Project 2 -Air Travel Flight Management System

Algoritmos e Estruturas de Dados-23/24

Elementos:

Beatriz Sonnemberg - up202206098
Lara Cunha - up202108876
Marta Costa - up202207879

The Classes used and their respective Attributes:

- Airport(code, name, city, country, latitude, longitude);
- Airline(code, name,country, callsign);
- Flight(source,target,airline);
- Vertex(Airport info, vector of Edges, visited, num, low, distance, arrives);
- Edge(Vertex* dest, Airline weight);
- Graph(unordered_map with aiport codes and their respective vertices)
- Manager(graph)
- Menu
- Read

Description of reading the given dataset:

```cpp
void Read::readAirports() {

    string Code,Name,City,Country;
    double Latitude,Longitude;
    string dir = "../dataset/airports.csv", line;
    ifstream ifile( s: dir);
    char sep = ',';
    getline( &: ifile, &: line);
    while (getline( &: ifile,  &: line)){
        stringstream iss( s: line);
        getline( &: iss, &: Code, dlm: sep);
        getline( &: iss, &: Name, dlm: sep);
        getline( &: iss, &: City, dlm: sep);
        getline( &: iss,  &: Country,  dlm: sep);
        iss>>Latitude>>sep;
        iss>>Longitude>>sep;
        Airport airport = Airport(Code,Name,City, Country, Latitude,Longitude)
        airportMap.emplace( &: Code, &: airport);
        flights.addVertex( in: airport);
    }
}
```

```cpp
void Read::readFlights() {

    string Source,Target,airline;
    string dir = "../dataset/flights.csv", line;
    ifstream ifile( s: dir);
    char sep = ',';
    getline ( &: ifile,  &: line);

    while (getline( &: ifile,  &: line)) {
        stringstream iss( s: line);
        getline( &: iss,  &: Source,  dlm: sep);
        getline( &: iss,  &: Target,  dlm: sep);
        getline( &: iss,  &: airline,  dlm: sep);

        flightSet.emplace(Source,Target,airline);

        flights.addEdge( &: airportMap.at( k: Source),  &: airportMap.at( k:

    }
}
```

```cpp
void Read::readAirlines() {
    string Code,Name,Callsign,Country;
    string dir = "../dataset/airlines.csv", line;
    ifstream ifile( s: dir);
    char sep =',';
    getline( &: ifile, &: line);
    while (getline( &: ifile,  &: line)){
        stringstream  iss( s: line);
        getline( &: iss, &: Code, dlm: sep);
        getline( &: iss, &: Name, dlm: sep);
        getline( &: iss, &: Callsign, dlm: sep);
        getline( &: iss, &: Country, dlm: sep);
        Airline airline = Airline(Code,Name,Callsign,Country);
        airlineMap.emplace( &: Code, &: airline);

    }
}
```

Description of reading the given dataset:

We read CSV files from this three function, implemented in the class Read.

All this functions follow the same methods:

-readAirports() ->This function processes each line of the corresponding CSV file to extract the individual data, creates Airport objects with this data and stores them in an unordered map (airportMap) and in a graph (flights);

-readAirlines() -> This function processes each row from the corresponding CSV file to extract the individual data, creates Airline objects with this data and stores it in a map (airlineMap);

-readFlights() -> This function reads a CSV file containing information about flights, processes each line to extract individual data, stores this data in a set (flightSet) and adds the information to the flight network represented by the flights graph.

Description of the graph used to represent the dataset:

This class is fundamental in the graph structure, stroring airport informationa, managing edges and providing methods for graph algorithms

- **Vertex:**

  - Airport info
  - Vector of Edges
  - Boolean visited
  - Integer num
  -Integer low
  -Integer distance
  -Integer arrives

```
class Vertex {
    Airport info;
    vector<Edge> adj;
    bool visited;
    int num;
    int low;
    int distance;
    int arrives;
```

```
public:
    Vertex(Airport in);
    Airport getInfo() const;
    int getArrives();
    void addArrives();
    bool isVisited() const;
    void setVisited(bool v);
    vector<Edge> getAdj() const;
    void addEdge(Vertex *dest, Airline w);
    int getDistance() const;
    void setDistance(int d);
    bool removeEdgeTo(Vertex *d);
    bool operator==(Vertex v) ;
    int getNum() const;
    void setNum(int num);
    int getLow() const;
    void setLow(int low);
};
```

