Jaywoo Jo
Holly Lee
Hemank Kohli

CS225 Updated Project Goals

*** Goals updated after our meeting with our Mentor, Zijie Lu, on (12/04)*

Initially we were interested in the Social Circles: Facebook data set using Stanford's SNAP. However, having to change the file contents to be readable seemed to be an unnecessary challenge, so we decided to use the OpenFlights (https://openflights.org/data.html) data set. We spoke with our mentor during our mid-project check in meeting and he advised us to update our project goals.

For our project, as the assignment describes, we will be loading a data file as a graph and use that graph to run algorithms of interest to the data set. For our project, we specifically decided to use the airports.dat and routes.dat files on the website. The airports.dat file contains a list of over 10,000 airports with data items such as the airport name and country, the 3-digit IATA code, and longitude and latitude values. The routes.dat file contains flight paths from one airport to another (for example, ICN to SIN). It's important to note that while all major airports have a 3-digit IATA code, for example "ICN" for Incheon Airport in Korea, some small airports don't have this 3-digit code, so we decided to not keep track of these values since the routes.dat files don't show routes going through these small airports.

Our first goal will be to find a way to read the data we need from the files., and our second goal will then be to construct the actual graph. We will be reading these data files to create a graph where vertices are the airports (the string of 3-digit IATA codes), directed edges represent flight paths, and edge weights represent the distance in kilometers. In order to calculate this weight, we will be using the Haversine formula that calculates distance using longitude/latitude values. Additionally, we will be using the graph implementation provided to us during CS225 labs.

Our second goal will then be to run our algorithms of interest. It needs to be one of the traversals covered in class, and two other options from the complex/uncovered options we were provided with. Firstly, we will be running a BFS traversal and outputting visited vertices; secondly, we will find the shortest path using Dijkstra's Algorithm; and thirdly, we will find strongly connected components.

Our last goal, and most important goal, will be to create test cases along the way and find effective ways to heavily test our algorithms to make sure they work.