CS 225 Final Project: Project Proposal

Can I be Friends with You?

Leading Question/Goal

The goal of this project is to analyze a social media network. It will show who has the most connections with everyone ("central node" of network graph), using BFS and DFS. This is to find who may be considered to be the most influential in the chosen social media. Next goal we aim to achieve is by using Djikstra's algorithm and Landmark algorithm. It will be used to find out how close a user is to another user which will allow users to see if they can be friends with the user through other connections you already have. More detailed description of how algorithms and dataset will be utilized and used is in the dataset and algorithms sections below.

Dataset

http://snap.stanford.edu/data/ego-Facebook.html

The dataset we will be using is from the Stanford Large Network Data Collection. This dataset consists of anonymized friends lists (cicles) from Facebook, including profiles and ego networks. The Facebook dataset has 4,030 nodes (profiles) and 88,234 edges which show the connections or "friends" each profile has. The files we will be using are: nodeId.edges, nodeId.circles, and nodeId.feat. nodeId.edges contains the edges in the network data for the node 'nodeId'. Edges are undirected for facebook (an edge means user a and user b are "friends"). nodeId.circles has the set of circles for the central node. Each line contains one circle, consisting of a series of node ids. The first entry in each line is the name of the circle. nodeId.feat lists the profiles for each of the nodes that appears in the edge file. All data files listed will be downloaded and stored in git repository. The data we will be using is already a set of graphs, so we won't need to process the data more than it already is.

For our backup dataset, we have chosen Twitter social circles, which have the same files and contents as our main dataset of Facebook, but from Twitter.

Algorithms

We aim to find the users with the most direct connections/adjacent nodes using BFS and DFS. We will analyze how close two users are in their social network and how they are connected using Dijkstra's Algorithm. We will also utilize the Landmark Path Algorithm to find a path that passes through a given set of nodes.

We will implement two functions using **BFS** and **DFS**. For the first one, we will use BFS. Given the one node, the function will count the number of connections the node has. For the second one, we will use DFS. Given the set of nodes, the function will rank all nodes in the set by the number of connections to other nodes. The time and memory complexity of both functions are O(n) since we will have to traverse through every node and maintain a queue and stack for the nodes.

For the implementation of **Dijkstra's Algorithm**, given the starting node and node to find, the functions will find the shortest length path between the two nodes. One function will return the length of the path and the other will return a vector of nodes from the starting node to the destination. The time complexity will be O(nlogn) and the space complexity will be O(n) if we use a min priority queue.

For the implementation of the **Landmark Path Algorithm**, given the starting node, a list of landmark nodes, and node to find, the functions will find the shortest length path between the two nodes that passess through the landmark node. One function will return the length of the path and the other will return a vector of nodes from the starting node to the destination. The time complexity and space complexity will be similar to Dijkstra's Algorithm.

Timeline

Due 04/07: Data processing and full structure/layout of program settings
Due 04/14: BFS and DFS algorithms written and verified with tests

- 04/15 Mid-Project Check-in -

Due 04/21: Dijkstra's Algorithm written and verified with tests
Due 04/28: Landmark Path Algorithm written and verified with tests
(If time allows, graphs visualized more than prints)
Due 05/05: Final written report and presentation video and slides (+ README file)