Description of the graph used to represent the dataset:

- **Edge:**
  - Pointer to object Vertex
  - Airline weight

The Edge class represents na edge in the graph, connecting two vértices and associated with a airline. This connection is fundamental for graph traversal and analysis.

```
class Edge {
    Vertex * dest;
    Airline weight;
public:
    Edge(Vertex *d, Airline airline);
    Vertex *getDest() const;
    Airline getWeight() const;
};
```

Description of the graph used to represent the dataset:

- **Graph:**
  - unordered_map<string, Vertex *> vertexSet;

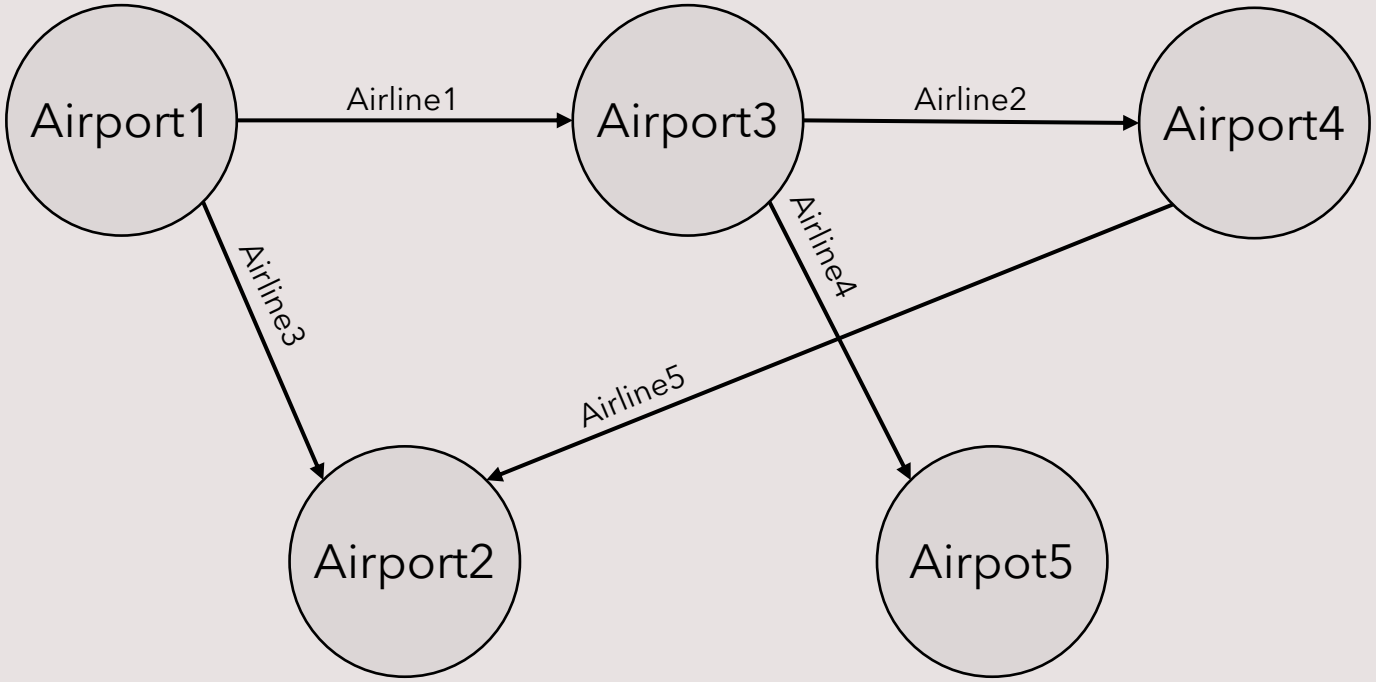
The Graph class provides essential methods for finding vértices, calculating in-degrees and managing edges and vértices.
The vertexSet is na unordered_mao where airport codes serve as keys, and the corresponding values are pointers to the associated vértices.

```
class Graph {
    unordered_map<string, Vertex *> vertexSet;
public:

    int inDegree(const std::string& airportCode) const;
    Graph();
    Vertex *findVertex( Airport &in) const;
    Vertex *findVertex(const string &airportCode) const;
    int getNumVertex() const;
    bool addVertex(Airport in);
    bool addEdge( Airport &sourc,  Airport &dest, Airline w);
    unordered_map<string, Vertex * > getVertexSet() const;

};
```

