CS225 Final Report: OpenFlights

Mingqi Wang, Kelly Dai, Teresa Zhang, Sihan Cheng

12/11/2022

In this project, we explore the OpenFlights dataset to understand how people in the world can be efficiently connected by airlines. Using the algorithms described below, a real-life application could generate a planned route to find the shortest path from one place to another place, visiting all the famous places in the world, etc. We plan to find the shortest flight path between the two locations, by using the nearest airports corresponding to each country.

We first downloaded the airport's dataset and the route dataset from OpenFlights. The airport dataset is in a CSV file. Each line of the airport dataset represents one airport, containing airport ID, airport name, longitude, and latitude. First of all we plan to use Python to clean our data, but we found that we can directly correct our input by ignoring the line that misses information using C++. We used the processed data to construct a weighted graph on the map. Vertices will be airports, and edges will be the route between the two airports. The weight is calculated using longitude and latitude. We used the datasets to create a graph and wrote a test case for the graph constructor. We also tested both the vector and string input for the airport class constructor.

Then we implemented our first algorithm, Breadth First Function. This algorithm was used to traverse the graph. We have three BFS functions to help us with later algorithm implementations. The first one returns a vector containing the airports that the given airport is connected to. The second function returns a vector based on the given airport and requires passing the airport. The last one returns a vector given starting airport and destination airport. Later, we used the BFS algorithm to realize the PageRank algorithm. The test case we built for BFS part is checking the successful implementation of BFS traversal.

The next algorithm we implemented was the PageRank, used to find the top important airports. We first found an adjacency matrix that contains the probability of every airport going to a different airport from the dataset graph. After that, we randomly generated a vector. After normalizing it, we put it into iterations for a long time to get the steady-state vector. Based on the number of important airports we want to find, we select the largest number from the steady-state vector. Using the index of the number, we can get the top most important airports' ID. The test cases we built for the PageRank algorithm tested the usability of topAirport, makeA, and rank function. In addition, we built a test case to test whether the PageRank algorithm worked well on partial data of our data set.

The last algorithm we implemented was Dijkstra's algorithm, used to find the shortest path. We used the algorithm and functions we learned from lectures. We first found the adjacent airports of the starting airport, choosing the shortest-distanced adjacent airports, and also making sure we don't accidentally create a circle. The result will be a vector of airports that are on the shortest path from the starting airport to the destination. We also built the test case for the Dijkstra's algorithm to test its functionality.

After we finished algorithms in our project, we created several command lines for our users. When users typed in the ID of the starting airport and destination, our algorithm will automatically print out the shortest distance between these two airports. Also, users can successfully find the most top important airports by simply typing a number in the terminal. Moreover, if users are interested in one airport and its adjacent airports, our algorithms can print the full adjacent matrix to users. Overall, we successfully finished our project and answered all questions in our proposal.