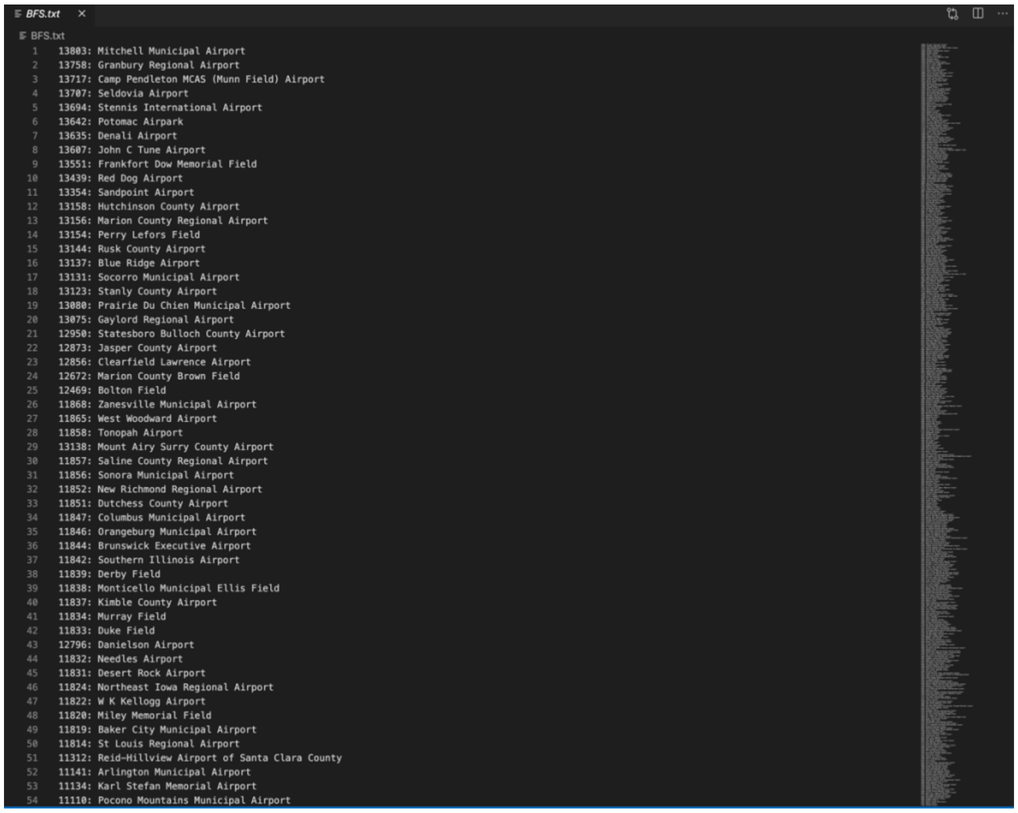
CS225 Final Project Result

Our CS225 final project strictly followed our goals: DFS traversals for traversing the whole graph (visiting nodes in graph in depth order), Floyd-Warshall algorithm for calculating shortest path between two points (in our chosen dataset they are airports, information provided by OpenFlight), airline routes data visualization on a world map. We successfully accomplished our goals and, I will briefly illustrate the outcome and our super convenient and friendly user interface. After resulting our executable in main.cpp, there will be specific instructions for choosing algorithms.