Graph Representation



Description of implemented functionalities and associated algorithms:

We implemented the following functions, that are in the class Manager:

- void getGlobalNumAirports() -> returns and prints the total number of airports present in the flight network represented by the flights_graph graph. Complexity: O(1);
- void getGlobalNumCities() -> calculates and prints the total number of unique cities within the network of flights represented by the graph flights_graph. It uses a set to track unique combinations of city and country for each airport vertex in the graph, and then outputs the count of these unique city-country pairs . Complexity: O(V·logK);
- void getGlobalNumContries() ->calculates and prints the total number of unique countries within the network of flights represented by the graph flights_graph. It utilizes a set to keep track of unique country names associated with each airport vertex in the graph. Complexity: O(V·logK);
- void getGlobalNumAirlines() -> calculates and prints the total number of unique airlines within the network of flights represented by the graph flights_graph. It utilizes a set to keep track of unique airline codes associated with each edge (flight route) in the graph. The function iterates through all vertices in the graph and, for each vertex, iterates through its adjacent edges to collect unique airline codes. Complexity: O(V·E·logK);
- void getNumAvailableFlights() -> calculates and prints the total number of available flights within the network of flights represented by the graph flights_graph. It does so by iterating through all vertices in the graph and adding the count of adjacent edges (flight routes) for each vertex. Complexity: O(V+E);

Description of implemented functionalities and associated algorithms:

- void getNumFlightsFromAirport()->prints the number of flights departing from a specified airport, identified by its airport code, within the network of flights represented by the graph flights_graph. It checks if the specified airport code exists in the graph, and if so, it retrieves the corresponding vertex and outputs the count of outgoing flights. Complexity: O(1);
- void getNumAirlinesFromAirport() ->calculates and prints the number of different airlines operating flights departing from a specified airport, identified by its airport code, within the network of flights represented by the graph flights_graph. It first checks if the specified airport code exists in the graph. If the airport is found, the function iterates through the outgoing edges of the airport vertex, collecting unique airline codes in a set. Complexity: O(E·logK);
- void getNumFlightsPerCity() -> calculates and prints the total number of flights associated with a specific city and country within the network of flights represented by the graph flights_graph. It iterates through all vertices in the graph, checking if the city and country of each vertex match the specified parameters (cityName and country). For the vertices that match, it adds the count of both outgoing and incoming flights to the total number of flights. Complexity: O(V+E);
- void getNumFlightsPerAirline() -> calculates and prints the total number of flights associated with a specific airline, identified by its airline code, within the network of flights represented by the graph flights_graph. It iterates through all vertices in the graph and, for each vertex, iterates through its outgoing edges (flights). If an edge has an airline with the specified code, it increments the count of flights. Complexity: O(V·E);

Description of implemented functionalities and associated algorithms:

- void getNumDestinationsPerAirline() ->calculates and prints the total number of unique destinations served by a specific airline, identified by its airline code, within the network of flights represented by the graph flights_graph. It uses a set to track unique destination airport codes associated with each edge (flight route) where the specified airline is involved. The function iterates through all vertices in the graph, checking outgoing edges for the specified airline code, and adds unique destination codes to the set. Complexity: O(V·E·logK);
- void getNumDestCountriesPerCity()-> calculates and prints the total number of different countries flown to from a specific city within the network of flights represented by the graph flights_graph. It utilizes a set to track unique destination countries associated with each outgoing edge (flight route) from the specified city and country. The function iterates through all vertices in the graph, checking if the city and country of each vertex match the specified parameters (cityName and country). For the vertices that match, it iterates through their outgoing edges and adds the destination countries to the set. Complexity: O(E·logK);
- void getNumDestCountriesPerAirport() -> calculates and prints the total number of different countries flown to from a specific airport within the network of flights represented by the graph flights_graph. It uses a set to track unique destination countries associated with each outgoing edge (flight route) from the specified airport. The function directly accesses the outgoing edges of the airport vertex identified by the provided airport code and adds the destination countries to the set. Complexity: O(E·logK);
- void getNumDestinations() -> performs a breadth-first search (BFS) starting from a specified airport identified by its airport code (airportCode) within the network of flights represented by the graph flights_graph. It calculates and prints various statistics, including the total number of reachable airports, the number of unique cities, and the number of unique countries. The function uses a queue for BFS traversal, updating counts and sets of cities and countries as it explores the connected vertices from the starting airport. Complexity: O(V+E+(V+E)·logK);

Description of implemented functionalities and associated algorithms:

- void getCountReachableDestinations() ->performs a modified breadth-first search (BFS) starting from a specified airport identified by its airport code (airportCode) within the network of flights represented by the graph flights_graph. It calculates and prints various statistics, including the total number of reachable airports, the number of unique cities, and the number of unique countries, with a limitation on the maximum number of stops (maxStops). The function uses a queue for BFS traversal, updating counts and sets of cities and countries as it explores the connected vertices from the starting airport, considering a maximum number of stops. Complexity: O(V+E);
- void getMaxTrip() -> finds and prints the pair(s) of airports with the maximum distance between them within the network of flights represented by the graph flights_graph. It uses a breadth-first search (BFS) algorithm to calculate the distance between each pair of airports. For each starting airport, it explores all other airports in the graph, updating the maximum distance and the set of corresponding pairs as needed. Complexity: O(V^2 + VE);
- void getTopKAirports() -> identifies and prints the top-k airports based on a specific criterion within the network of flights represented by the graph flights_graph. The criterion is determined by the sum of outgoing and incoming flights for each airport. The function creates a vector of pairs, where each pair contains a vertex (airport) and its corresponding score (sum of outgoing and incoming flights). It then sorts this vector in descending order based on the scores and outputs the top-k airports along with their scores in the console. Complexity: O(V log(V));
- void getIdentifyEssentialAirports() -> identifies and prints the essential airports, specifically the articulation points, within the network of flights represented by the graph flights_graph. It creates an undirected graph based on the original graph and performs a depth-first search (DFS) to find articulation points. Articulation points are crucial nodes whose removal would increase the number of connected components in the graph. Complexity: O(V + E);
- void findBestFlightOption() -> determines the optimal flight paths, considering preferred airlines and minimizing stops, between a specified source and destination within a network of flights. Complexity: O(V * (V + E));

Description of the user interface:

The Menu class is responsible for handling user interaction and displaying menus related to the Flight Management System

```
 Global Statistics Menu
----------------------
1. Number of Airports
2. Number of Cities
3. Number of Countries
4. Number of Airlines
5. Number of Available Flights
6. Display the maximum trip
7. Exit to Main Menu
----------------------
```

```
Flight Management System
------------------------
1. Display Statistics
2. Display Airport Information
3. Find Best Flight Option
4. Identify Essential Airports
5. Exit
------------------------
```

```
 Airport Information Menu
-----------------------
1. Number of Flights from an Airport
2. Number of Airlines from an Airport
3. Number of Flights per City
4. Number of Flights per Airline
5. Number of Destinations per Airline
6. Number of Destination Countries per City
7. Number of Destination Countries per Airport
8. Number of Destinations for an Airport
9. Number of Reachable Destinations from an Airport
10. The top-k airport with the greatest air traffic capacity
11. Exit to Main Menu
-----------------------
```

One or two functionality(s) to highlight:

Our program offers numerous features that allow users to have a comprehensive, objective, and informative experience.

The highlight in our work is the fact that we implemented extra requests that the users may need, and we pay attention to the fact that there are cities with the same name but from different countries.

Main difficulties and participation of each group member:

During the execution of this project, we encountered some difficulties and errors along the way, which fortunately we managed to overcome.

The work tasks were distributed equally, and it was carried out with the cooperation of all elements.