1. **Leading Question:**

   We decided to make use of the OpenFlights dataset in our final project. The dataset we chose provides information related to airports and routes, which supports us in computing the travel time of all possible routes between departure and arrival to generate the most optimal one. The travel time will be calculated depending on the distance and the number of stops during the route. The project will develop a digital map that provides a virtual demo of the optimal flight between departure and arrival.

2. **Dataset Acquisition and Processing:**
   - **Data format:**

     The dataset used in the project is the airport information and routes information from the OpenFlight database.

     1. The airport dataset contains the airlines' ID, the airport's name, departure, arrival, longitude, and latitude. The information for each is separated by a comma.
     2. The routes dataset provides information on the airline, airline ID, departure airport, and destination airports. At the same time, this dataset includes the number of stops on the specific flight.

   - **Data Correction**:

     We will take a random sample of data and ensure they are not out of bounds of the data. Also, we will randomly check if the coordinates of airports and IDs in the dataset are consistent with the real situation. There may be some rows that have missing data entries. We will try to correct it from information online if only one or two of the elements are missing and remove it from the dataset when half of the information is missing since the data may be causing bias and waste a lot of time on it.

   - **Data Storage**:

We plan to use a weighted and connected graph, as each node will present an airport. Each edge will be a single route of all the flight routes. At worst, the total storage cost for the dataset will be O(n log n).

3. **Graph Algorithms**
    - **BFS:**

      The function input for the BFS algorithm would be the source airport (departure/start). This algorithm will have no output because it is used for iterating through all possible airports. The efficiency for this algorithm would be O(n log n).

    - **Dijkstra's Algorithm:**

      The function input would be the start (source airport) and destination airport. The output will be the shortest path between these two locations. Considering n as the num of nodes between two locations(node) and e as the edge between them, the efficiency for this algorithm will be O((n + e) log n).

    - **Betweenness centrality:**

      Given the graph's vertex (an airport), our algorithm will output how many shortest paths pass the vertex. We expect the efficiency of our algorithm to be approximately the same as our finding shortest path algorithm.

4. **Timeline**
    - **Week 1(10.28-11.6):**

      Complete proposal. Convert the locations in the dataset to valid coordinates, and create a class to store data for furthering analysis.

    - **Week 2(11.7-11.13):**

      Build graph based on the result of the first week.

    - **Week 3 & 4(11.14-11.27):**

      Implement BFS and start with the Dijkstra's Algorithm

    - **Week 5(11.28-12.4):**

Betweenness centrality

- **Week 6(12.4-before 12.8.):**

  finish the project and visualization