## Leading Question

Which airports are the most important in planning a trip? We will be using Dijkstra's algorithm as our covered algorithm to determine the shortest path between two selected airports and Tarjan's strongly connected components algorithm to determine the strongly connected components of the dataset. By identifying strongly connected components, we can identify possible groups of airports for roundtrips.

This will help us see what is the fastest way to get travel between two points and help us identify possible routes for roundtrips..

Overall, with this data we will be able to improve the ease of planning large trips..

## Dataset Acquisition and Processing

We will be using the airports and routes datasets from OpenFlights. The dataset is provided in the format of comma separated values and we will be using airports.dat and routes.dat.

Each entry in **airports.dat** includes Airport ID, Name, City, Country, Latitude, Longitude, Altitude, Timezone, Daylight savings time, Database Time zone, and Type. Each entry in **routes.dat** represents the routes provided by each airline. This dataset includes Airline, Airline ID, Source Airport, Destination Airport, Source Airport ID, Destination Airport ID, Codeshare, Stops, and plane type.

The nodes within our graph will represent various airports and the directed edges will represent the routes which are the outgoing/incoming flights from that airport. The direction of the edges on our graph will be determined by the source and destination airport IDs from the routes.dat dataset. Using the airport data, we will calculate distance between nodes and use that as the weights for the routes on our graph.

Dataset link: https://openflights.org/data.html

## Graph Algorithms

We will use DFS as our traversal algorithm and Dijkstra's Algorithm as our covered algorithm. Dijkstra's Algorithm will be used to determine the shortest path between two input airports (start and destination). Depth first search will be used as a helper function in our implementation of Tarjan's algorithm as well as for any applications that require a traversal of the dataset. For our complex algorithm, we will be implementing the Tarjan algorithm to identify strongly connected components.

## Algorithm Runtimes

In a graph of (V) vertices and (E) edges, DFS traversal algorithms have a total runtime of O(V+E).

In a graph of (V) vertices and (E) edges, Dijkstra's algorithm has a total runtime of O((V + E) log(V)).

In a graph of (E) edges and (V) vertices, Tarjan's algorithm has a worst case runtime of O(V + E).

## Timeline

Week 1 (Nov 8 - Nov 14): Set up git on everyone’s system Study the dataset and implementation of selected algorithms Download dataset Convert the dataset into usable data Start working on traversal.

Week 2 (Nov 15 - Nov 21): Work on the implementation of the selected traversal method, finish by end of week Begin work on implementation of our covered algorithm Implement the make file Work on producing a test case suite.

Fall Break (Nov 22- Nov 28): If necessary, work on remaining details necessary for the mid-project check-in.

Nov 29: Mid-Project Check In Successful traversal of the dataset or test cases Successful shortest path identification in the dataset or test cases Consult w/ TA regarding any difficulties in implementation or suggestions.

Week 4 (Nov 29 - Dec 5): Work on the implementation of betweenness centrality algorithm Project functional by end of this week.

Week 5 (Dec 6 - Dec 12): Write up a report. Make a presentation. Film the presentation.

Dec 13: Final Project Deliverables Submission.