Week of December 6 - December 12

In this week, we implemented a separate BFS algorithm upon feedback from our mentor that implementing BFS and Dijkstra may not be acceptable. Additionally, we created test cases for our code testing our graph constructor, BFS algorithm, shortest path algorithm, and the final graphical output. During this time, we experienced some bugs which included receiving segmentation faults from empty lines in the datasets. Also, our line drawn on the graph is really sketchy and can only go from left to right. We successfully overcame and fixed the bugs for our final presentation. Accounting for user input was also implemented into our project. Now, users can provide a source and destination airport ID and we will provide the shortest path between them as well as a graphical output to a PNG image. Finally, we finished any extraneous files needed such as the README.md and also completed our final report and presentation.

Week of November 29 - December 5

For this week, we started with some optimizations for our Dijkstra's algorithm which included a minimal improvement in runtime as well an edge case if there was no route between a source and destination. We also chose to not implement A* search, because we had implemented a shortest path algorithm using Dijkstra. Instead, we chose to implement the graphical output of our graph using the cs225 files and write our own algorithm to draw a line between two pixels, which will represent our vertices (airports). We also decided to write some test cases for our BFS and Shortest Path algorithm as well as our graphical output implementation.

Week of November 22-28

In this week, we have successfully implemented our graph creation using the airports and routes databases from OpenFlights. To process the datasets, we read each line from the respective dataset using the template from the AMA slides and stored them in a vector<vector<string>> to later insert them as vertices and edges to our graph. To set the vertices, we used data from the airports dataset which included information such as the airport ID, longitude, latitude, and airport name. For the edges, we used the data from the routes dataset to set variables such as the source airport, destination airport, and the airline traveled between them. Also, we calculated the distance between source and destination airport using latitude/longitude and set it as our weight. To find the shortest path between a source and destination airport, we used BFS traversal and a priority queue (min heap) to successfully implement Djikstra's algorithm.