PROJECT 2:

Recommend Friends in Online Social Networks

Introduction

Online social networks (OSN), such as Facebook, Twitter and Google+, have gained enormous popularity in the past few years. Billions of users interact through online activities such as posting statuses and pictures, sharing interests and thoughts. Further, these online social networks can help you discover new friends from your current friends. In this project, you will have an opportunity to find new potential friends for users by analyzing a social network graph on Facebook.

In this assignment, you will build a graph of Facebook users. Each user is a vertex of the graph. There is an edge between two users if they are friends on Facebook. Through analyzing this graph, your program can recommend potential friends to users, and find a chain of friends that can introduce you to a person that you want to connect as a friend.

Task Overview

In this assignment you need to accomplish the following specific tasks.

1. Read the Facebook data from file “facebook_network.txt” and build a Facebook user connectivity graph.
2. The friends of your friends are good resources to discover new friends. Recommend new friends for a given user by selecting the users who are friends of your friends. In other words, you are recommending users who have a mutual friend for the given user.
3. You may have hundreds of people in your friends of friend list. Rather than recommending them all, it is better to rank them. People who have more mutual friends are more likely to be friends. Recommend the top 10 users who have the most mutual friends with a given user.
4. You might want to know a certain person but you don’t know who can introduce you to the person. With the help of the online social network, you can find a chain of users to help you get to know that person. The less number of users in the chain, the better.

Recommend a chain of users between two given users. You can modify the breath first search algorithm to achieve this goal.

Getting Started

The best way to complete any programming assignment is to take tasks step-by-step. The first decision you need to make is the number of classes you need. For this assignment, you should have at least three classes, including the *Graph* class and *Vertex* class.

The Facebook user information you need is generated by the Stanford Large Network Dataset Collection at *http://snap.stanford.edu/data/.* Because of the user privacy issue of Facebook, the user data is anonymous. Here we randomly set a common user name for each user ID. Each line in the file contains a friendship relation of two users. For example, a line contains "David, Bob" means they are friends on Facebook. This relation represents an edge in our Facebook user connectivity graph. The input file is given by the file, facebook_network.txt. For reading and parsing the "facebook_network.txt" file, you can refer to the codes in Project 1.

Adjacency matrix and adjacency list are two ways to represent the friendship connection between vertices in our graph. But in this task, you are required to use adjacency list as the data structure for the graph. The social network graph is usually a sparse graph, which means the number of adjacent edges for each vertex is much smaller than the total number of vertices. The adjacency list will be space efficient.

Required Output

We provide 5 pairs of Facebook users in a file named "find_friends.txt" for testing your methods. Please provide the following output from your program:

1. Print out the number of the friends who get recommended in task 2 for those 10 Facebook users. Remember, the recommended friends should not be current friends.
2. Print out the top 10 recommended friends for those 10 Facebook users in task 3. For each recommended friend, indicate the number of mutual friends.
3. Print out the chain of users between these 5 pairs of Facebook users. For example, for two users David and Alice, the output should be "David -- Bob -- Ann -- Alice"**.** It is possible there are several chains of users. Printing out only one of them is sufficient.

Hints and Suggestions

1. Try to get portions of the program to work step-by-step. Debug your program using a small list of users and edges first. Only use the complete file once you feel confident you program works.
2. Start early. Even though we provide a lot of the basic code from the lectures, it will still take a while to write and debug these methods.
3. To find a chain of users for two given users, you can modify the breath first search (BFS) algorithm by recording the parent user who connects him/her to the first user. You can get the chain of users by tracing back to parent users through the second user.

What to Submit

Please submit all source files for your project. All the codes should be well commented.

Reminder: The course honesty policy requires you to write all code yourself, except for the code that we give to you. Your submitted code will be compared with all other submitted code for the course to identify similarities. Note that our checking program is not confused by changed variable or method names.

Grading

- Code works (50%)
- Comments (20%)
- Program structure (20%)
- Readability (10%)