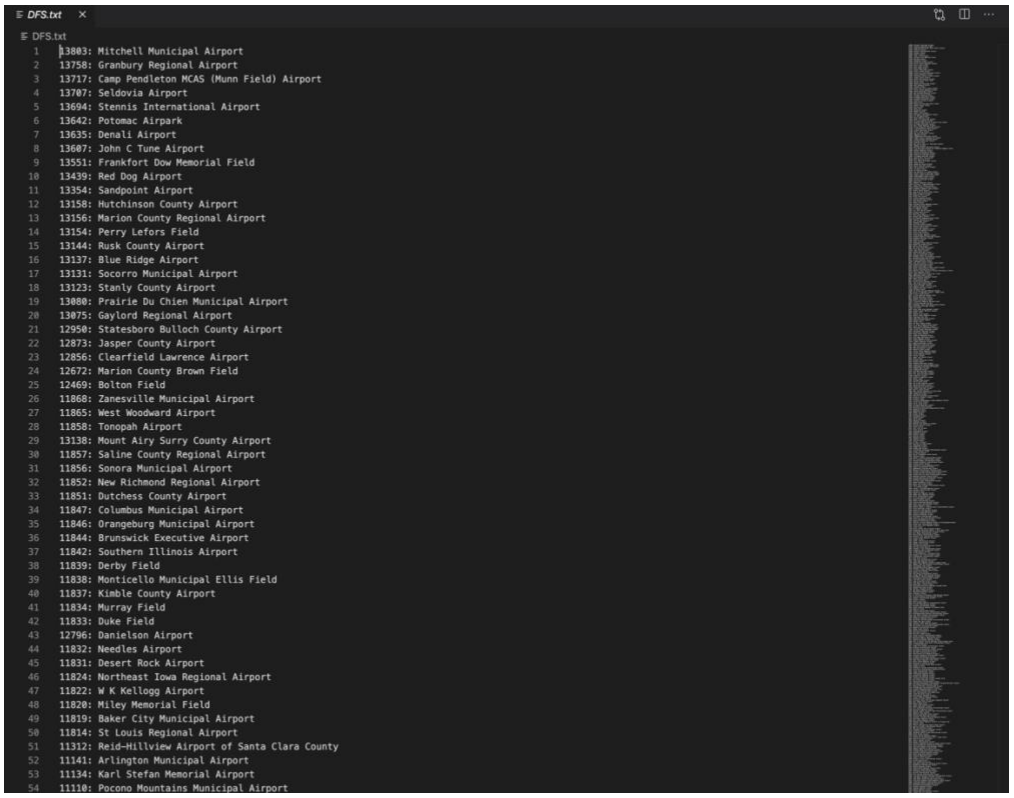
The available algorithms are: DFS/BFS traversal; Shortest path between two points using distance as weight; Project onto map based on Openflight dataset", as an option visualization for users to use. By choosing BFS or DFS, the program will respectively run each traversal algorithm and save the result to the same folder as "BFS.txt" or "DFS.txt" with a list of vertices based on the traversal order; choosing shortest path would ask the program to calculate the distance matrix and the path matrix, then the program is able to demonstrate the shortest path as a list between any source and destination included in the dataset entered by users; by choosing visualization, program will result two PNGs: "world_map_with_airports.png", and "world_map_with_airports_and_routes.png", as two pictures with airports as nodes on the world map as well as the routes as edges.

We did have discoveries while proceeding our final project. In particular, we found mapping edges as lines that of routes to the picture (world_map_with_airports_and_routes.png) surprisingly challenging. To solve this problem, we searched about the Bresenham's Line Algorithm and implemented it to drawing lines for edges. Beyond this, we needed to divide the line situations into cases as to making sure that every edge was mapped correctly. Moreover, we also found many disconnected airports and lots of airlines with unrecorded source or destination airports. That’s because the two dataset we used were collected in different years. We modifies several our data processing functions to avoid problems caused by this situation.

Traversal Result:

