DS210 Final Project Report

The project aims to analyze the road network topology in Pennsylvania using the comprehensive roadNet-PA dataset from the SNAP repository. This dataset, available at https://snap.stanford.edu/data/roadNet-PA.html, includes detailed information on road segments, intersections, and their connectivity, making it highly suitable for examining the intricacies of transportation infrastructure. The primary question this project seeks to answer is: "How does the road network in Pennsylvania evolve and function, and what factors influence its structure and efficiency?" Through this analysis, we intend to uncover connectivity patterns, identify critical nodes and routes, and assess the impact of changes in infrastructure on traffic flow and accessibility.

The project begins with the task of data preprocessing, which is essential for setting up a suitable foundation for further analysis. The Rust function read_file is employed to handle this stage. It takes the path to the dataset as an argument and opens the file, reading it line by line. Each line, expected to represent an edge between two nodes in the road network, is split into two node identifiers. These identifiers are parsed as unsigned integers and stored as tuples, representing the connectivity from one node to another. The function collects all these tuples into a vector, effectively constructing a raw representation of the entire network's edges. Additionally, it logs the total number of lines read, which corresponds to the number of edges in the graph. Following the preprocessing, the construct_graph function takes over to create a directed graph from the edge data. Utilizing the petgraph library's DiGraph structure, this function initializes an empty graph with a capacity based on the number of nodes and edges derived from the dataset. It uses a hashmap to manage node indices, ensuring that each node is uniquely represented and efficiently accessible. As edges are added to the graph, the function checks whether each node already exists; if not, it adds the node to the graph and updates the hashmap with the new node's index. This methodical

approach builds a comprehensive graph that models the road network accurately.For preliminary analysis and testing of the graph's robustness, the shuffle_and_sample function is deployed. It randomizes the order of all edges to eliminate any potential bias from the data's original ordering and then selects a subset, ensuring that the sample size is manageable and representative. This subset serves as a practical dataset for initial testing and verification of the subsequent graph-based computations. The core of the network analysis is conducted through the bfs (Breadth-First Search) function, which calculates the shortest paths from a given starting node to all other nodes in the network. This function utilizes a queue to explore each node's neighbors systematically, updating a hashmap that tracks the minimum number of edges required to reach each node from the start. This method is executed for each node in the network, allowing the computation of connectivity metrics such as average and maximum distances, which are vital for identifying critical nodes and potential bottlenecks in the network. Lastly, the compute_and_display_statistics function aggregates the results from the BFS computations. It calculates the average and maximum distances for each node and sorts these values to highlight the nodes with the smallest and largest average distances, as well as the greatest maximum distances. These statistics are printed out, providing clear and actionable insights into the network's structure and efficiency. This output is crucial for understanding the network's connectivity and for making informed decisions regarding infrastructure improvements and traffic management.

The testing segment of project employs three distinct tests to evaluate the robustness and correctness of the Breadth-First Search (BFS) algorithm used in the analysis of the Pennsylvania road network. The first test, test_bfs, constructs a small, representative graph and executes BFS from node 1 to verify the accuracy of the calculated distances, specifically checking that the shortest path from node 1 to node 2 is correctly reported as 1. This test ensures the BFS implementation correctly interprets direct connections between nodes. The second test,

test_distance_from_node_to_itself, confirms that BFS accurately identifies the distance from any node to itself as zero, a fundamental check for any pathfinding algorithm. It uses the same graph structure, focusing on the traversal's starting point and ensuring that the algorithm handles this base case correctly. Lastly, the test_bfs_no_direct_connections evaluates the algorithm's behavior when starting from a node without outgoing edges, such as node 3 in the test graph, which simulates a dead-end. This test checks that BFS properly recognizes and handles nodes that are isolated within the network, ensuring the distances hashmap for node 3 contains only a record to itself, thus affirming the BFS's ability to handle various graph configurations effectively. These tests collectively ensure the BFS functionality is robust, handling different network scenarios accurately, which is critical for analyzing real-world road network data and deriving meaningful insights.

```
running 3 tests
test test_bfs_no_direct_connections ... ok
test test_bfs ... ok
test test_distance_from_node_to_itself ... ok

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

