CS 225: OPEN-FLIGHTS FINAL PROJECT

By: Alan Huerta Magdaleno, Juan David Campolargo, Patricia Salinas, Samuil Donchev

GOALS OF THE PROJECT

- Set up project for Compilation
- Store Open Flights data files into CSV format
- Filter through data and removing faulty datapoints
- Create Graph with airports as nodes and routes as edges
- Complete BFS
- Complete PageRank Algorithm
- Complete Dijkstra's Algorithm

DEVELOPMENT PHASE: PROJECT COMPILATION

- At the start of the project, we ran into our very first issue trying to get the project to compile and run.
- We tried different methods and had multiple people trying their hand at it with no success.
- Cmake was used to automatically create Makefiles for everything to compile smoothly.

DEVELOPMENT PHASE: DATA STORING & FILTERING

- We downloaded the data
- Converted it to CSV
- Found out that there were unmapped/missing information for some airports
- Used Excel build it functions and a macro to delete and filter bad data
- Parsed through the data
- Airport Data set was used to create nodes and Routes Data set was used to create edges.

DEVELOPMENT PHASE: GRAPHING DATA

- For the graph implementation we decided to use an adjacency matrix to improve runtime of the project. If two nodes are connected, the adjacency matrix would return the distance between the two nodes.
- If the weight is 0 then the nodes **do not** share a connection.
- The AirportToIndexMap allowed us to save all the Airports based on their airport code/ID. Once inputting an airports ID into the map, we would know where the airport appears on the adjacencyMatrix with the index returned. This allowed us to look at any airport we wanted using just the airports code/ID

DEVELOPMENT PHASE: CREATING BFS

- In the BFS we set it up so that we can start at any node given its airport code/ID or in other words our start position.
- With that, we would use our AdjacencyMatrix in conjunction with our airportToIndexMap to get our starting position and look for any possibly connected nodes.
- However, we ran into a small problem where we had nodes that weren't being visited at all even though we made sure that there were no nodes that had null codes or information.
- We later found out that some airports do not have any connections as our Routes Data set did not include those airports. Thus, Airports with 0 connections were ignored.

DEVELOPMENT PHASE: PAGERANK ALGORITHM

- When implementing the PageRank Algorithm, we knew that we wanted to have some way of ranking the popularity of an airport.
- To do this we used the number of routes as the determining factor. Therefore, the more routes connected to the current node, the more popular it was.
- We ran into an issue where the ratios for airport popularity were correct but higher than what PageRank should output. This was a logical error in the calculation that was adjusted for and ultimately, we were able to get a list of the top popular airports.

DEVELOPMENT PHASE: DIJKSTRA'S ALGORITHM

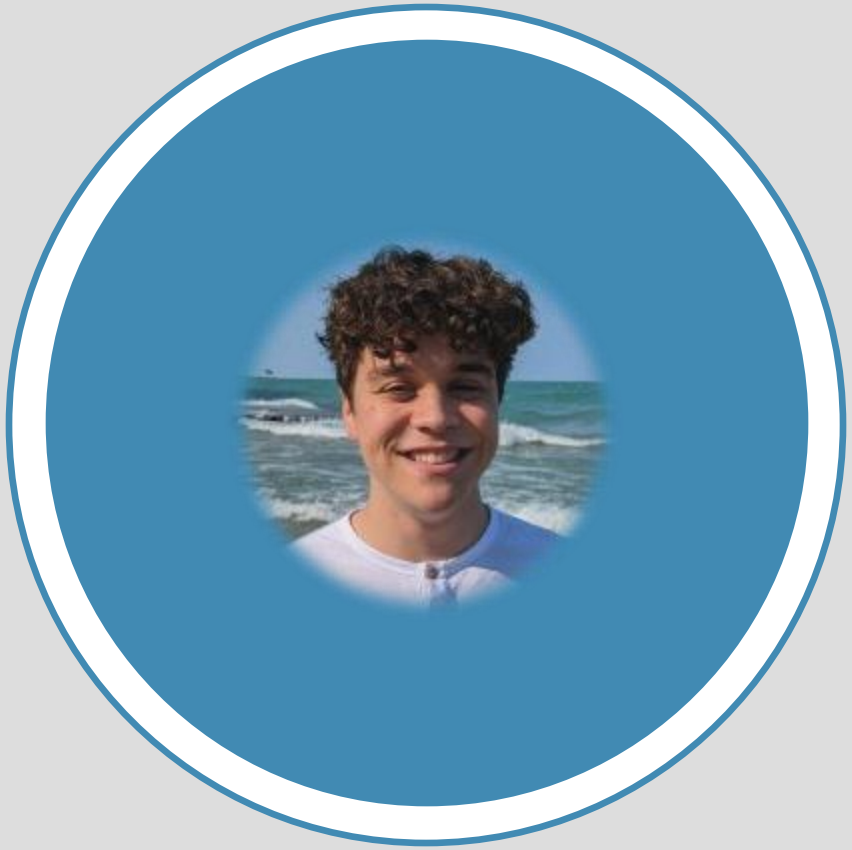
- When we created Dijkstra's Algorithm, we referenced a few online sources to see how we should go about our implementation of it.
- Once we knew what to do, we began by having it take the graph of nodes that we have, a starting airport, and a destination airport.
- At first, understanding how to implement this problem in the context of our project took some time. However, this problem was overcome by testing the algorithm on a small custom data set that allowed us to easily see how the algorithm was functioning and adjust it accordingly.

