DS210 Final Project: Write-Up

Juliette Peng

OVERVIEW

My project takes in a dataset with 1174 nodes and 1416 edges. The dataset "euroroad.csv" deals with roads, and the connectivity between them. Every node represents a city, and the edges between them represent the road that connects the cities. As the required number of nodes for this project was 1000+, since my dataset was just slightly above that minimum there was no necessity for me to sample size my data. Instead, I used the entire dataset and added more code in terms of the statistical functions I created. My goal for this project was to find meaningful results that would showcase any sort of statistical significance. After reading the dataset as a file, I constructed an adjacency list representation of the graph, which then allowed me to perform breadth-first search on the graph to find and calculate the distances between nodes. Then, a variety of statistics were found through creating different functions, and two tests were made to ensure the accuracy and validity of these functions.

WALK-THROUGH OF CODE

First, my "read_file" function reads my dataset file (euroroad.csv) and returns a vector of tuples representing the edges. Next, my "unique_nodes" function takes a reference to a graph represented as a vector of tuples and returns a HashSet containing unique nodes in the graph. My "adjacency_list" function constructs an adjacency list representation of the graph and returns a large vector containing vectors that represent the adjacency list. The "main" function is at the heart of my code, and is where most of the calculations, computations, and print statements are. This function utilizes the previous ones to run and print out the results of breadth-first search on the graph which finds the distances between nodes (my BFS function is in my stats module). It is also the place that runs and prints out the computations of my statistic functions in my stats module.

To go further in depth about my stats module, I chose to put all my statistic functions within this module to break up my code and make it more readable and easy to understand. First, my "bfs" function performs breadth-first search on the graph starting from a given node. It then returns a vector containing distances from this start node to all other nodes, including itself. The "mean_distance" function calculates the mean distance of the graph from all of the distances obtained by breadth-first search. To be more precise, every node has multiple distances to many different nodes. This function looks at every distance and calculates what the average distance is. The "std_dev" function utilizes the mean achieved from the previous function to calculate the standard deviation of distances in the graph. My last statistical function "max_distance" calculates the maximum distance between any two nodes in the graph. In essence, this function

just iterates through all of the distances calculated by the breadth-first search function and finds the largest one. In my dataset, 41 is the maximum distance between any two nodes.

Tests were also performed to ensure my statistical functions were working properly. My project contained two tests. I created a small sample size dataset, and manually found the answers for these tests to ensure complete accuracy. For my first test “test_bfs_zeros”, I tested if when running breadth-first search, when a node is given itself, it should always result in a distance of 0. My second test “test_max_distance” tests the reliability of my “max_distance” function to see if it truly does calculate the maximum distance in the graph. Both of my tests passed as seen below in images of my outputs.

OUTPUTS & RESULTS ANALYSIS

The output of my code includes the length of the graph (number of edges), the distance from each node to all other nodes in the graph, the mean distance of the graph, the standard deviation of distances in the graph, and the maximum distance between any two nodes in the graph.

To look further into how my project relates to analyzing road networks, I want to first briefly explain the “euroroad.csv” dataset file. This dataset represent a specific network of roads, where each road is represented as a connection between two nodes, which are cities. Each line in the dataset likely corresponds to a road segment, where the two numbers separated by a comma denote the nodes or intersections connected by that road segment. For example, the first number represents the starting node and the second number represents the ending node.

To analyze my statistical functions outputs, the mean distance between any two nodes is 8.23 while the standard deviation is 7.07. The mean distance between any two nodes in the road network represents the average length of the shortest paths between all pairs of nodes. A mean distance of 8.23 indicates that, on average, it takes 8.23 units (the dataset does not give the exact metric, although we can presume that it represents miles or kilometers since it is discussing road connectivity) to travel from one city to another within the network. This measure helps understand the overall connectivity and accessibility of the road network. A lower mean distance suggests that locations within the network are more closely connected, facilitating quicker travel between them, while a higher mean distance suggests the opposite.

The standard deviation of distances in the road network deals with the variability of distances between pairs of nodes around the mean distance. A standard deviation of 7.07 indicates how much each distance between nodes deviate from the average distance of 8.23. A higher standard deviation suggests that there is greater variability in the distances between pairs of nodes. This could imply that the road network has areas with significantly longer or shorter distances between nodes, leading to less uniformity in distances across the whole network.

The maximum distance between two nodes in the road network is 41, meaning that there exists at least one pair of nodes in the network that are 41 units apart from each other along the shortest path. This suggests that there are nodes in the network that are relatively far apart from each other. This might indicate the presence of widely separated regions within the network, and a lack of connectivity in specific areas.

By analyzing the the “euroroad.csv” dataset using algorithms and statistical measures, my project aims to provide some insights into the connectivity of the road and city network. These insights can be valuable in the real world for a variety of different purposes, such as urban planning and optimizing transportation. For instance, understanding the mean distance between nodes can help estimate travel time or distances for route planning, while knowing the maximum distance can highlight less accessible areas within the network.

IMAGES

Snippet of BFS Output:

```
Distance from node 1142
To 196: 1
To 1142: 0
Distance from node 1143
To 1143: 0
Distance from node 1144
To 141: 4
To 268: 2
To 599: 3
To 864: 1
To 1144: 0
Distance from node 1145
To 9: 7
To 18: 3
To 22: 7
To 32: 13
To 34: 9
To 35: 20
To 38: 7
To 42: 4
To 59: 6
To 85: 6
To 89: 14
To 109: 15
To 138: 14
To 139: 11
To 144: 5
To 145: 11
To 154: 29
To 165: 1
To 169: 1
To 174: 11
To 178: 14
To 179: 9
To 181: 15
To 194: 20
To 212: 8
To 244: 7
To 252: 7
```

Statistic Outputs:

```
(base) juliettepeng@crc-dot1x-nat-10-239-17-253 project2 % cargo run
   Compiling project v0.1.0 (/Users/juliettepeng/DS210-Project/project2)
    Finished dev [unoptimized + debuginfo] target(s) in 0.40s
     Running `target/debug/project`
Mean Distance: 8.230995717344754
Standard Deviation: 7.066274870842413
Max Distance: 41
```

Cargo Test Outputs:

```
(base) juliettepeng@crc-dot1x-nat-10-239-17-253 project2 % cargo test
   Compiling project v0.1.0 (/Users/juliettepeng/DS210-Project/project2)
    Finished test [unoptimized + debuginfo] target(s) in 0.28s
     Running unittests src/main.rs (target/debug/deps/project-915f8ce40abc7688)

running 2 tests
test test_max_distance ... ok
test test_bfs_zeros ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```