Project Report - DS210 - May Al Khalifa

The dataset I have used is the *Amazon product co-purchasing network and ground-truth communities* from the Stanford SNAP dataset collection *(https://snap.stanford.edu/data/com-Amazon.html).* This dataset is an undirected network of Amazon products that are frequently purchased together. The dataset is composed of 334,863 nodes and 925,872 edges.

The project can be found on my GitHub repository (https://github.com/mayhalk/DS210-VSsetup). To run, download the source files to your local machine and navigate to the project directory in your terminal where you can run *Cargo build* and *Cargo run –release* to execute the program with the necessary optimizations. Ensure that the data.txt file is in the same directory so the program can access it.

This analysis uses the Breadth-First Search technique along with other path related metrics and connected components analysis to explore the network dynamic. The aim of this project is to uncover the underlying structure of the network in order to gain helpful insights regarding product co-purchasing on Amazon to give insights on customer purchasing and how to influence it or how to position one's product in the network to influence co-purchasing of their product and thus, increase profit.

Project Goals:

- To identify connected components within the network
- To calculate the average path length based on a sample of the network
- To execute Breadth-first Search analysis to uncover the network structure
- To execute Shortest Paths to uncover network structure in regards to a specific node

- To analyze degree centrality and its implications on the network structure by finding the highest degree node

Each of these analysis techniques adds unique insights in understanding the Amazon co-purchasing network in terms of product relationship and connectivity that may aid corporations in identifying new opportunities and correctly positioning their products, this will be expanded upon in the results.

Implementation:

The project is organized into four files: utils.rs, graph.rs, analysis.rs and main.rs.

The utils.rs file is where the data is read and a graph is loaded in based on the data. The function first reads the file line by line and skips any comments or empty lines and creates a graph with all the nodes and edges between them. The function is designed to successfully parse the data with correct error handling to avoid crashes.

The graph.rs file handles graph construction of a basic undirected graph. It defines many methods used for graph structure and other functions within analysis.rs. The graph is represented using an adjacency list of a node and its neighbors. It includes methods: add_edge, get_neighbors, nodes, and get_adj_list which is essential for the more advanced analysis functions. All these essential functions are tested on a simple graph that was constructed to ensure they are functioning correctly with the correct outputs.

The analysis.rs file contains all the analysis functions starting off with bfs, bfs is used to traverse the graph and list all reachable nodes it visited from node 1. This gives us unique insights of the network structure in terms of connectivity. The bfs shortest path is also executed,

although similar to bfs, it also records path distance via the number of edges traversed and the start vertex is the node with the highest degree centrality. The average path length is then calculated based on a subset of the graph for efficiency purposes, this subset is a random sample of 1,000 nodes that is used to approximate the average path length of shortest path for all nodes in the network. Then a function to find connected components in the network is used, it has a similar approach as bfs and all unvisited nodes are traversed to find smaller clusters of connected components. Finally, degree centrality is calculated by counting the number of neighbors (edges) each node has in the adjacency list, this is then used to find the node with the highest degree centrality by finding the node with the maximum number of edges. Lastly, analysis.rs tests that each function is working correctly by creating a small simple graph and testing all the functions on it to see whether its outputs are correct.

The main.rs file then executes the program and writes all the results from the functions into a well-formatted results.txt file for readability.

Results:

The results of this project can be found in the results.txt file due to the size of the data set. This txt file organizes all the outputs of the functions into an easily readable format, with headings and strategic spacing. The file is composed of the results of all the analysis functions including: connected components, highest degree node, average path length, breadth-first search, and shortest paths from the highest degree node.

The Breadth-First algorithm was used to gain insights on product reachability and market influence. By looking at a given product, through BFS we are able to determine which other

products we could reach by a series of co-purchases, allowing for the possibility of influencing consumer purchasing decisions to reach our intended node or product.

With this idea in mind, I decided to implement a few other analysis techniques. I first started off by analyzing the product ecosystem through connected components to see whether there were any market opportunities in smaller isolated areas where products did not have many co-purchasing activity. After running the analysis it was discovered that the graph consisted of one large connected component of size (334,863), which is all the nodes. Meaning the graph is strongly connected with every product being co-purchased directly or indirectly with other products. There were no isolated components for unique market opportunities, so instead I focused on other analysis techniques to gain a better market understanding and thus, advantage.

I then calculated the degree centrality of all the nodes and was able to find the highest degree node (Node: 548091 with a degree of 549) meaning this product is popular for co-purchasing other products and can be easily linked with other product purchases. The product holds a strong position in the network that corporations of other products would want to emulate. Since this product seems to be a hub for co-purchasing with many other products I decided to use it as the start vertex in the shortest path analysis in order to gain insight on how to more strategically place/advertise other products in order to increase the amount of co-purchases.

I first calculated the average path length of the network as a whole which came out to be (12.00), meaning that the shortest paths between a sample of 1,000 nodes approximates an average path length of (12.00), thus, all products, on average, can reach the product that acts as a hub of co-purchases, allowing corporations to identify stronger and more strategic positions for their product to increase co-purchasing.

Lastly, the shortest paths analysis was conducted, which traverses the dataset similar to bfs but lists all distances from the highest degree node and can be seen in the results.txt due to its size, the results can help identify product positioning opportunities to transform products into co-purchasing hubs similar to the start node.

Sources:

- Lecture notes
- https://snap.stanford.edu/data/com-Amazon.html
- https://github.com/git-guides/git-commit
- https://www.youtube.com/watch?v=mJ-qvsxPHpY
- https://doc.rust-lang.org/std/io/struct.BufReader.html
- https://doc.rust-lang.org/book/ch09-02-recoverable-errors-with-result.html
- https://doc.rust-lang.org/std/collections/struct.VecDeque.html
- https://docs.rs/adjacency-list/latest/adjacency_list/
- https://doc.rust-lang.org/book/ch11-01-writing-tests.html
- https://medium.com/@thomasmeissnerds/connected-components-using-rust-50140683bfbe
- https://dev.to/oliverjumpertz/how-to-write-files-in-rust-m06?comments_sort=top
- Chatgpt for debugging purposes