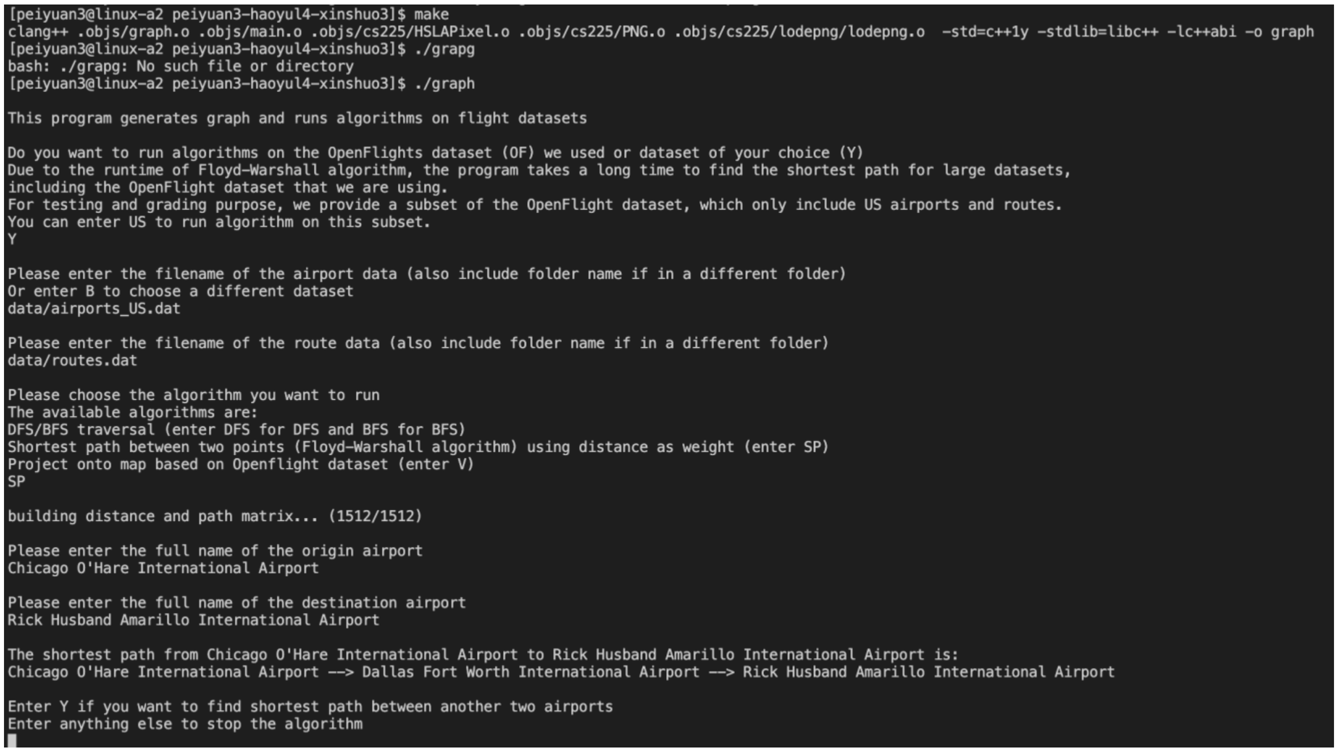

BFS.txt

```
13803: Mitchell Municipal Airport
13758: Granbury Regional Airport
13717: Camp Pendleton MCAS (Munn Field) Airport
13707: Seldovia Airport
13694: Stennis International Airport
13642: Potomac Airpark
13635: Denali Airport
13607: John C Tune Airport
13551: Frankfort Dow Memorial Field
13439: Red Dog Airport
13354: Sandpoint Airport
13158: Hutchinson County Airport
13156: Marion County Regional Airport
13154: Perry Lefors Field
13144: Rusk County Airport
13137: Blue Ridge Airport
13131: Socorro Municipal Airport
13123: Stanly County Airport
13080: Prairie Du Chien Municipal Airport
13075: Gaylord Regional Airport
12950: Statesboro Bulloch County Airport
12873: Jasper County Airport
12856: Clearfield Lawrence Airport
12672: Marion County Brown Field
12469: Bolton Field
11868: Zanesville Municipal Airport
11865: West Woodward Airport
11858: Tonopah Airport
13138: Mount Airy Surry County Airport
11857: Saline County Regional Airport
11856: Sonora Municipal Airport
11852: New Richmond Regional Airport
11851: Dutchess County Airport
11847: Columbus Municipal Airport
11846: Orangeburg Municipal Airport
11844: Brunswick Executive Airport
11842: Southern Illinois Airport
11839: Derby Field
11838: Monticello Municipal Ellis Field
11837: Kimble County Airport
11834: Murray Field
11833: Duke Field
12796: Danielson Airport
11832: Needles Airport
11831: Desert Rock Airport
11824: Northeast Iowa Regional Airport
11822: W K Kellogg Airport
11820: Miley Memorial Field
11819: Baker City Municipal Airport
11814: St Louis Regional Airport
11312: Reid-Hillview Airport of Santa Clara County
11141: Arlington Municipal Airport
11134: Karl Stefan Memorial Airport
11110: Pocono Mountains Municipal Airport
```


