CS 225 Final Project Presentation

By Eric Zhou, Joe Ye, Yucheng Feng, Lucas Lu

Our Goals

Data Source: OpenFlights

Traversal method: BFS, DFS

Algorithms to be used: Dijkstra's Algorithms to find the shortest path, Page Rank to display the frequencies of each airport

Our Development

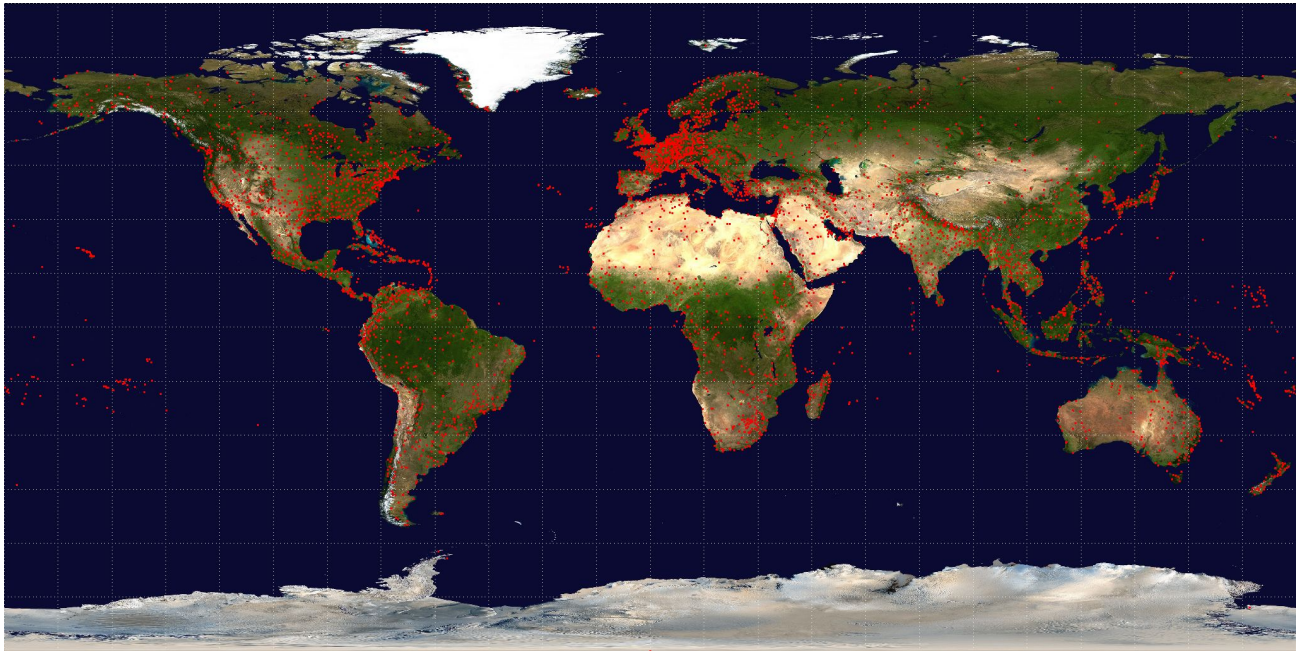
First Week:

- function “readEntry”
- function “getDistance”
- access data from data file
- function “getXXbyID”

Second Week:

- Implemented Dijkstra’s algorithm
- Implemented BFS and DFS traversal
- Implemented pagerank algorithm
- Implemented disjoint sets method

Data Processing



Data Processing

Load Data:

Construct Innerclasses to store the one set of data.

Using unordered map to store groups of data, Choose appropriate key to map to the corresponding class.

507, "London Heathrow Airport", "London", "United Kingdom",

"LHR", "EGLL", 51.4706, -0.461941, 83, 0, "E", "Europe/London",

"airport", "OurAirports"

```
// for airlines.dat
class Airlines {
    public:
        /**
         * Interal storage element for all entries in airlines.dat
         */
        class AirlineNode {
            public:
                /**
                 * Constructs one storage element
                 * @param name name of the airline
                 * @param alias alias of the airline
                 * @param IATA IATA code of the airline
                 * @param ICAO ICAO code of the airline
                 * @param callsign callsign of the airline
                 * @param country country where the airline belongs
                 * @param active whether the airline is active
                 */
                AirlineNode(string name, string alias, string IATA, string ICAO,
                        string callsign, string country, bool active) :
```

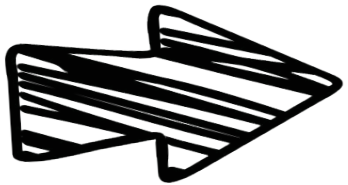
```
    private:
        /* Internal storage that holds all the airline infomation */
        unordered_map<int, AirlineNode*> _map;
};
```

