Adrianne Sun (ajsun2)
Justin Song (js52)
Walker Rickord (walkerr2)
Daniel Kaufman (dbk3)
CS 225: Data Structures and Algorithms

Final Project Proposal

1. **Leading Question:**

Using the OpenFlights open source data, we hope to develop at least two different algorithms to calculate the shortest path for any given route or transportation method. Overall, the project will serve as a general search tool that returns the shortest path given two points and a transportation type.

2. **Dataset Acquisition and Processing:**

For this project, the airports extended version of the dataset provided on the OpenFlights websites will be used. The dataset, in CSV format, will be downloaded from the website, stored on GitHub (and possibly locally), and processed with a simple text editor (if necessary). If the text editor is insufficient, Python or some other convenient data processing language can be utilized. The dataset currently includes a myriad of different things, all of which are not necessary. For the project, the most likely data types that will be used are as follows: Airport ID, Name, City, Country, Latitude, Longitude, (transportation) Type, and Active. For the purposes of this project, only airports, ferries, bus stations, and train stations are going to be considered. To parse out data entries that may contain errors, the data set can be filtered out using the Active tag. If the type is unknown but the transportation type is clear in the name, the entry may be kept and edited accordingly. In addition, entries with any incomplete fields will be discarded. Any other errors will be resolved in a similar fashion.

3. **Graph Algorithms:**

On a high level, Dijkstra's Algorithm will be used for the shortest path algorithm. For the shortest path algorithms, the primary inputs will be the two locations and the transportation method the user wishes to find a route between. A secondary user-provided input can specify the format of the output. The expected output would be a sequence representing the shortest path printed in the terminal. Said shortest path could be a list of transportation hub (e.g. airport) identifiers such as the airport ID, the name of the airport, or the location of the airport. For the graph traversal, DFS and Iterative Deepening DFS will most likely be implemented. The input will primarily be a starting point and the output will be a sequence printed to the terminal in a format similar to that of the algorithms. Overall, the algorithms shall be implemented as efficiently as possible. The estimated big-O runtime of the DFS traversal will be O(V+E) where

V is the number of nodes and E is the number of edges. The estimated big-O runtime of Dijkstra's Algorithm is O((V+E) log(V)). The estimated big-O runtime of IDDFS is O(b^d) where b is the branching factor and d is the depth of the goal. The edge weight will be the haversine distance between two locations using latitude and longitude.

4. **Timeline:**

The following is a list of tasks and the approximate (optimistic) date at which we plan to finish the tasks. For reference:

- Each subteam will lead one of algorithms to lighten the load
- The data processing and creation of any classes will be done in parallel
- Three day leeway given at end for potential obstacles
- If ahead on schedule, the next task shall be worked on immediately rather than waiting

Schedule:

- Data processing (11/10)
- "Graph Class" creation (11/13)
- DFS (Traversal) (11/19) (i.e. right before fall break begins)
- Iterative Deepening DFS (Uncovered Traversal) (11/30) (i.e. 2 days after end of fall break)
- Dijkstra's Algorithm (11/30) (i.e. 2 days after end of fall break)
- Written report (12/10)
- Final video presentation (12/10)