Project Goals

Team Members:

- Panav Munshi: **pmunshi2**
- Rohan Batra: **rohanb4**
- Anuraag Agarwal: **anuraag3**
- Abhay Narayanan: **abhayn2**

Dataset Link:

- http://stat-computing.org/dataexpo/2009/the-data.html
- https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/HG7NV7

Note: The aforementioned links point to the same dataset.

Description:

We will be using the above dataset and extracting domestic flight data within the USA from them in the form of CSV files. These files contain data on the airline, the flight code, its routes, its departure and arrival timings, the elapsed time, the origin, and the destination. We will be applying this data to create a program that works to allow a user to plan a trip between two airports within the US. This will be implemented using two conditions:

1. Finding the path with the *least number of flights* needed to reach the destination
2. Finding the path to the destination which traverses the *least distance*.

Architecture:

As the dataset has many fields, which includes the name of the airports, the distance between airports, elapsed time, etc., we believe that arranging the data in a **graph structure** will enable us to look for shortest paths between the connected components. For this problem, we implement three well known algorithms to provide solutions for a traveler (the user) in planning a journey. We use the **Breadth-First Search (BFS) traversal** to find the path from a source point to the destination point which goes through the least number of edges. As an edge corresponds to a flight between two points, it is clear that the BFS algorithm will be used to solve the problem posed by the condition 1. The other algorithms used will be the **A* search** and **Dijkstra** algorithms, and will be implemented using two separate methods to find the path with the least distance between a source point and the destination point. This approach will use the data set and our built weighted graph (where each edge has an associated distance) in tandem to solve the problem posed by the second condition at the end of the previous paragraph. We will also further extend the A* search algorithm to implement the Landmark Path algorithm, which is used when there is a desired intermediary stopover in between the source and destination points. The Landmark Path will use the A* search algorithm at its backend, and implement it twice. These algorithms will solve the problem posed by condition 2.