Final Project Proposal

Question

Our aim is to find the shortest path it would take to visit the 40 of the most popular cities to visit/travel throughout the world. We specifically selected 40 cities based on their population, tourist population, and airport traffic numbers. (https://docs.google.com/spreadsheets/d/10qROWMp9avkBSwOBKGd6O9PxxSMO2SQRJCa5j3WpNPQ/edit?usp=sharing).

We set up a graph where each node represents the selected cities and each edge represents the flight time between two cities. We are creating a scenario where the passenger at hand does not want to spend more than 12 hours (720 minutes) on a flight (essentially no overnight flights). The passenger inputs their desired starting city and ending city, and in return will get a mapped output of the shortest path between them based on optimal predicted flight times. The mapped output will highlight the shortest path given by the algorithm's output on a world map.

Dataset Acquisition and Processing

We will be using the airport data from the open flights airport database to get relevant information about the airports in the 40 cities, such as airport, city, country, continent, IATA code, latitude, and longitude. We will be filtering out the necessary airports from the list of thousands of airports found on the open flights database. To find data on flight times and routes between cities we will be manually inputting data from https://www.flightsfrom.com/ into a separate database, adding information about the route between two cities simply including: departure city, flight time in minutes, and destination city.

Sample airport data: "London Heathrow Airport", "London", "United Kingdom", "Europe", "LHR", 51.4706, -0.461941
Sample Route Data for shortest path: "London", 75, "Paris" / "Paris", 65, "London" (directed paths between cities)

Our graph will be created by each airport representing a node/vertex and each route from one city to another will be directional edges

Algorithms

Traversals(BFS/DFS) are used to decide the order of updating distance of neighboring cities. By analyzing, the traversal method chosen doesn't affect time complexity of the algorithm or the result.

Dijkstra's Algorithm decides the minimum path from one predetermined starting point to any of the other 40 cities. From the starting city, each neighbor has a cumulative distance from it's previous visited site. When a shorter value presents, the current distance is updated. It has a time complexity of O(V^2), V representing vertices. In this scenario, (40 cities) vertices and (Duration of a flight between those cities) edges. Dijkstra's Algorithm gives us the shortest path connected by flights between two cities without restriction on the number of places visited.

Dijkstra's Input: 2D array with directed paths representing routes from one city to another.

Output: List of cities visited, order of routes taken, flight time for each of the routes, and total time.

We will be using the output generated from the Djikstra's algorithm to graphically map the shortest path between the two selected cities. We will be creating the graphical output in the form of a gif, initially highlighting the start and end points and generating the routes between the calculated paths proportional to the flight time it takes to actually travel that path. This entire graph will be set on top of the world map that shows the locations of all 40 airports in the database.

Timeline

Week 1(11/8-11/14) :

- Clean the airport data, cutting out extraneous data points and columns from the openflights database
- Create the undirected and directed databases from the FlightsFrom data

Week 2 (11/22- 11/28 Thanksgiving):

- Finish implementing the Djikstra algorithm and use basic tests to ensure they are working
- Start working on the gif of the world map with points indicating locations of airports

Week 3 (11/29 - 12/5):

- Test and improve the code and algorithms running test on the full sets of data
- Work on adding paths atop the world map to display the shortest path

Week 4/Final Week (12/6-12/13):

- Finalize all presentation materials, report, final video, and presentation, conduct extreme last minute testing to make sure the code works as expected