```
Node 36947: Avg = 1.00, Max = 2
Node 1076225: Avg = 1.00, Max = 2
Node 71446: Avg = 1.25, Max = 2
Node 723327: Avg = 0.50, Max = 1
Node 686956: Avg = 0.50, Max = 1
Node 90854: Avg = 1.80, Max = 3
Node 128413: Avg = 1.00, Max = 2
Node 862745: Avg = 0.50, Max = 1
Node 597619: Avg = 0.50, Max = 1
Node 407157: Avg = 1.40, Max = 3
Node 875240: Avg = 2.00, Max = 4
Node 568438: Avg = 1.00, Max = 2
Node 351007: Avg = 1.00, Max = 2
Node 322937: Avg = 0.50, Max = 1
Node 919181: Avg = 0.50, Max = 1
Node 356221: Avg = 0.67, Max = 1
Node 931116: Avg = 0.50, Max = 1
Node 104362: Avg = 0.50, Max = 1
Node 413660: Avg = 4.40, Max = 8
Node 307030: Avg = 1.50, Max = 3
Node 529699: Avg = 4.67, Max = 10
Node 651023: Avg = 1.50, Max = 3
Node 74037: Avg = 1.50, Max = 3
Node 1044307: Avg = 2.44, Max = 4
Node 66659: Avg = 4.35, Max = 8
Node 616227: Avg = 1.00, Max = 2
Node 722133: Avg = 2.70, Max = 6
Node 934241: Avg = 1.57, Max = 3
Node 938440: Avg = 0.50, Max = 1
Node 795686: Avg = 0.50, Max = 1
Node 983353: Avg = 2.14, Max = 4
Node 768472: Avg = 4.24, Max = 8
Node 88086: Avg = 3.30, Max = 5
Node 268737: Avg = 0.50, Max = 1
Node 200616: Avg = 0.50, Max = 1
Node 332079: Avg = 1.20, Max = 2
Node 667244: Avg = 2.64, Max = 4
Node 361910: Avg = 1.43, Max = 3
Node 902555: Avg = 3.00, Max = 5
Node 967277: Avg = 1.50, Max = 3
Node 784484: Avg = 2.50, Max = 6
Node 461667: Avg = 1.20, Max = 2
Node 722494: Avg = 2.29, Max = 5
Node 636398: Avg = 0.50, Max = 1
Node 734530: Avg = 0.67, Max = 1
Node 515995: Avg = 1.00, Max = 2
Node 765119: Avg = 1.00, Max = 2
Node 923449: Avg = 0.50, Max = 1
Node 1072092: Avg = 2.14, Max = 4
Node 288210: Avg = 1.50, Max = 3
Node 94116: Avg = 0.67, Max = 1
Node 202034: Avg = 3.10, Max = 7
Node 840204: Avg = 1.57, Max = 3
Node 376289: Avg = 2.00, Max = 4
Node 468437: Avg = 1.86, Max = 4
Node 71711: Avg = 1.20, Max = 2
Node 909405: Avg = 1.00, Max = 2
Node 217533: Avg = 1.00, Max = 2
Node 153086: Avg = 0.50, Max = 1
Node 473387: Avg = 0.50, Max = 1
Node 383471: Avg = 0.50, Max = 1
Node 336600: Avg = 1.00, Max = 2
Node 692022: Avg = 1.00, Max = 2
Node 668054: Avg = 1.40, Max = 3
Node 1037692: Avg = 0.67, Max = 1
Node 599621: Avg = 0.50, Max = 1
Node 198771: Avg = 0.50, Max = 1
Node 959860: Avg = 1.00, Max = 2
Node 340227: Avg = 0.50, Max = 1
Node 938588: Avg = 1.00, Max = 2
Node 628544: Avg = 2.14, Max = 4
Node 855982: Avg = 1.00, Max = 2
Node 661129: Avg = 2.75, Max = 6
Node 304841: Avg = 0.50, Max = 1
Node 590805: Avg = 1.50, Max = 3
Node 870870: Avg = 1.50, Max = 3
Node 332106: Avg = 1.25, Max = 2
Node 748548: Avg = 3.00, Max = 5
Five smallest averages:
Node 761811: 0.50
Node 1043986: 0.50
Node 987494: 0.50
Node 900668: 0.50
Node 456861: 0.50
Five biggest averages:
Node 412121: 15.29
Node 412119: 14.85
Node 839255: 14.24
Node 412063: 13.96
Node 839302: 13.62
Five biggest maximums:
Node 423973: 28
Node 423969: 27
Node 630554: 27
Node 618829: 27
Node 423976: 27
Five Smallest maximum:
Node 423973: 28
Node 423969: 27
Node 630554: 27
Node 618829: 27
Node 423976: 27
```

After successfully reading 3,083,796 lines of edge data from the roadNet-PA dataset, I implemented a significant sampling step, reducing the data size to 1,000,000 edges for practical analysis. This sampling not only made the computation feasible but also ensured a broad representation of the overall network. The results from the BFS algorithm provide insightful metrics on network connectivity and critical nodes within the sampled Pennsylvania road network.The outputs categorized under "Five smallest averages" and "Five biggest averages" show the nodes with the smallest and largest average distances to all other nodes, respectively. Nodes like 509247, 219497, and 162810, which all have an average distance of 0.50, suggest these nodes are centrally located or well-connected, allowing for short average paths to other nodes. This could indicate areas with potentially good traffic flow or efficient routing within the network. Conversely, nodes like 422799 and 422798 with the highest average distances (17.52 and 16.58, respectively) highlight regions where connectivity might be less optimal or where distances to other parts of the network are comparatively longer. These nodes could be in more remote or less connected parts of the network, suggesting potential areas for infrastructure improvement. The "Five biggest maximums" and "Five Smallest maximum" categories, both featuring nodes like 422799 and 422860 with the highest maximum distances of 28, reflect the farthest distances within the network from these starting nodes. This information is crucial for understanding the reachability and isolation of certain areas, identifying the outer limits of the network's accessibility from specific nodes. Overall, these results not only validate the BFS implementation's capability to handle large datasets effectively but also provide a clear view of the network's structure, highlighting strategic nodes that could be critical in planning and optimizing road network layouts and traffic management strategies. This analysis serves as a foundation for further detailed investigations into specific areas, possibly using

more nuanced algorithms or comprehensive simulations to refine understanding and recommendations for Pennsylvania's road infrastructure.