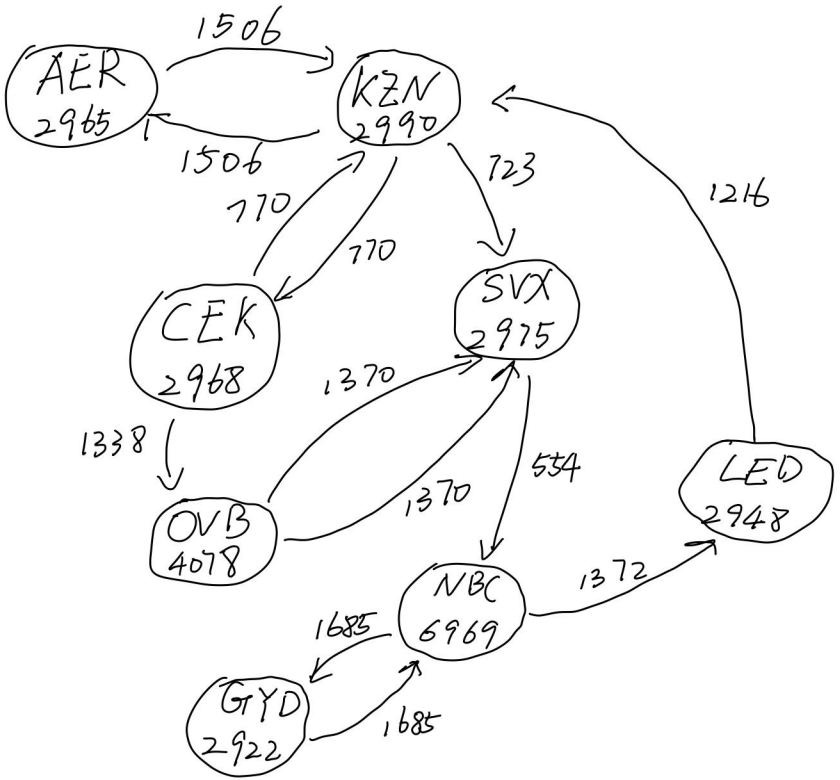
Data Processing - Convert to Graph

EdgeLabel: number of airlines that fly this route

EdgeWeight: distance between two airports

An Example Graph





Algorithm Coding

BFS & DFS:

In BFS, we use the vector to show whether or not the vertex is visited. Then, we set a queue. While queue is not empty, we delete the front vertex and add its neighbors into the queue. If queue is empty, we traverse all of the vertices and achieve the goal.

In DFS, we also use the vector to show whether the vertex is visited. However, this time we would achieve the deepest vertex of the start vertex. Then, we would shift to other vertex. We also use the while loop to do this.

Algorithm Coding

Dijkstra's Algorithm:

- Took advantage of std::make_heap, std::push_heap, std::pop_heap in the "algorithm" library"
- Used unordered_map to map vertex to distance and current vertex to previous vertex
- Output: changed from Graph* to vector<string> --> easier to read
- Debug process:
  - test on small routes and then on actual data
  - Cout to check the distances

Algorithm Coding

PageRank:

We construct Matrix class and implement matrix multiplication and Norm calculation to achieve the probability prediction by markov chain.

We accomplish label rank to find airport with largest probability as destination, standing for the popularity of the airports.

We also create the weight(distance) rank to find the airport with longest distance travelling airlines.

Our Deliverables

Test:

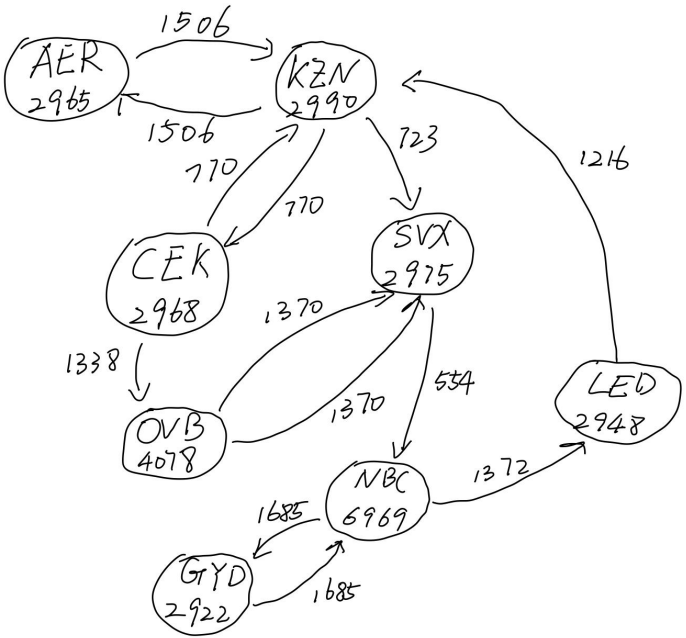
We include our tests in "tests" folder, and there are many tests and corresponding cases for all algorithms we use. Please run "make test" to see the outcome.

Usage:

We include some of our fundamental functions in main.cpp for testing and displaying results. Run “make” to see the outcome.

