FP 1.2 : Project Proposal

Leading Question

Using OpenFlights – an open-source tool that provides airport, airline, and route data – our goal is to create the shortest route from a starting airport to a destination airport by utilizing Dijikstra's Algorithm, as well as identify the most relevant/connected airport in the provided data set by applying the PageRank algorithm. We will also use BFS to visit all the airports from the starting airport.

Dataset Acquisition

We will use OpenFlights.org and download the *airports.dat* dataset, which contains information regarding airports, their cities, and locations in coordinate format. We will model the dataset as a directed graph, marking each airport as vertices, and the distance between each airport as the weights.

Data Format

The source of the data come from 3 different categories: Each airport entry is either sourced from OurAirports, an open-source airport data site, legacy (Digital Aeronautical Flight Information File) or unverified user contributions. Only the data that is verified for high quality is included in the dataset. The input format of the dataset is .dat format, a conventional data file that contains specific information about the program used to create it. The size of the dataset is 14110 entries, one for each airport. If there are no limitations to data storage size or speed of the program, we intend to use the entire dataset- However, if we run into problems with storage space or speed, we will define a subset of the data to only include airports in major first-world countries.

Data Correction

When we parse the data, we will filter the data to exclude all incomplete data entries, which include all entries missing fields or with incorrect data types. We will also exclude any invalid entries, such as those with impossible latitude/longitude or timezone numerical values. If, when parsing the data, there is any error or exception that occurs because of that specific entry, we would skip over that entry and continue parsing.

Data Storage

We would initially store our data in the Github repository and later as a vector of nodes, each representing a different airport with fields about its information. We would later create an adjacency matrix with entries as weights representing the distance between each airport. Lastly, after solving for the shortest distance between 2 given airports, we would represent this route through a vector of nodes. Overall, because we need an adjacency matrix and a vector with all nodes, the Big O storage cost of our dataset would be O(n^2) because of the nxn adjacency matrix.

Algorithm

We will use a DFS traversal to traverse our graph structure and visit the nodes that represent airports. We will be using Dijikstra's algorithm to solve for the shortest path between airports. We also want to implement the PageRank algorithm to find the most important airports around the world.

Function inputs: Since we are using a graph algorithm, we would need to convert our dataset into a graph represented by a weighted adjacency matrix of nodes, each representing an airport. The first input into Dijikstra's algorithm would be the source node in question from the graph storing all the airport nodes, and the second input would be the adjacency matrix. For the PageRank algorithm implementation, the first input would be a node from the graph storing all the airport nodes, and the second input would be setting the range of the airports that will make up the adjacency matrix. Using these two inputs, the implementation will first create the adjacency matrix and then use C++'s linear algebra libraries to solve for the importance of airports.

Function output: The output of the Dijikstra's algorithm will be stored in a map containing the shortest paths from the source node to all different destination nodes; However, we will only return the vector containing nodes representing the shortest path between a given start and single end destination airport. We will also print this vector to output to show the solution. On the other hand, the output of PageRank algorithm will be a map of nodes mapping from each airport (node) to its PageRank value.

Function efficiency: The Big O time efficiency of our Dijikstra's algorithm is O(ElogV), where V is the number of vertices and E is the total number of edges. On the other hand, the Big O of the Page Rank algorithm is O(n+m), where n is the number of nodes and m is the number of edges. For memory, the Big O estimate is O(n^2) for the adjacency matrix.

Timeline

November 11: Parse through data set and create skeleton graph with airports as nodes,
November 18: Calculate distances and assign weights, finish converting data to graph completely; check with tests; Implement BFS and check with tests
November 25: Implement Dijikstra's Algorithm, verify correctness using tests, Implement PageRank
December 2: Finalize PageRank, verify correctness by passing tests
December 8: Finish Final Project Deliverables and Presentation
–