Group Members:

Adi Kandula (kandula4)

Avadh Patel (apate429)

Roshun Navin (rnavin2)

Neehar Sawant (neehar2)

Results

Our goal was to utilize the OpenFlights dataset to implement a shortest path algorithm to determine the most efficient and ideal path to take. We were able to utilize Dijkstra's Algorithm and Landmark Algorithm to calculate the shortest distance between point A and point B as well as with a connection point C. Lastly, we were able to implement our Breadth First Search traversal which outputs the order in which the airports are reached.

We were able to parse through the airport data set in a manner that gave us the three letter IATA for the airport, the longitude coordinate of the airport, and the latitude coordinate of the airport. We split each line of the airport data by the commas separating all the information to extract the specifics we need. We discovered that this was the best way to parse through the data. We were able to parse the data for the routes data set in a way that gave us the starting airport's three letter IATA and the destination airport's IATA. We wanted to have our input take in airport names, but the airport data required an IATA. Therefore, we mapped the IATA to the airport name and then mapped the airport name to an Airport Object which contained all the information we needed.

We utilized the parsed data into a graph by inserting a vertex for every airport and inserting an edge for every route. For creating the graph, we utilized the graph implementation from lab_flow as we realized the adjacency list was the most efficient way to create the graph.

Once we were able to implement our Dijkstra algorithm in order to figure out the minimum distance between two vertices, we were able to apply that to our map containing all the airport objects. Using the latitude and longitude coordinates, we were able to find the distance between each route. We used this distance as the weight of the edges between each vertex (airport). We then ran our Dijkstra algorithm on the map, which took in the starting airport and the destination airport as its parameters. This gave us a vector of the fastest path between two airports.

In order to implement our Landmark Path algorithm, which would find the shortest path between two airports given a specific connecting airport, we were able to use our Dijkstra's algorithm. We called Dijkstra's from the first airport to the connecting airport, and then we called it again from the connecting airport to the destination airport. This was all put in a vector which was used to output the flight path between the three airports.

We were able to implement the BFS traversal of the graph with the help of the lecture and TA notes. The BFS traversal goes through the graph from a specific starting point/airport. We discovered that BFS utilizing a queue makes the implementation much easier to write in code. This traversal is unique and no other search will follow the same traversal. This function returns a vector of the order in which the airports are visited with a BFS traversal.