Dijkstra's Algorithm Shortest Path

```
(base) joeye@JoeyedeMacBook-Air 225project % ./main
The shortest path distance is 4468 kilometers.
KZN
SVX
NBC
GYD
(base) joeye@JoeyedeMacBook-Air 225project %
```



Shortest Path - Interesting findings

```
(base) joeye@JoeyedeMacBook-Air 225project % ./main
an alternative route from Guangzhou to Chicago: Guangzhou -> Shanghai -> Chicago with a distance of 12536 km.
The shortest path distance is 12443 kilometers.
Beijing Capital International Airport
Chicago O'Hare International Airport
(base) joeye@JoeyedeMacBook-Air 225project %
```

Home to School Route ^^

```
(base) joeye@JoeyedeMacBook-Air 225project % ./main
direct route from LA to Chicago with a distance of 2802 km.
The shortest path distance is 2801 kilometers.
City of Colorado Springs Municipal Airport
Chicago O'Hare International Airport
(base) joeye@JoeyedeMacBook-Air 225project %
```

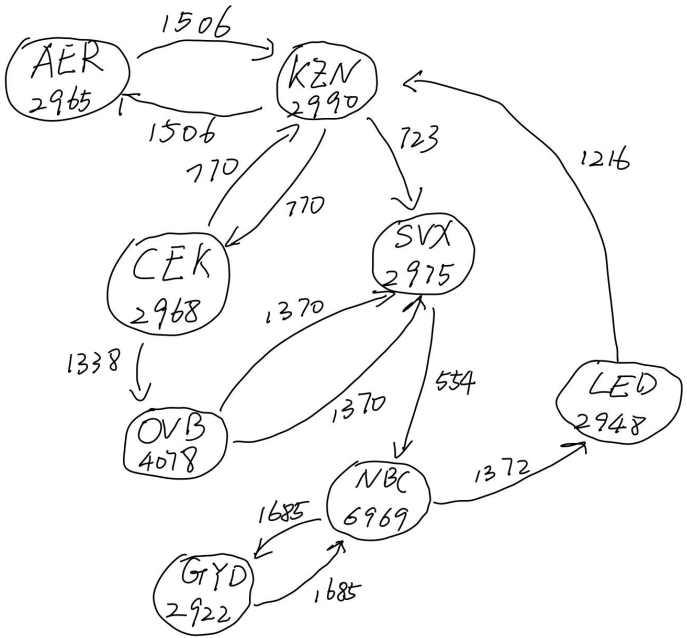
There seems to be a faster route from LAX to ORD

➔ Might due to some errors in rounding

Pagerank

```
airportsID 4078
airportsID 2922
airportsID 2948
airportsID 2965
airportsID 2975
airportsID 6969
airportsID 2968
airportsID 2990
0       0       0       0       2       0       0       0
0       0       0       0       0       1       0       0
0       0       0       0       0       0       0       1
0       0       0       0       0       0       0       1
0       0       0       0       0       1       0       0
0       1       1       0       0       0       0       0
1       0       0       0       0       0       0       1
0       0       0       1       1       0       1       0
```

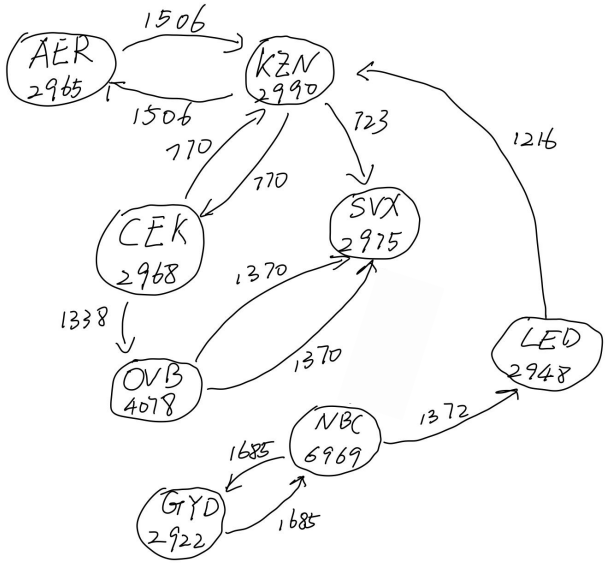
```
1: 3682      Hartsfield Jackson Atlanta International Airport
2: 3830      Chicago O'Hare International Airport
3: 3484      Los Angeles International Airport
4: 3670      Dallas Fort Worth International Airport
5: 1382      Charles de Gaulle International Airport
6: 507       London Heathrow Airport
7: 3364      Beijing Capital International Airport
8: 3316      Singapore Changi Airport
9: 3751      Denver International Airport
10: 340      Frankfurt am Main Airport
```



Disjoint set vs. BFS & DFS

Disjoint set is used to find if there are disconnected parts. After removing all the airports that does not have incoming and outgoing routes, we find that there are 6 disconnected parts in total.

However, the largest connected part found by disjoint set and BFS & DFS are different due to some parts are connected only in one direction.



Thanks for watching!