Madison Marchionna
DS210
12/12/23

Final Project Report

Overview

I analyzed two undirected graph datasets from Stanford's Large Network Dataset Collection. My project is a social media graph analysis tool that can help researchers determine how connected users on different platforms are. The code works by representing Facebook and GitHub networks as undirected graphs, where the labeled nodes represent users and the edges are mutual follower relationships between them.

BFS Module

The BFS module defines the graph struct and implements several functions, almost all of which are public. The struct has one field, adjacency list, which is a vector of vectors representing the adjacency list of the graph. For example, the first nested vector contains the indices of vertices that are connected to node 0. To create the graph I implemented two functions, new and add_undirected_edge. The new method is simply a constructor for the Graph struct, and it works by initializing an empty adjacency list with a specified number of vertices.
Next, the add_undirected_edge method adds an undirected edge between two vertices by updating each of their adjacency lists. The remainder of the functions are vital to the graph analysis. Additionally, many of them are used internally within another function.

The bfs method performs a breadth-first search (BFS) starting from a given vertex and returns a HashMap containing the distances from the start vertex to all other vertices in terms of the number of edges.

Similarly, the avg_path_length method calculates the average path length between any two nodes in the graph. It achieves this by performing BFS for each node. Using a nested for loop, it iterates through each distance in the HashMap. It adds all of the distances and divides by the total number of paths.

The bfs_max_distance method finds the maximum distance from a starting vertex to any other vertex in the graph. It uses the bfs function and maintains an array to store distances from the start vertex. The furthest_points method determines the maximum distance between any two nodes in the graph, using a loop to iterate through the nodes. It uses the previous function.

The local_clustering_coefficient method calculates the local clustering coefficient for a given node. In the main file, I chose to output the average coefficient for the whole graph, to give researchers a general sense of the tendency of users to form local clusters or groups. It measures how often a node's neighbors are directly connected with each other. The function uses the number of connected neighbors divided by the total possible connections among them.

Finally, the check_neighbors method is a helper function that is used by the local_clustering_coefficient function. It determines whether two nodes are neighbors by checking the graph's adjacency list.

Graph Module

The graph module contains one function, read_file, that takes a file path (&str) and a number (usize) as parameters. It returns a graph, using the struct defined in the previous module. I set the maximum number of vertices to be 15000 to reduce the run time of the program. The function works by reading the text file, extracting the edges into a vector, then constructing a graph based on the collected edges. It utilizes two functions from the BFS module, new and add_undirected_edge, to create the graph.

Main

The main.rs file uses the functions defined in the other modules to print information regarding six degrees of freedom for each dataset. For each file path, it constructs a graph and prints the average distance between any 2 nodes, the shortest path between the two furthest points, and the graph-wide clustering coefficient.

To run the code, type "cargo run" in the terminal. The code takes about 90 seconds to run. The output should look like this:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\mamar\DS210 HW\210 Final Project> cargo run
   Compiling my_crate v0.1.0 (C:\Users\mamar\DS210 HW\210 Final Project)
    Finished dev [unoptimized + debuginfo] target(s) in 0.81s
     Running `target\debug\my_crate.exe`
Average distance between two nodes for Facebook: 3.354
Minimum distance between the two furthest nodes: 8
Average local clustering coefficient: 0.291

Average distance between two nodes for GitHub: 4.448
Minimum distance between the two furthest nodes: 12
Average local clustering coefficient: 0.004
PS C:\Users\mamar\DS210 HW\210 Final Project>
```

Similarly, to run the tests type "cargo test" in the terminal. The output should look like this:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\mamar\DS210 HW\210 Final Project> cargo test
   Compiling my_crate v0.1.0 (C:\Users\mamar\DS210 HW\210 Final Project)
    Finished test [unoptimized + debuginfo] target(s) in 0.76s
     Running unittests src\main.rs (target\debug\deps\my_crate-010a4e58051bcf37.exe)

running 3 tests
test test_furthest_points ... ok
test test_bfs ... ok
test test_avg_path_length ... ok

test result: ok. 3 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

Resources

https://stackabuse.com/courses/graphs-in-python-theory-and-implementation/lessons/breadth-first-search-bfs-algorithm/
https://doc.rust-lang.org/std/collections/struct.VecDeque.html