CONCLUSION

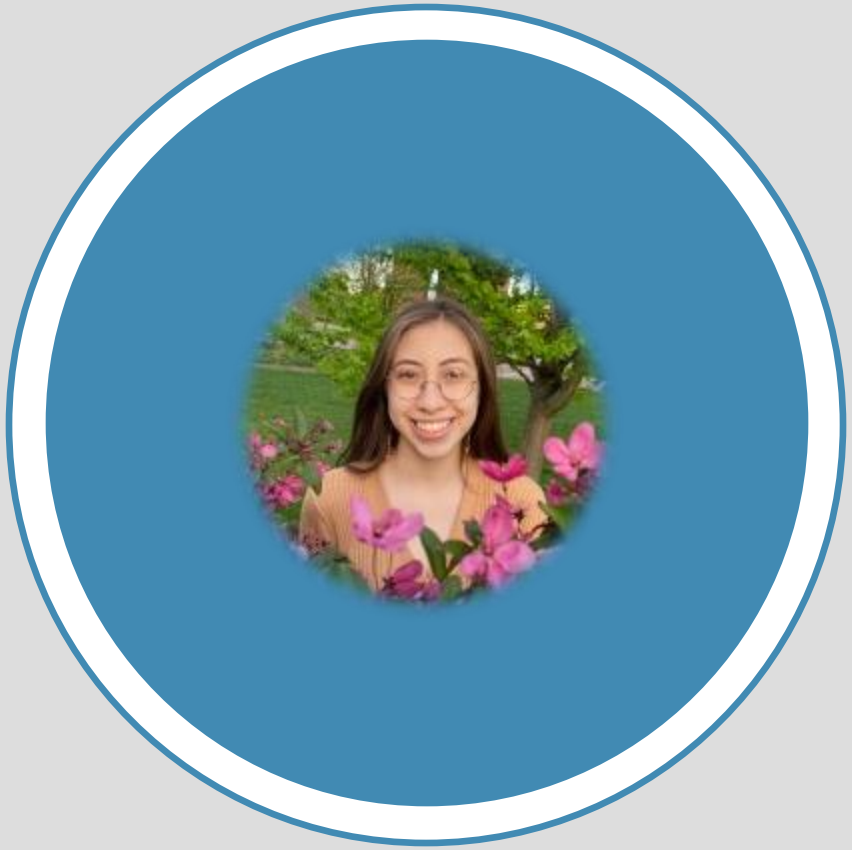
Our Question: Given an OpenFlights large dataset of airplane airports and routes, can we find the shortest route from any airport, to its closest major airport?

Overall, we managed to answer our leading question because we were able to find the shortest route from any starting airport to the wanted destination. By inputting the code of our starting airport and the code of our destination airport and the graph where all our nodes are stored, we managed to traverse the graph and find the shortest most optimal path to take from our starting point to our destination. Given the list of most popular airports provided by the PageRank algorithm, we were able to identify the closest major airport from the starting airport as well.

“From this project, I was able to learn a bit more about how our day-to-day tools like google maps work and why they're able to do the things they do so well. It helped me demystify the behind-the-scenes work put into them and understand just how complex some of the tools we have at our disposal can be. “



- “The process of breaking down a large problem into more manageable tasks and seeing everything come together at the end is very satisfying. Implementing a graph data structure, BFS, and Dijkstra has given me a lot of confidence to tackle problems in the future. Furthermore, Having a strong base to your code structure is very important as it makes implementation of future features much easier.”



- "Throughout this project, I was able to apply a variety of the concepts covered in class and see how they all come together. It was also exciting implementing an uncovered algorithm such as PageRank and understanding how it used the graph to iterate through and find the 'popularity' of each node."

- “I enjoyed learning about how to implement these algorithms to solve a problem in the world. One thing, it's to learn these algorithms and learn how they work. Another thing, it's to make them work and solve an actual problem. I greatly enjoyed working with my team because of how much they taught me. ”