friendship between personAname - personBname into the network
        public void addPeople(String personAname, String personBname) {
        }

        // process multiple friendship requests from a multiline string where
        // each line is in the form: personA - personB
        // Hint: this function should be using addPeople()
        public void processConnections(String multiStringWithConnections) {
        }
```

[1] You can copy code for this file from: SocialNetwork.java

```
        // given a name return an array of people who should be invited to the
            party
        public String[] inviteToParty(String name) {
        }

        // return true if there is a link of friends between personAname and
            personBname
        public boolean areConnected(String personAname, String personBname) {
        }

        // print return a string containing communities of the network
        public String printCommunities() {
        }

}
```

Interface *Network.java* [2] is given below:

```
public interface Network {

        // add a connection between two people in the network. Note that it is
        // possible that connection of the person's name you are to process is
            not
        // present in the arraylist 'names'
        void addPeople(String personAname,String personBname);

        //process a multiline string containing multiple personA - personB
            connections
        void processConnections(String multiStringWithConnections);


        // return true if there is a link of friends between two people
        boolean areConnected(String personAname,String personBname);


        // main function which, given a name, will return array of people which
            are
        // connected to it via link of its friends
        String[] inviteToParty(String name);


}
```

[2] Network.java

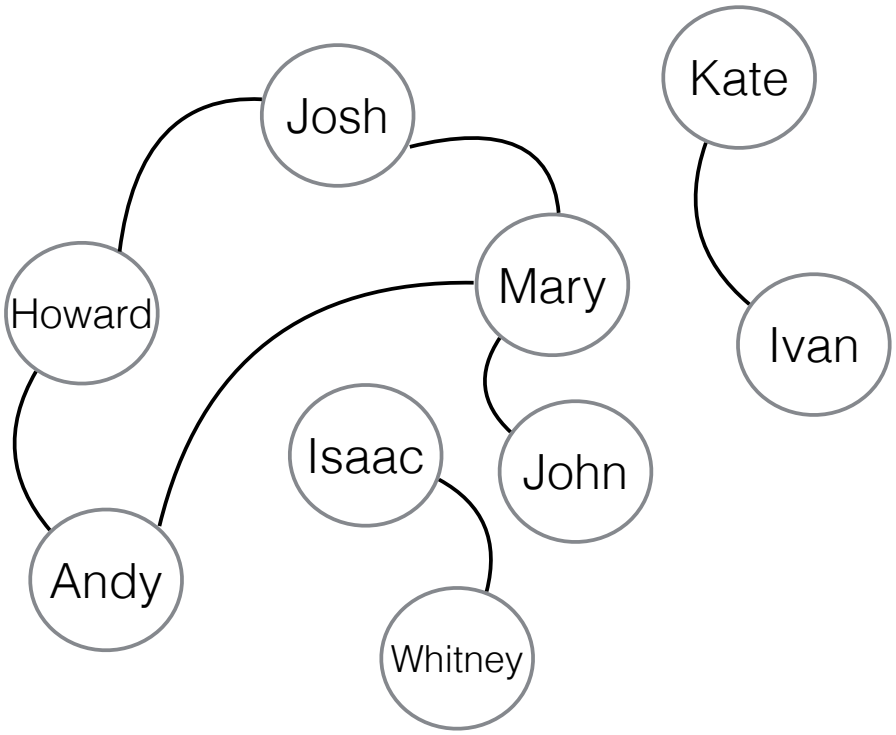

Figure 1: Visualization of the initial network used in the main (after application of *processConnections()* function)

3 Sample input-output

Create the file *Driver.java* [3] whose modified version will be used for testing.

3.1 Input

```
public class Driver {

        public static void main(String[] args) {

                int n=30;
                SocialNetwork social = new SocialNetwork(n);

                String connections="Mary - John\n"
                                + "Josh - Mary\n"
```

[3] Driver.java

```
                                + "Kate - Ivan\n"
                                + "Isaac - Whitney\n"
                                + "Josh - Howard\n"
                                + "Andy - Howard\n"
                                + "Andy - Mary";

                // print out connection list
                System.out.println(connections+"\n");

                // process friendship requests
                social.processConnections(connections);

                System.out.println(social.printCommunities());

                String[] friends=social.inviteToParty("Mary");

                System.out.print("Friends to be invited to the party: ");
                for (int i = 0; i < friends.length; i++) {
                        System.out.print(friends[i]+" ");
                }
                System.out.println("\n");

                // add some more connections. Notice that we add people which
                    haven't been seen previously.
                social.addPeople("Steven", "Jim");
                social.addPeople("Ivan", "Isaac");

                System.out.println(social.printCommunities());

                social.addPeople("Mary", "Kate");

                System.out.println(social.printCommunities());
        }
}
```

3.2 Output

```
Mary - John
Josh - Mary
Kate - Ivan
Isaac - Whitney
Josh - Howard
Andy - Howard
Andy - Mary

Community #1 Mary John Josh Howard Andy
Community #2 Kate Ivan
```

```
Community #3 Isaac Whitney

Friends to be invited to the party: John Josh Howard Andy

Community #1 Mary John Josh Howard Andy
Community #2 Kate Ivan Isaac Whitney
Community #3 Steven Jim

Community #1 Mary John Josh Kate Ivan Isaac Whitney Howard Andy
Community #2 Steven Jim
```

4 Grade breakdown

basis	grade
Implementation	(60)
intiveToParty()	25
printCommunities()	25
other	10
Comments	(20)
General	20
Overall	(20)
Compiled	5
Style	5
Runtime	10
Total	100