Josephine Kim
December 11, 2023
DS 210 Final Project Write Up

For this project, I used a euroroad.csv dataset from http://konect.cc/networks/subelj_euroroad/. I have 3 modules in my project: `main.rs`, `sixd.rs`, and `tests.rs`. This dataset is the international E-road network, a road network located mostly in Europe. The network contains 1,174 nodes and is undirected; nodes represent cities and an edge between two nodes denotes that they are connected by an E-road. I chose this dataset because it represents a comprehensive road network primarily in Europe, providing valuable insights into the connectivity between cities through the E-road system. It also offers a suitable basis for exploring graph algorithms and analyzing the relationships between different urban centers in the region.

The Rust code in my `sixd.rs` defines a `Graph` struct equipped with functions to analyze the structural and connectivity aspects of a graph. I started by creating an instance of the `Graph` struct with a vector of edges representing connections between nodes. The `degrees_of_separation` function calculates the shortest path distance between two specified nodes using breadth-first search, providing a measure of how closely connected the nodes are. The `average_degree_of_separation` function computes the average distance from a given node to all other nodes in the graph, offering insights into the centrality of the specified node. The `mean_degree_of_separation` and `median_degree_of_separation` functions extend this analysis to provide average and median separation values across all nodes.

My `main.rs` code declares mod sixd and mod tests. It then begins by reading and parsing the "euroroad.csv" file. The code cleans the file to then construct a graph with edges representing connections between nodes. The random selection of nodes demonstrates the flexibility of the code in analyzing different parts of the graph dynamically. The subsequent calculations and print statements for distance, average degree of separation, mean degree of separation, and median degree of separation showcases various aspects of the graph's structure. The utilization of random nodes ensures a diverse exploration of the graph, contributing to a comprehensive understanding of its connectivity and centrality. To get the results, type “cargo run” into the terminal, which should be in the “project” directory, and click “Enter.”

The distance between nodes 70 and 32 (chosen at random), calculated as 12, indicates the length of the shortest path connecting these nodes. This distance metric reflects the efficiency of connectivity between the two nodes, with a smaller distance indicating a more direct route. The average degree of separation for node 3, computed as 6.58, offers insights into the overall connectivity of node 3 to the remaining nodes in the graph. This metric quantifies, on average, how many steps are needed to traverse from node 3 to any other node. The higher value of 6.58 suggests that node 3 is reasonably well-connected to the broader network. The mean degree of separation represents the average distance from any node to all other nodes in the graph, or the number of steps required to traverse from one city (represented by a node) to

any other city in the network. In this context, a mean degree of separation around 13.45 suggests that, on average, nodes in the graph are connected by relatively short paths. Lower values would indicate a more tightly connected graph, while higher values might suggest a more loosely connected or fragmented graph. The median degree of separation, recorded as 15.00, shows that half of the nodes in the graph have a separation less than or equal to 15.00, and the other half have a separation greater than or equal to 15.00.

The `tests.rs` file in Rust serves as a module for organizing tests to ensure the correctness of the code within the `sixd` module. The code within the module is enclosed in a `#[cfg(test)]` attribute, indicating that it should only be compiled when running tests. The module imports the `Graph` struct from the `sixd` module and includes several test functions. The `test_six_degrees_of_separation` function defines a test graph and asserts that nodes 1 and 6 are connected within six degrees of separation. The `test_average_degree_of_separation` function creates another test graph and verifies the average degree of separation for nodes 1 and 3. Lastly, the `test_mean_and_median_degree_of_separation` function checks the mean and median degree of separation for a different test graph. Each test function uses the `assert_eq!` macro to compare the actual results with the expected results, ensuring that the implemented functions behave as intended. To see if the tests run, type “cargo test” into the terminal, which should be in the “project” directory, and click “Enter.”