Griffin Tomko

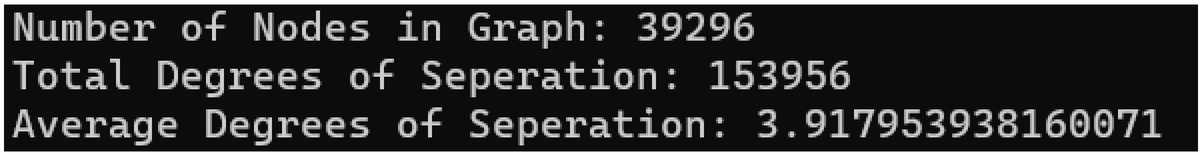
Professor Kontothanassis

DS210

15 December 2023

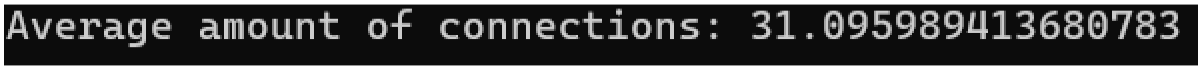
Project Write Up

When I was looking for datasets for this project, I came across a netflix data set that looked interesting to work with. This data set possesses many characteristics about each movie and tv show, but the one I want to look at is the cast. From this I wanted to compute the average degrees of separation of the actors, also known as six degrees of separation. The data was given in an excel file which I converted to a csv file in order to be accessible to me. To start my code, I created types that would be used throughout my code. These types were: Vertex, ListOfEdges and AdjancenyLists. Vertex is of the type usize, ListOfEdges is a vertex of tuples, and Adjacency list is a vector of vectors that is usize. After this I created two structs, one called Data that will be used to store the cast, and the other called Graph that will be used to create the undirected graph needed to compute the six degrees of separation. The functions I used to open and set up the csv file are under the module setup. Setup holds two functions called create_list and open_file. Create_list takes an input of a Data struct that represents the cast, a mutable hashmap that will map the actors names to indexes, the number of actors, and a mutable hashmap that stores an actor and the list of actors they are connected to. This function maps the actors name to indexes, and stores the indexes in a vector. This vector is then iterated through into a new vector that will now hold the list of edges between actors. The function open_file takes a csv file for an input. Open_file takes the csv file, in this case netflix data, and puts the cast into a vector. Then the function create_list is implemented in open_file to make the list of edges between actors. This list of edges and the number of actors is returned. After doing this, I created a function outside of this module called reverse edges and 5 more functions that implement the Graph struct. Reverse edges takes a list of the type ListOfEdges and switches the position of the tuples and returns them, hence reversing the direction of the edge. The 5 functions created under Graph are called sort_graph_list, add_edge, create_directed, create_undirected, and conn_vec. Sort_graph_list takes a reference of the struct Graph and sorts the adjacency list. The function add_edge takes a reference of the struct Graph and vector of tuples. This function then adds the edge and makes a direct connection of the inputted tuples. The create_directed function takes the number of nodes and a vector of tuples. In the function, I created a variable of the struct Graph, and used the created functions add_edge and sort_graph_lists to create a directed graph. The function create_undirected takes the same arguments and does almost the same thing as create_directed. The only difference is that create_undirected uses reverse_edges to get the graph to be undirected. Finally the last function that implements the graph struct is conn_vec. This function takes the reference of the struct Graph and returns a vector of floats. This function iterates over the adjacency list and stores the number of connections each actor has. The next two functions are within a module called dist_calc. These functions are called distance_vector and total_distance. Distance_vector takes a starting value and a reference to the struct graph. It uses a VecDeque to compute the distances from the starting point to any other point. Then it stores and returns the value as the type option in a vector. The function total_distance takes a distance vector and the number of nodes, and totals the values from the distance vector. In my main function, I used my open_file function to open the csv file and store the edges and number of nodes. Then from this I used the create_undirected function to make the graph. After making the graph, I used the distance_vector formula from point zero to get a vector of distance. After this, I calculated the total distance using the total_distance formula. Then I divided this by the number of nodes minus 1. I used n minus 1 to get rid of the connection a node has with itself. I was interested in the number of nodes, the total degrees of separation and the average degree of separation. After calculating it, these values ended up being:



```
Number of Nodes in Graph: 39296
Total Degrees of Seperation: 153956
Average Degrees of Seperation: 3.917953938160071
```

Now I was curious about the average connections of actors, so using the function conn_vec I found the vector of connections for each node. After iterating and totaling the vector, then dividing it by the average nodes, I found out an actor in this dataset has an average amount of connections:



```
Average amount of connections: 31.095989413680783
```