Development

The First Week (Nov 18 - Nov 22; counting from the due date of Project Goals)

At the November 19 meeting, we created several files for the basic structure of our project, such as vertex.{h, cpp} (the vertex of the graph), edge.{h, cpp} (the edge of the graph), get_data.{h, cpp} (the class that transforms the data files into vectors of vertices and edges), the data files, and the Makefile. We implemented a version of get_data.cpp that successfully can receive the data and return a vector of vertices. At the November 21 meeting, we applied the final project demo Makefile template to improve our Makefile. We use the similarity to implement the function that returns the vector of edges. It worked as expected when we tested it in main. We added Doxygen style documentation to some files and references.

The Second Week (Nov 23 - Nov 29)

We applied the adjacency list implementation to our graph. The container of vertices is a simple hash table that hashes based on the unique airport id. Each vertex has a deque of edges. The container of the full list of edges is also a deque. For the traversal part of our goals, we have implemented the BFS algorithm, while it has not been tested. We plan to write test cases for it next week. For the clustering finder algorithm, we have written it in the way that it is able to return the locations (vertices) that are not connected to the major part of the world, and its test will also be done next week. We have also made significant progress in drawing on the world map. In the Edge_coloring class, we are able to draw an airline between two given airport's location in latitude and longitude. We are going to plot the airport on the world map using different colors by the weight of each vertex.

The Third Week (Nov 30 - Dec 6)

For the algorithm part of our project, we have tested our BFS as planned and are on our way to complete our Dijkstra's algorithm to find the shortest path between two airports. We are implementing our own priority queue as we cannot find a way to fit the C++ priority queue implementation into ours. Our clustering finder algorithm provides more choices for users. We have some improvements in the visual effect of our graph drawing. For instance, the airports that are more like a traffic center (ie. more edges connected to it) have more eye-catching color and illuminance. Also, the airline drawing method is upgraded such that the airlines that fly across the Pacific Ocean can be drawn (from the wester airport to the left graph boundary, and then from the right graph boundary to the easter airport).

The Fourth Week (Dec 7 - Dec 11)

This week, we spent most of our time debugging and testing our classes and algorithms. We generated three pictures using our line drawing algorithm and shortest path algorithm. “draw_all_airports.png” highlighted the airports that have more incident edges on the world map. The airports that do not have incident edges are colored with red. The more incident airlines it has, the hue is closer to 240(purple). draw_all_airline.png shows all airlines in airlines.dat. The shortest_path.png shows the shortest path between two airports and sketches the path in white dots. We create a readme.md and post these three pictures on it and introduce the usage of our project. Then we wrote a RESULT document to explain the progress we achieved in our final project.