DFS.txt

```
13803: Mitchell Municipal Airport
13758: Granbury Regional Airport
13717: Camp Pendleton MCAS (Munn Field) Airport
13707: Seldovia Airport
13694: Stennis International Airport
13642: Potomac Airpark
13635: Denali Airport
13607: John C Tune Airport
13551: Frankfort Dow Memorial Field
13439: Red Dog Airport
13354: Sandpoint Airport
13158: Hutchinson County Airport
13156: Marion County Regional Airport
13154: Perry Lefors Field
13144: Rusk County Airport
13137: Blue Ridge Airport
13131: Socorro Municipal Airport
13123: Stanly County Airport
13080: Prairie Du Chien Municipal Airport
13075: Gaylord Regional Airport
12950: Statesboro Bulloch County Airport
12873: Jasper County Airport
12856: Clearfield Lawrence Airport
12672: Marion County Brown Field
12469: Bolton Field
11868: Zanesville Municipal Airport
11865: West Woodward Airport
11858: Tonopah Airport
13138: Mount Airy Surry County Airport
11857: Saline County Regional Airport
11856: Sonora Municipal Airport
11852: New Richmond Regional Airport
11851: Dutchess County Airport
11847: Columbus Municipal Airport
11846: Orangeburg Municipal Airport
11844: Brunswick Executive Airport
11842: Southern Illinois Airport
11839: Derby Field
11838: Monticello Municipal Ellis Field
11837: Kimble County Airport
11834: Murray Field
11833: Duke Field
12796: Danielson Airport
11832: Needles Airport
11831: Desert Rock Airport
11824: Northeast Iowa Regional Airport
11822: W K Kellogg Airport
11820: Miley Memorial Field
11819: Baker City Municipal Airport
11814: St Louis Regional Airport
11312: Reid-Hillview Airport of Santa Clara County
11141: Arlington Municipal Airport
11134: Karl Stefan Memorial Airport
11110: Pocono Mountains Municipal Airport
```

Shorted path example result:



```
[peiyuan3@linux-a2 peiyuan3-haoyul4-xinshuo3]$ make
clang++ .objs/graph.o .objs/main.o .objs/cs225/HSLAPixel.o .objs/cs225/PNG.o .objs/cs225/lodepng/lodepng.o  -std=c++1y -stdlib=libc++ -lc++abi -o graph
[peiyuan3@linux-a2 peiyuan3-haoyul4-xinshuo3]$ ./grapg
bash: ./grapg: No such file or directory
[peiyuan3@linux-a2 peiyuan3-haoyul4-xinshuo3]$ ./graph

This program generates graph and runs algorithms on flight datasets

Do you want to run algorithms on the OpenFlights dataset (OF) we used or dataset of your choice (Y)
Due to the runtime of Floyd-Warshall algorithm, the program takes a long time to find the shortest path for large datasets,
including the OpenFlight dataset that we are using.
For testing and grading purpose, we provide a subset of the OpenFlight dataset, which only include US airports and routes.
You can enter US to run algorithm on this subset.
Y

Please enter the filename of the airport data (also include folder name if in a different folder)
Or enter B to choose a different dataset
data/airports_US.dat

Please enter the filename of the route data (also include folder name if in a different folder)
data/routes.dat

Please choose the algorithm you want to run
The available algorithms are:
DFS/BFS traversal (enter DFS for DFS and BFS for BFS)
Shortest path between two points (Floyd-Warshall algorithm) using distance as weight (enter SP)
Project onto map based on Openflight dataset (enter V)
SP

building distance and path matrix... (1512/1512)

Please enter the full name of the origin airport
Chicago O'Hare International Airport

Please enter the full name of the destination airport
Rick Husband Amarillo International Airport

The shortest path from Chicago O'Hare International Airport to Rick Husband Amarillo International Airport is:
Chicago O'Hare International Airport --> Dallas Fort Worth International Airport --> Rick Husband Amarillo International Airport

Enter Y if you want to find shortest path between another two airports
Enter anything else to stop the algorithm
```

Visualization result: