PROJETO AED

Grupo 65

Duarte Gonçalves – up202108772
Gonçalo Miranda – up202108773
Jorge Restivo – up202108886

Classes Used (1/2)

Airline Class

private:

```
string code;
string name;
string callsign;
string country;
```

public:

```
Airline(string code, string name, string callsign, string country);
string getCode() const;
string getName() const;
string getCallsign() const;
string getCountry() const;
```

Files:

Airline.h
Airline.cpp

Airport Class

private:

```
string code;
string name;
string city;
string country;
double latitude;
double longitude;
```

public:

```
Airport(string code, string name, string city, string country, double latitude, double longitude);
string getCode() const;
string getName() const;
string getCity() const;
string getCountry() const;
double getLatitude() const;
double getLongitude() const;
double distance(double lat, double lon) const;
```

Files:

Airport.h
Airport.cpp

Flight Class

private:

```
string source;
string target;
string airline;
```

public:

```
Flight(string source, string target, string airline);
string getSource() const;
string getTarget() const;
string getAirline() const;
```

Files:

Flight.h
Flight.cpp

MenuHandler Class

private:

```
string source;
string target;
string airline;
```

public:

```
Flight(string source, string target, string airline);
string getSource() const;
string getTarget() const;
string getAirline() const;
```

Files:

MenuHandler.h
MenuHandler.cpp

Classes Used (2/2)

Graph Class

private:

```
        struct Edge {
                int dest;
                set<string> airlines;
        };
        struct Node {
                list<Edge> edges;
                int dist = -1;
                bool visited = false;
                vector<int> previous;
        };
    int n;
    vector<Node> nodes;
```

public:

```
        explicit Graph(int n);
        void addEdge(int src, int dest, const string &airline);
        void bfs(int src);
        void bfs(set<int> src);
        vector<list<int>> shortestPaths(int src, int dest);
        vector<list<int>> shortestPaths(int src, int dest, const set<string> &airlines);
        vector<list<int>> shortestPaths(set<int> src, set<int> dest);
        vector<list<int>> shortestPaths(set<int> src, set<int> dest, const set<string> &airlines);
        int getNumOutgoing(int src) const;
        int getNumAirlines(int src) const;
        set<int> getDestinations(int src) const;
        set<int> reachable(int src, int hops);
```

