# Flight Network management system 1.0

Generated by Doxygen 1.11.0

1 Air Travel Flight Management System	1
1.1 Overview	
1.2 Dataset	
1.3 Features	
1.3.1 Data Handling	
1.3.2 Flight Management System	
1.3.3 Network Statistics	
1.3.4 Maximum Trip and Essential Airports	
1.3.5 Best Flight Options	
1.3.6 Flight Filtering	
1.3.7 Documentation	
1.4 Implementation Details	
1.5 How to Use	
1.5.1 To run the program, run the following commands:	
1.6 Authors	
1.6.1 Happy flying!	
2 Class Index	3
2.1 Class List	
3 File Index	5
3.1 File List	
4 Class Documentation	7
4.1 Airline Class Reference	
4.1.1 Detailed Description	
4.1.2 Constructor & Destructor Documentation	
<b>4.1.2.1 Airline()</b> [1/2]	
<b>4.1.2.2 Airline()</b> [2/2]	
4.1.3 Member Function Documentation	
4.1.3.1 getCallsign()	
4.1.3.2 getCode()	
4.1.3.3 getCountry()	
4.1.3.4 getName()	
4.1.3.5 setCallsign()	
4.1.3.6 setCode()	
4.1.3.7 setCountry()	
4.1.3.8 setName()	
4.1.4 Member Data Documentation	
4.1.4.1 callsign	
4.1.4.2 code	
4.1.4.3 country	
4.1.4.4 name	

4.2 Airport Class Reference	1
4.2.1 Detailed Description	1
4.2.2 Constructor & Destructor Documentation	1
<b>4.2.2.1 Airport()</b> [1/2]	1
<b>4.2.2.2 Airport()</b> [2/2]	1
4.2.3 Member Function Documentation	1
4.2.3.1 getCity()	1
4.2.3.2 getCode()	1
4.2.3.3 getCountry()	1
4.2.3.4 getName()	1
4.2.3.5 getPosition()	1
4.2.3.6 operator<()	1
4.2.3.7 operator==()	1
4.2.3.8 setCity()	1
4.2.3.9 setCode()	1
4.2.3.10 setCountry()	1
4.2.3.11 setName()	1
4.2.3.12 setPosition()	1
4.2.4 Member Data Documentation	1
4.2.4.1 city	1
4.2.4.2 code	1
4.2.4.3 country	1
4.2.4.4 name	1
4.2.4.5 position	1
4.3 App Class Reference	1
4.3.1 Constructor & Destructor Documentation	1
4.3.1.1 App()	1
4.3.2 Member Function Documentation	1
4.3.2.1 bestFlightMenu()	1
4.3.2.2 globalStatistics()	1
4.3.2.3 goBackStatisticsMenu()	1
4.3.2.4 mainMenu()	1
4.3.2.5 numberOfDestinations()	1
4.3.2.6 reachableDest()	1
4.3.2.7 showNumFlights()	1
4.3.2.8 statisticsMenu()	1
4.3.3 Member Data Documentation	1
4.3.3.1 flightnetwork	1
4.4 Edge< T > Class Template Reference	1
4.4.1 Detailed Description	1
4.4.2 Constructor & Destructor Documentation	1
4.4.2.1 Edge()	1

