DS210 Final Project Report

Rebekah Lewi

The dataset that is used in this project is called Euroroad. It represents a network of roads across Europe, where each node corresponds to a specific location (such as a city or road intersection) and each edge (pair of nodes) represents a direct road connection between these locations. To integrate this dataset into my project, I created a separate module called readfile.rs which contains the public function to read the file.

It starts by opening the file using File::open and applies the question mark operator (?) for error handling, which propagates errors up the call stack. The function iterates over each line of the file using the lines() iterator, which yields results that may be either an Ok value containing a line or an error. Each line is expected to contain two unsigned integers separated by a comma, representing an edge between two nodes in a graph. These strings are split and parsed into usize. If both numbers are successfully parsed, they are collected into a tuple. If not, lines with parsing errors are discarded. This filtered and parsed data is then collected into a vector of tuples and returned wrapped in an Ok, ensuring that the function handles potential I/O and parsing errors and then it returns a clean set of data.

Next, in the module called analysis.rs is where I set up the graph structure and implement essential graph algorithms such as breadth first search (BFS). It first defines a Graph data structure and provides methods for constructing the graph and analyzing it with BFS. The Graph struct includes two fields: Nodes, a vector of i32 that lists all nodes in the graph, and AdjacencyList, a HashMap where each key is a node and its value is a HashSet of connected nodes, representing the graph's connections. The Graph::new function takes a slice of tuples, each representing an edge between two nodes given as usize. Here, each node is converted to i32 using try_into(). It adds both nodes to the Nodes vector and updates the AdjacencyList to reflect the bidirectional connectivity inherent in most graphs by inserting each node into the other's set. After processing all edges, it ensures that each node is listed once. The are_connected function checks if two nodes are directly connected by looking up the first node in the AdjacencyList and checking if it contains the second node in its set. The graph_analysis function performs a BFS starting from a specified node. It initializes a queue with the start node and its initial distance (0) and a HashMap to store distances from the start node to others. As the function iterates through the queue, it examines each node's neighbors (from the AdjacencyList), and for each neighbor not already visited, it adds them to the queue with an incremented distance and records this in the distances map. The distances HashMap is then returned, providing a complete map of shortest paths from the start node to all reachable nodes in the graph.
In the stats.rs module there are 3 functions that perform statistical analysis on a set of distances stored in a HashMap<i32, i32>, where the key is an identifier for a node and the value is the distance to that node. The average function calculates the average distance by iterating through the values of the hash map, summing them up, and then dividing by the total count of entries. The max_distance function takes all distance values into a vector, sorts them in descending order, and returns the largest value, which represents the maximum distance. Lastly, the

find_threshold function categorizes the distances into three groups: less than 7, between 7 and 15 inclusive, and greater than 15. It counts how many entries fall into each category and returns these counts as a tuple.

Lastly, the main.rs contains the main function which begins by reading the dataset from the file "euroroad.csv" using the read_file function from the readfile module, which returns a list of edges, each defined as a pair of nodes. These edges are then used to construct a Graph object, defined in the analysis module. The graph is represented by nodes and their connections, which are stored in an adjacency list format within the Graph structure. After constructing the graph, the program iterates over each node. For each node, it performs a graph analysis starting from that node, calculating distances to all reachable nodes in the graph. This analysis is executed by the graph_analysis function from the analysis module. The results of the graph analysis are then used in three different statistical evaluations provided by the stats module. The average function calculates the average distance to all nodes from the starting node. The max_distance function finds the maximum distance from the starting node to any other node. Lastly, the find_threshold function categorizes the distances into three groups: distances less than 7, distances between 7 and 15 inclusive, and distances greater than 15, and counts the number of nodes in each category.

For each node used as a starting point in the graph, the program prints the node number along with its average distance to other nodes, the maximum distance found, and the distribution of distances according to the defined thresholds. It also prints the total number of connections (nodes within each distance threshold) for each starting node as shown in the output example below.

Output:

```
Node: 1170, Average distance: 22.512993, Maximum distance: 56
Less than 7: 31, Less than 15: 204, Greater than 15: 804, Total Connections: 1039

Node: 1171, Average distance: 27.418673, Maximum distance: 61
Less than 7: 13, Less than 15: 77, Greater than 15: 949, Total Connections: 1039

Node: 1172, Average distance: 28.416746, Maximum distance: 62
Less than 7: 12, Less than 15: 61, Greater than 15: 966, Total Connections: 1039

Node: 1173, Average distance: 0.5, Maximum distance: 1
Less than 7: 2, Less than 15: 0, Greater than 15: 0, Total Connections: 2

Node: 1174, Average distance: 0.5, Maximum distance: 1
Less than 7: 2, Less than 15: 0, Greater than 15: 0, Total Connections: 2

(base) rebekahlewi@crc-dot1x-nat-10-239-35-150 final-project %
```

From the output, we can see that most of the nodes have a total connection of 1039 which means that they are a part of a large, densely connected network. The average distance, maximum distance and the threshold counts varies within each node. A low average distance means that a node is closely connected to most other nodes in the network. This suggests that the node is centrally located within a densely connected cluster, where paths to other nodes are

relatively short. A low maximum distance indicates that the furthest reachable node is not very far away, suggesting that the node's subnetwork is compact or limited in reach. The threshold counts, below 7, between 7 and 15, and greater than 15, help identify how many connections fall within certain distance ranges, providing a distribution profile of connectivity.

To ensure that the whole program works properly, I implemented several tests in main.rs that ensure each function runs. It first creates a small dataset that is similar to the euroroad.csv format and then simulates reading the file, creating the graph, and analyzing it. The test then checks if the analysis results meet expected outcomes along with other statistical functions such as average, maximum distance and the threshold counts. The test is then successful proving that the whole program works.