CS 225 Final Project Goals

Saket Vissapragada, Siddhartha Adatrao, Sruthi Kilari, Shreya Sharma

November 18, 2020

1 Dataset

The OpenFlight data set contains information about airports: name, city, latitude/longitude, and timezone for over 10,000 airports. The source of the data set is `https://github.com/jpatokal/openflights` . If this data set ends up not fitting our goals, we will pivot our project to use the Amazon data set from `http://snap.stanford.edu/data/amazon0302.html`. This data set contains nodes that represent items and edges between items that are known to be "co-purchased". We will use the same traversal and graph algorithms to find shortest paths to certain "key items" from a bought item.

1.1 Project Goal

The goal of this project is to detect the shortest flight path from a start point to an end point with certain restrictions regarding the maximum distance that can be traveled in one path between two airports to create situations in which layover is needed. To achieve, this goal, we will be incorporating BFS traversal, Dijkstra's Algorithm, and Landmark Path Algorithm.

2 Traversal Method

2.1 BFS (Breadth First Search)

We will incorporate BFS traversal to traverse our graph and find the most proximal airport nodes to a given root node. Our BFS traversal will ignore the weights on our graph and only consider the nodes (airports). We want to store the data of our BFS traversal in a linear data structure for easy ordering and retrieval of data. Our end-goal of using a BFS traversal is to properly format and output a sorted list of the most proximal airport nodes to specified root node

3 Graph Algorithms

3.1 Dijkstra's Algorithm (Covered Algorithm)

We will be incorporating Djikstra's Algorithm for the purposes of finding the shortest path between any two given airports. To do this, we will define the weights of our graph to be the distances between the connected nodes (the different airports).

3.2 Landmark Park Algorithm (Complex/Uncovered Algorithm)

We will incorporate the Landmark Path Algorithm in order to determine the shortest path between any two airports, given a third airport that must be visited in between. To do this, we will define the weights of our graph to be the distances between the connected nodes (the different airports).

4 Testing

4.1 Testing Traversal Method

We will calculate five paths for three cohorts of distances (within 500 miles, 5000 miles, and within the world). Then we will see whether our Breadth First Search traversal returns the same calculation for each cohort.

4.2 Testing Graph Algorithms

We will make test cases to ensure that all airports can be reached by chosen landmarks by associating a landmark by region based on latitude/longitude. We will calculate five paths for three cohorts of distances (within 500 miles, 5000 miles, and within the world). Then we will see whether our Breadth First Search traversal returns the same calculation for each cohort.