4.4.3 Member Function Documentation	. 19
4.4.3.1 getDest()	. 19
4.4.3.2 getInfo()	. 19
4.4.3.3 getWeight()	. 20
4.4.3.4 setDest()	. 20
4.4.3.5 setInfo()	. 20
4.4.3.6 setWeight()	. 20
4.4.4 Friends And Related Symbol Documentation	. 21
4.4.4.1 Graph < T >	. 21
4.4.4.2 Vertex < T >	. 21
4.4.5 Member Data Documentation	. 21
4.4.5.1 dest	. 21
4.4.5.2 info	. 21
4.4.5.3 weight	. 21
4.5 FlightNetwork Class Reference	. 21
4.5.1 Detailed Description	. 23
4.5.2 Constructor & Destructor Documentation	. 23
<b>4.5.2.1 FlightNetwork()</b> [1/2]	. 23
<b>4.5.2.2 FlightNetwork()</b> [2/2]	. 24
4.5.3 Member Function Documentation	. 24
4.5.3.1 airlineCodeToName()	. 24
4.5.3.2 airportCodeToName()	. 24
4.5.3.3 bestFlight()	. 26
4.5.3.4 cityCriteria()	. 26
4.5.3.5 codeCriteria()	. 27
4.5.3.6 coordinateCriteria()	. 27
4.5.3.7 getAiportsGraph()	. 28
4.5.3.8 getAirportsDestinations()	. 28
4.5.3.9 getCitiesDestinations()	. 28
4.5.3.10 getCountriesDestinations()	. 29
4.5.3.11 getDiffCountriesAirport()	. 29
4.5.3.12 getDiffCountriesCity()	. 29
4.5.3.13 getEssentialAirports()	. 30
4.5.3.14 getGlobalNumOfAirports()	. 30
4.5.3.15 getGlobalNumOfFlights()	. 30
4.5.3.16 getGreatestTraffic()	. 30
4.5.3.17 getReachableAirports()	. 31
4.5.3.18 getReachableCities()	. 31
4.5.3.19 getReachableCountries()	. 31
4.5.3.20 listBestFlights()	. 32
4.5.3.21 maximumTrip()	. 32
4.5.3.22 nameCriteria()	. 33

4.5.3.23 numFlightsAirline()	33
4.5.3.24 numFlightsAirport()	33
4.5.3.25 numFlightsCity()	34
4.5.4 Member Data Documentation	34
4.5.4.1 airportsGraph	34
4.6 Graph < T > Class Template Reference	34
4.6.1 Detailed Description	35
4.6.2 Member Function Documentation	36
4.6.2.1 addEdge()	36
4.6.2.2 addVertex()	36
4.6.2.3 bfs()	36
4.6.2.4 bfsDistance()	37
<b>4.6.2.5 dfs()</b> [1/2]	37
<b>4.6.2.6 dfs()</b> [2/2]	37
4.6.2.7 dfsVisit()	38
4.6.2.8 EdgesAtDistanceDFS()	38
4.6.2.9 findVertex()	38
4.6.2.10 getNumVertex()	39
4.6.2.11 getVertexSet()	39
4.6.2.12 inDegree()	39
4.6.2.13 nodesAtDistanceDFS()	39
4.6.2.14 removeEdge()	40
4.6.2.15 removeVertex()	40
4.6.3 Member Data Documentation	40
4.6.3.1 vertexSet	40
4.7 Vertex < T > Class Template Reference	41
4.7.1 Detailed Description	42
4.7.2 Constructor & Destructor Documentation	42
4.7.2.1 Vertex()	42
4.7.3 Member Function Documentation	42
4.7.3.1 addEdge()	42
4.7.3.2 getAdj()	43
4.7.3.3 getDistance()	43
4.7.3.4 getInDegree()	43
4.7.3.5 getInfo()	44
4.7.3.6 getLow()	44
4.7.3.7 getNum()	44
4.7.3.8 isProcessing()	44
4.7.3.9 isVisited()	44
4.7.3.10 removeEdgeTo()	44
4.7.3.11 setAdj()	45
4.7.3.12 setDistance()	45

4.7.3.13 setInDegree()	 45
4.7.3.14 setInfo()	 46
4.7.3.15 setLow()	 46
4.7.3.16 setNum()	 46
4.7.3.17 setProcessing()	 46
4.7.3.18 setVisited()	 47
4.7.4 Friends And Related Symbol Documentation	 47
4.7.4.1 Graph < T >	 47
4.7.5 Member Data Documentation	 47
4.7.5.1 adj	 47
4.7.5.2 distance	 47
4.7.5.3 indegree	 47
4.7.5.4 info	 47
4.7.5.5 low	 48
4.7.5.6 num	 48
4.7.5.7 processing	 48
4.7.5.8 visited	 48
5 File Documentation	49
5.1 inc/Airline.hpp File Reference	
5.2 Airline.hpp	
5.3 inc/Airport.hpp File Reference	
5.4 Airport.hpp	
5.5 inc/App.hpp File Reference	
5.5.1 Function Documentation	
5.5.1.1 clearScreen()	
5.6 App.hpp	
5.7 inc/FlightNetwork.hpp File Reference	
5.7.1 Function Documentation	
5.7.1.1 dfs_art()	
5.7.1.2 haversineDistance()	
5.8 FlightNetwork.hpp	
5.9 inc/Graph.hpp File Reference	
5.9.1 Function Documentation	
5.9.1.1 nodesAtDistanceDFSVisit() [1/2]	
5.9.1.2 nodesAtDistanceDFSVisit() [2/2]	
5.10 Graph.hpp	
5.11 README.md File Reference	
5.12 src/Airline.cpp File Reference	
5.13 src/Airport.cpp File Reference	
5.14 src/App.cpp File Reference	
5.14.1 Function Documentation	

