# CS225 Final Project

## OpenFlights Dataset Betweenness Centrality

Group Members:
Yifei Kang (yifeik3)
Justin Nie (justinn6)
Ariana Tarafdar (arianat4)
Chengbin Zhu (cz43)

# Goals

- Dataset: from OpenFlights database, including the ‘airports’ and ‘routes’ dataset
- Objective: examine the OpenFlights dataset and determine betweenness centrality by mapping the most significant airports
- Design: directed weighted graph
- Airports were mapped as vertices, and flights between airports were drawn as directed edges. The weight of the edges were determined by the combination of distance and the number of flights passing through

# Goals

- Algorithms
  - BFS (Breadth First Search) to traverse connected components
  - Johnson’s Algorithm to calculate the shortest path from one airport to another
  - Calculate the betweenness centrality of an airport
- Summary of accomplishments
  - Complete all the proposed algorithms
  - Finished the written report
  - Created test datasets and graphs to test code
  - Generated additional visualization methods
  - Use Dijkstra’s algorithm instead of Johnson’s algorithm.

# Parsing through Datasets

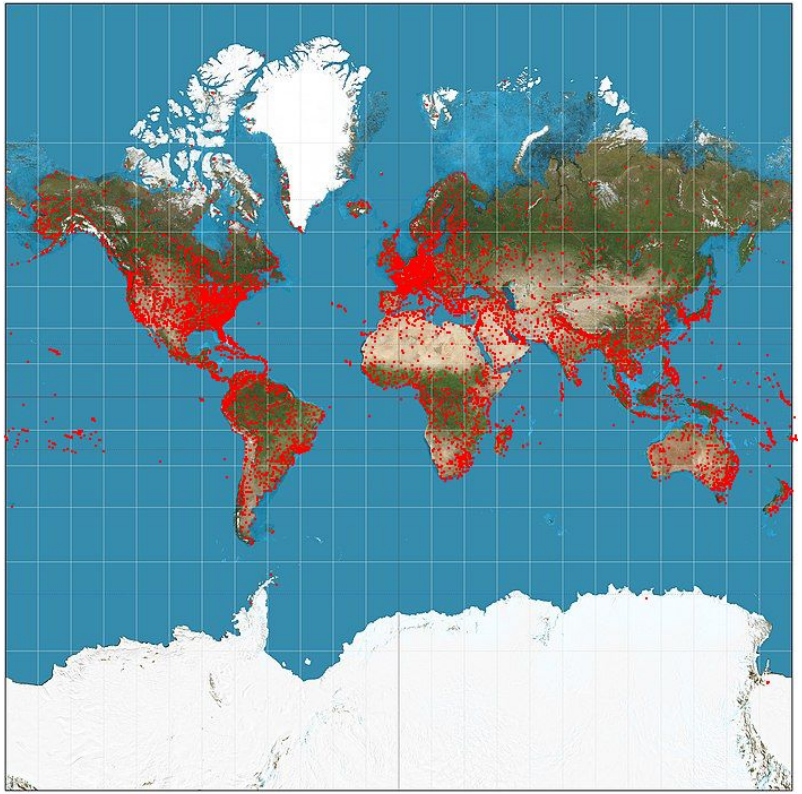
- Parsed through OpenFlights airport and routes datasets
- Parsed from Airport Dataset:
  - Vertices map - Airport IDs mapped to Airport index
  - Airports map - Airport index mapped to Pair of Longitude and Latitude
  - Convert map - Airport index mapped to Airport name
- Parsed from Routes Dataset:
  - Adjacency list of airports
  - Vector of maps of ints to ints - represents weight for each route
- Originally planned to use object called Airport to store ID, name, and longitude and latitude
  - Changed so that code is easier to read and understand
  - Simplifies code so that we don’t need to worry about memory allocation and deallocation

# BFS Visualizer

- BFS visualizer with different graphs
- Shows current nodes in queue
- Color coded nodes
  - Green indicates recently added
  - Red indicates recently removed
  - Blue indicates nodes remaining
- Loading bar with percentage

# Visualization of Airports on Map

- Each red dot represents an airport
- Map contains 95% of airports in dataset
- Used MP Stickers to plot each point
- Formula to convert longitude and latitude to Cartesian coordinates on a Mercator Map Projection



# Betweenness Centrality

- Used Dijkstra's algorithm to get the number of shortest path between two vertices.
  - Implement two version of Dijkstra's algorithm.
  - One is for only shortest path, We just use vector for the implementation, and it takes O(|E| + |V|^2) runtime.
  - Another is a modified version, calculating all the shortest paths. We use priority_queue here to make the runtime reduced to O(|E|+ |V|*log(|V|)).
- Node with greatest betweenness centrality value represents the most vital node
  - We are capable of getting the betweenness centrality of one node, which takes about twenty minutes.
  - The runtime of this algorithm is O(|V||E| + |V|^2*log(|V|)).

# Conclusion

- We are capable of finding shortest path from one airport to another.
- We write a relatively complete test to test our algorithms on simple graphs and test our data loadings.
- We calculated the betweenness centrality of two airports:
  - Hamad International Airport: 151564
  - Handan Airport: 0
  - So Hamad International Airport has more importance than Handan Airport
- Theoretically, we can find highest betweenness centrality value, but would take too much time
- Slow runtime of betweenness centrality combined with large dataset

# Individual thoughts

- If given more time, we could implement betweenness centrality more efficiently, and find the most important airport.
  - Or reduce size of dataset to allow for faster calculations of betweenness centrality
- Airport visualization requires a lot of memory to show every airport
- Plot the most impactful airport on the airport visualization map
  - Add routes to map to show connections between airports