Brandon Wong
DS 210

“Friends of Friends” on Facebook

Introduction

For this project, I wanted to find two friends with the most similar and most dissimilar sets of connections. I found it interesting because I wanted to see how connects relate to similarity. So for my measure of similarity I defined it as the two people with the most number of the same connections.

Data

The dataset I used was taken from “https://snap.stanford.edu/data/ego-Facebook.html”. I used the facebook_combined.txt.grz file. It consists of 'friends lists' from Facebook and was made up of 4039 nodes and 88234 edges. Facebook data was collected from survey participants using the Facebook App. The dataset includes node networks showing who is connected to who.

Methodology

This code reads in a file containing pairs of integers, (EX: (1,2),(1,4)) representing connections between nodes in a graph. This is done using the read_file function that reads in the input file and returns a vector of tuples, where each tuple contains two nodes that are connected to each other. Then the breadth_first_search function uses this input to generate an adjacency list of the nodes in the graph, and performs a breadth-first search on the graph to find all of the nodes that are reachable from the starting node.

The find_least_similar_nodes and find_most_similar_nodes functions take the adjacency list generated by breadth_first_search as input, and return a tuple containing the two nodes with the least and the most number of similar connections, respectively. They do this by creating a map of nodes to their connections, and then iterating over the entries in the map. For each entry, they compare the node's connections to the connections of all other nodes in the map, and update the least/most similar pair of nodes if necessary. So the pair with the highest or lowest number of the same connections are returned.

Results

I was successful in finding the most and least connected people. I was able to make a list nodes and their connections which can be called through the breadth_first_search function. The nodes (1941, 1912) and have the most similar connections while the nodes (905, 1337) and have the least similar connections. So in other words they have the most or least number of similar friends. The BFS algorithm was a good algorithm to use as its runtime was O(n + m), where n is the number of nodes in the graph and m is the number of edges. But I did take a while around 2 minutes to run both the find_least_similar_nodes and find_most_similar_nodes functions.