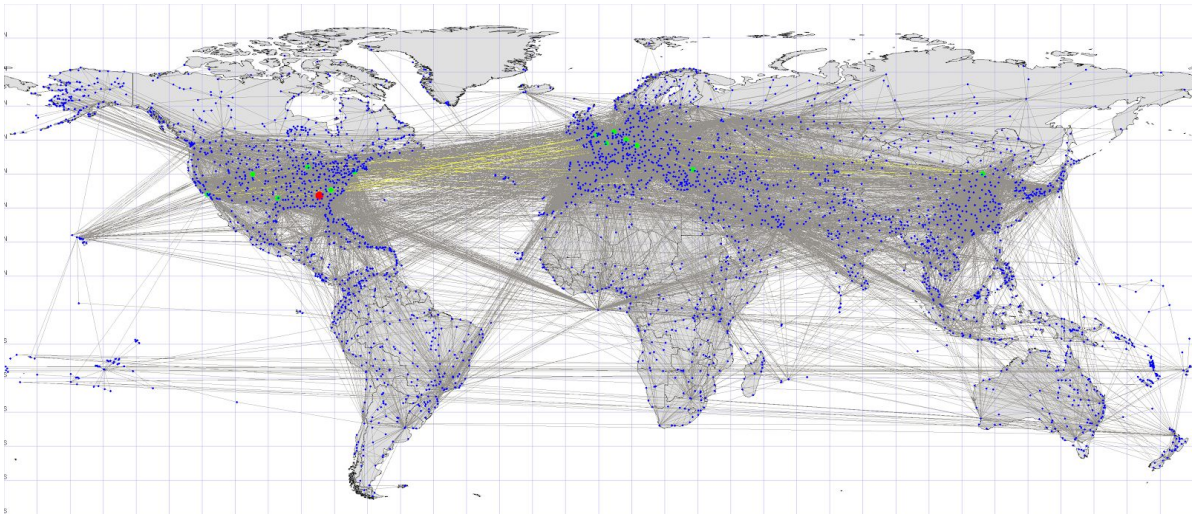
Final Project Results

The data set was chosen first for the project, which is the openflights dataset. We divided our tasks into two different topics to fully utilize the dataset. The first task was to use the route data to discover the busiest or the most important airport in the world. The second task was to use the route data to run a breadth first search. Lastly, the third task was using the airport locations and route data to calculate the landmark path to between specified airports.

For calculating the busiest airports, among all the centralities, we chose degree centrality because it calculated the weight on the vertices and edges directly by the number of routes itself. It was appropriate to calculate the most important airport with the highest number of routes. At first, the weights were stored inside the edges, but soon we created a variable that stores the vertices' weight separately in the map. To show these weights, we chose to draw them on the world map. Airports' location was extracted from the airports.dat file. The latitude and the longitude was used to calculate the location on the worldmap. Different colors and the radius was used to show the difference in weight among the airports. As a result, we found out that the Hartsfield-Jackson Atlanta airport(ATL) was the busiest airport in the world. In fact, there was no airport that had a weight close to ATL.



In the figure above, the color of the circles depends on the size of the weight. Red indicates that the weight is greater than two-third of the ATL weight, green indicates that the weight is greater than one-third of the ATL weight, and blue indicates that the weight is smaller than one-third of the ATL weight. ATL is the only red vertex that exists on the map.

A breadth first search is a standard search through a graph, where the closest vertices are visited first. In order to run BFS on the airport route data, the routes first had to be converted in to a graph. This graph is different from the one used for calculating the busiest airports in that this graph was unweighted and undirected. Using a standard BFS algorithm in which a queue is used to store and decide which vertex to visit next, we were able to return a full breadth first traversal path through all of the airport routes given. We also returned a list of all the edges and whether they were a discovery edge or a cross edge.

Path

```
AER
NSK
VKO
LBD
OMS
IST
DME
KIV
KRR
SVO
EVN
TAS
KJA
LED
TZX
```

Edges

```
AER<-->NSK DISCOVERY
AER<-->VKO DISCOVERY
AER<-->LBD DISCOVERY
AER<-->OMS DISCOVERY
AER<-->IST DISCOVERY
AER<-->DME DISCOVERY
AER<-->KIV DISCOVERY
AER<-->KRR DISCOVERY
AER<-->SVO DISCOVERY
AER<-->EVN DISCOVERY
AER<-->TAS DISCOVERY
AER<-->KJA DISCOVERY
AER<-->LED DISCOVERY
AER<-->TZX DISCOVERY
AER<-->SVX DISCOVERY
```

In the two snippets above, you can see the formatting of the results for the first 15 entries in the returned path file and the returned edges file. The traversal starts at the first airport given in the route data and traverses through every single unique airport in the data.

Lastly, we implemented a function that would calculate a landmark path through given parameters. Landmark path was intended to give results of the shortest path from a to b through c. As intended, the landmark path that we implemented takes 3 entries (source, landmark, destination). In the landmark path class, we implemented the graph to use the calculated distances of each flight route as the weight of the edges. This way, our landmark path function calculates not just the route with the fewest number of stops, but the route with the actual shortest travel distance. Using the getResult method and printPath method, we were able to print out the IATA codes of all the airports one would have to go through in order to go from the source airport to the destination airport, through the landmark. When the list of airports gets printed out, the source, landmark and destination are marked in a different color. Moreover, if any of the

airports in the input do not exist in the route data, an error message would get printed out.

```
--------------------------------------------------------------------------------
Shortest path from YAM-->LPP-->VXO
YAM
YYZ
DUB
STN
NRN
LPP
NRN
STN
AMS
VXO
--------------------------------------------------------------------------------
```

In the example above, the source airport chosen was YAM (Sault Ste. Marie Airport), the landmark was chosen as LPP (Lappeenranta Airport), and the destination airport was chosen as VXO (Växjö Småland Airport). These are pretty small and obscure airports chosen at random, thus the shortest route to get from source to the destination through the landmark is actually not that short.

In conclusion, we were successfully able to implement all of the goals we had for this project, and in the process we were able to learn a lot about world flight patterns, as well as create a useful tool that we may possibly use to in the future to calculate the shortest flight path to a destination we may want to visit.