Communities within a Social Network

UCSD Team

Overview

This project investigates the communities within a social network. The first part of the project will look at relationships between a user's friends (specifically, are they all connected as friends too). In the second part, the goal will be to identify sub-communities within a larger social network.

Data

The provided UCSD facebook data.

Easier Question

For a given user, which of their friends aren't connected as friends? For example, if the given user, Maria, is friends with both Jamaal and Huang, if Jamaal and Huang are not friends, we'll suggest them as potential friends.

Algorithms, Data Structures, and Answer to your Question:

Main Data Structure: The network has been laid out as a classic graph using an adjacency list. Each individual in the graph is a vertex and an edge between vertices represents a friendship.

Algorithm:
Input: Specific User (u)
Output: List of Pairs of Unconnected Potential Friends

```
Create a List of friends (vertices) of u (just explore all edges of u)
Create a return List of Pairs of Users
For each friend x in the List:
  For each friend y in the List:
    if(x and y are not the same and x is not already friends with y)
      add pair <x, y> to the return list
return the return list
```

Answer: For the dataset given, running this on nearly any user created a fairly large output of potential friends. For sake of brevity, this list is omitted.

Algorithm Analysis:

Let V be the number of friends of a given user (this could be as large as number of vertices in the graph). Testing if a user is friends with another user (is x already friends with y) can be done in O(1) because I use a HashMap in my Vertex class to store my edges. Adding a pair to the return list is also O(1) because I use a LinkedList for my return list. Because I have doubly nested loops over the list of friends and the operations within that loop are O(1), the runtime would be $O(|V|^2)$. Given that the nature of the problem requires examining all possible pairings of friends, there does not seem to be a more efficient approach.

Testing:

I created three small datasets. The first was a small example network I'd used to develop the algorithm. The second was still small but very sparse, to see if all edges got caught. The third was just a single vertex to test a corner case. The testing was useful, I caught a bug where pairings of the same vertex with itself were returned (I'd forgotten the first condition in the if statement in the algorithm). More testing could be done, but the algorithm succeeded on all three cases.

Reflection:

No changes were required yet. I think I setup my data structures with the "easy" problem in mind, so I suspect there will be changes when I move to the larger problem.