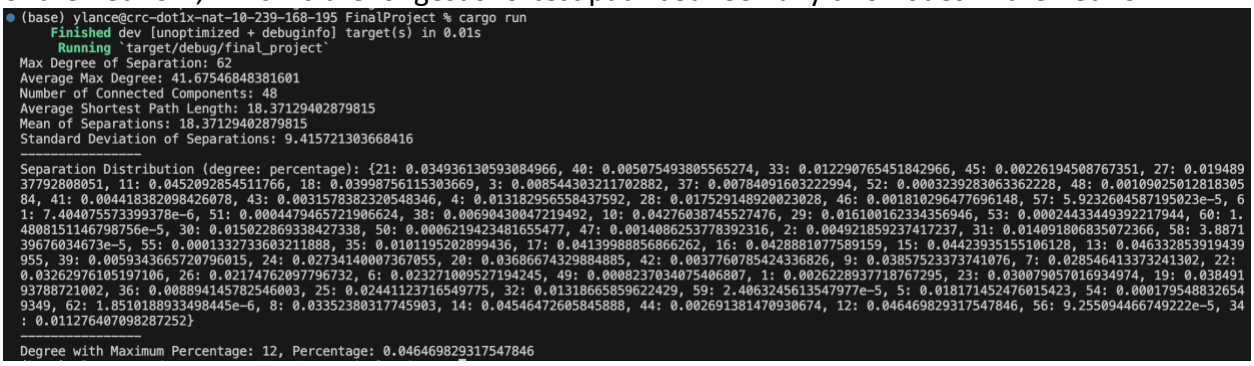
The dataset I chose represents the international E-road network, which is a network of roads across Europe and neighboring countries. The dataset consists of rows where each row represents an edge in the road network graph. In this context, an edge is a direct road connection between two cities, or nodes. The dataset consists of rows where each row contains two numeric identifiers. Each pair (row) indicates that there is a direct connection (road) between these two nodes. This forms an undirected edge in graph terms, as the roads can typically be traveled in both directions.

The adjacency list is a data structure that lists each node and its directly connected neighbors. In this dataset, constructing an adjacency list would involve iterating through each line of the CSV file and adding the corresponding cities to each other's list of neighbors. This is effectively a representation of the graph in memory that makes it easy to navigate from one node to its adjacent nodes. When constructing an adjacency list, each node will have an entry in the HashMap and the value associated with each node contains all the nodes directly connected to it. For example, if there's a row "2,17", node 2 will have node 17 in its set, and node 17 will also have node 2 in its set, indicating a bidirectional connection.

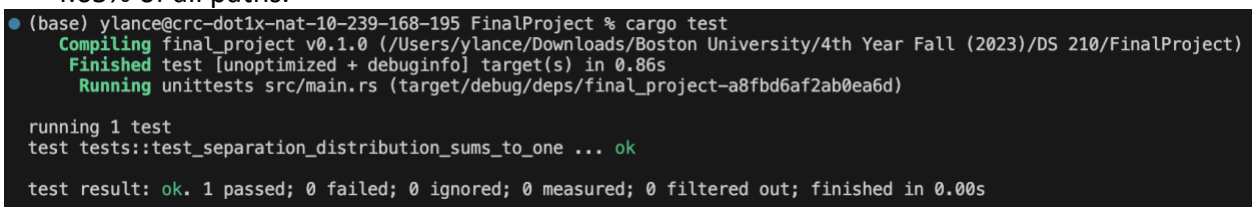
BFS can be used on this dataset to explore the road network starting from a given city. It proceeds level by level, first visiting all cities directly connected to the starting city, then all cities two steps away, and so on. This method can be used to measure degrees of separation: by tracking the number of steps from the starting node, BFS can calculate the degree of separation between nodes, which can be interpreted as the minimum number of roads needed to travel from one city to another within the network. This method can also be used to find shortest paths between the starting city and all cities reachable from it. By applying BFS to each city in the network, you can measure the overall connectivity of the network and find the "diameter" of the network, which is the longest shortest path between any two nodes in the network.



From the output, one can say that:

1. Max Degree of Separation is 62, which is largest number of edges in the shortest path between any two nodes in the graph. (It's the longest shortest route you would have to take when traveling from one city to another within the network.)
2. Average Max Degree is 41.67546848381601. This indicates how "deep" the network is on average if you start at a random node and travel as far as possible.
3. Number of Connected Components is 48. This might indicate that there are 48 isolated road systems within the overall network.
4. Average Shortest Path Length is 18.37129402879815, which measures how closely knit the network is; a lower number would suggest a more interconnected network.

5. Mean of Separations is 18.37129402879815, which tells us that the average number of edges that you must travel to go from one node to any other reachable node is approximately 18.37.
6. Standard Deviation of 9.415721303668416 and a high standard deviation indicates that the path lengths vary widely from the average.
7. Separation Distribution shows how many paths exist for each length. For example, {21: 0.034936130593084966} means that about 3.49% of all the shortest paths in the network have a length of 21 edges.
8. Degree with Maximum Percentage: 12, Percentage: 0.046469829317547846, which indicates that the most common shortest path length is 12 edges, comprising about 4.65% of all paths.



The `build_sample_network` function is presumably a smaller or simplified version of a larger network graph, making it easier to test and verify the correctness of the functions. It then calls calculate_normalized_separation_distribution with the sample network as an argument. This function is expected to calculate the normalized separation distribution in the graph, which is a measure of how nodes are connected. Specifically, it should return a distribution of the shortest path lengths between all pairs of nodes in the graph, normalized so that the sum of all probabilities equals 1. After calculating the normalized separation distribution, the test sums all the percentage values in the distribution. The assertion checks that the sum of all normalized separation percentages is approximately 1.0, within the margin of error allowed by floating-point precision. In essence, this test ensures that the normalization part of the function is working correctly and that the overall distribution of separations in the sample network is a valid probability distribution.

Finally, I would like to mention that this project has been successfully completed with the major help I received from Zachary Gentile, as he provided the dataset that will be easier to work with, compared to the BankChurning dataset I had. In addition, I've used ChatGPT's help to get the necessary functions; however, I thought through what needed to be tested and printed in the console and ChatGPT was there to assist my progress in this project.