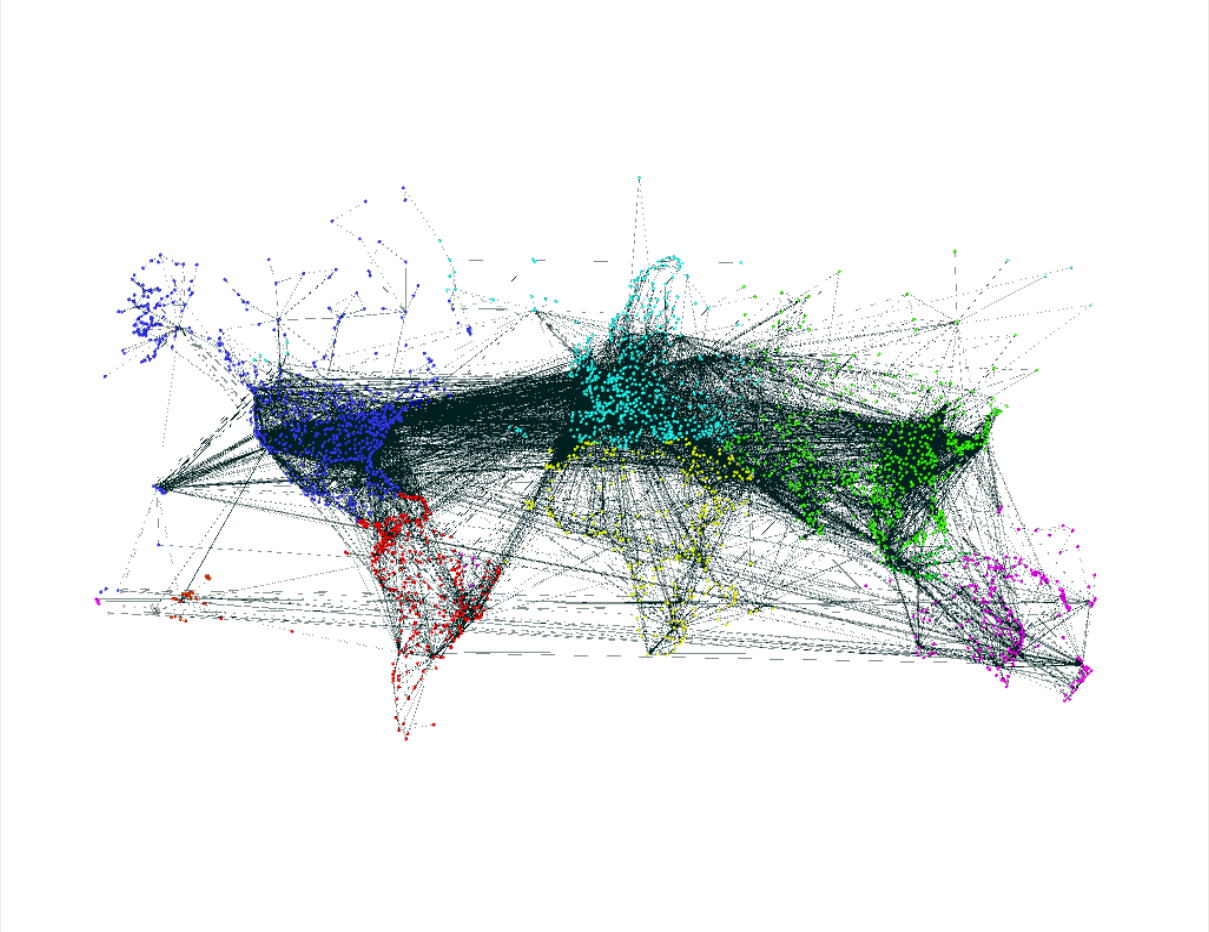
Files Used:
Graph.h
Graph.cpp

Reading Dataset

- ifstream function to get files;
- getline(file, line);
  One time to skip first line, then do a while loop to get each line;
- getline(iss,attribute, ‘,’);
  Separate each line into each attribute;
- .insert function to add each attribute to its correspondent vector;
- graph.addEdge to add each flight possible (edge) to correspondent codes (nodes);

Graph Used

- The graph used in this project represents the airports as nodes (every continent has color-coded nodes) and their respective flights are represented as edges.



Reading Dataset

```
void buildGraph (const CodeMap codes, Graph &graph) {
    ifstream file("../data/flights.csv");

    string line;
    getline(file, line); // skip first line

    while (getline(file, line)) {
        istringstream iss(line);
        string airline, src, dest;

        getline(iss, src, ',');
        getline(iss, dest, ',');
        getline(iss, airline, ',');

        graph.addEdge(codes.at(src), codes.at(dest), airline);
    }
}
```

```
void readAirports(CodeMap &codes, AirportMap &airports, CityMap &cities) {
    ifstream file("../data/airports.csv");

    string line;
    getline(file, line); // skip first line

    while (getline(file, line)) {
        istringstream iss(line);
        string code; string name; string city; string country; double latitude; double longitude;

        getline(iss, code, ',');
        getline(iss, name, ',');
        getline(iss, city, ',');
        getline(iss, country, ',');
        iss >> latitude;
        iss.ignore();
        iss >> longitude;

        codes.insert({code, codes.size()});
        airports.insert({codes[code], Airport(code, name, city, country, latitude, longitude)});
        cities[city].insert(codes.size() - 1);
    }
```

Main Feature

Implemented Features – Finding Minimum Route:

- This feature allows the user to find the minimum route between two different airport codes, two different cities or between two different locations (using their coordinates).
- The user also has the option to select either one or more prefered airlines to search for.
- The options using cities and location have a complexity of O(V(V+E)), while the codes one has a complexity of O(V+E).

```
Find the minimum route between:
--------------------------------------------------
1 - codes
2 - cities
3 - locations (coordinates)


0 - Back
--------------------------------------------------
Option:
```

```
Do you have a preference for an airline? no (0) / yes (1):
Option:
```

Implemented Features – Airport Statistics

- This feature allows the user to find the minimum route between two different airport codes, two different cities or between two different locations (using their coordinates).
- The user also has the option to select either one or more prefered airlines to search for.
- This feature has a complexity of O(V+E).

```
Airport code:MAD

Number of flights departing from MAD: 158
Number of airlines operating from MAD: 67
Number of destinations from MAD: 158
Number of cities with direct flights from MAD: 146
Number of countries with direct flights from MAD: 60

Set number of maximum flights:3

Number of airports reachable from MAD in 3 flights: 2580
Number of cities reachable from MAD in 3 flights: 2485
Number of countries reachable from MAD in 3 flights: 222

Press enter to continue...
```

Example using Madrid's Airport

User Interface:

- Here are the various menus that compose the user interface.

```
 Welcome!
-------------------------------------------------
 1 - Find the minimum route
 2 - Airport stats


 0 - Exit
-------------------------------------------------
 Option:
```

```
Airport code:MAD
Number of flights departing from MAD: 158
Number of airlines operating from MAD: 67
Number of destinations from MAD: 158
Number of cities with direct flights from MAD: 146
Number of countries with direct flights from MAD: 60

Set number of maximum flights:3

Number of airports reachable from MAD in 3 flights: 2580
Number of cities reachable from MAD in 3 flights: 2485
Number of countries reachable from MAD in 3 flights: 222

Press enter to continue...
```

```
Airport code:MAD
Number of flights departing from MAD: 158
Number of airlines operating from MAD: 67
Number of destinations from MAD: 158
Number of cities with direct flights from MAD: 146
Number of countries with direct flights from MAD: 60

Set number of maximum flights:3

Number of airports reachable from MAD in 3 flights: 2580
Number of cities reachable from MAD in 3 flights: 2485
Number of countries reachable from MAD in 3 flights: 222

Press enter to continue...
```

Biggest Difficulties and Work Distribution

- Our biggest difficulty resided in properly listing all the shortest paths, since it required some engineous
- The work was evenly distributed along all group members.