Inde	•													65
	5.16.1.1 main()		٠.											64
	5.16.1 Function Documentation													64
5	.16 src/main.cpp File Reference													64
	5.15.1.2 haversineDistance()													63
	5.15.1.1 dfs_art()													63
	5.15.1 Function Documentation													63
5	.15 src/FlightNetwork.cpp File Reference													62
	5.14.1.1 clearScreen()													62

## **Chapter 1**

## Air Travel Flight Management System

## 1.1 Overview

Welcome to the Air Travel Flight Management System! This system is designed to help users explore and plan their air travel efficiently. It leverages real-world data about airports, airlines, and flights to provide a comprehensive tool for making informed decisions.

## 1.2 Dataset

The system utilizes a dataset containing information about 3019 airports, 444 airlines, and 63832 flights. This dataset includes details such as airport codes, names, cities, countries, latitude, and longitude, among other information.

## 1.3 Features

## 1.3.1 Data Handling

• Read and parse provided data, loading it into an appropriate graph data structure.

## 1.3.2 Flight Management System

· Develop a user-friendly system with functionalities for exploring and planning air travel.

## 1.3.3 Network Statistics

- · Calculate and list statistics such as the global number of airports and flights.
- · Provide statistics on flights per airport, per city, per airline, and more.

## 1.3.4 Maximum Trip and Essential Airports

- · Identify the maximum number of stops for a round-trip.
- · Identify essential airports for the network's circulation capability.

## 1.3.5 Best Flight Options

Present the best flight options based on user-specified criteria, such as airport code, name, city, or coordinates.

## 1.3.6 Flight Filtering

 Allow users to filter flight options based on preferences, such as specific airlines or minimizing the number of different airlines.

## 1.3.7 Documentation

· Include Doxygen documentation for relevant functions, indicating their time complexity.

## 1.4 Implementation Details

This system is implemented in C++ using the provided Graph class for managing the network of airports and flights. The data is loaded from CSV files, and various functionalities are provided to assist users in navigating the flight network.

## 1.5 How to Use

To get started, instantiate the FlightNetwork class with filenames for airlines, airports, and flights data. Then, use the provided functions to explore the features mentioned above.

## 1.5.1 To run the program, run the following commands:

mkdir build
cd build
cmake ..
make
./aed\_project2

## 1.6 Authors

Leonardo Garcia Marcel Madeiros Pedro Castro

## 1.6.1 Happy flying!

## **Chapter 2**

## **Class Index**

## 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Airline		
	Represents an airline with specific attributes such as code, name, callsign, and country	7
Airport		
	Represents an airport with attributes such as code, name, city, country, and position	11
App		16
Edge<	Γ>	
	Represents an edge between two vertices in a graph with generic information of type T	18
FlightNe	twork	
	Represents a flight network consisting of airports and flights	21
Graph<	T >	
	Represents a generic graph with vertices of type T	34
Vertex<	T >	
	Represents a vertex in a graph with generic information of type T	41

4 Class Index

## **Chapter 3**

## **File Index**

## 3.1 File List

Here is a list of all files with brief descriptions:

inc/Airline.hpp	 	 49
inc/Airport.hpp	 	 50
inc/App.hpp	 	 50
inc/FlightNetwork.hpp	 	 51
inc/Graph.hpp	 	 54
src/Airline.cpp	 	 62
src/Airport.cpp	 	 62
src/App.cpp	 	 62
src/FlightNetwork.cpp	 	 62
src/main.cpp	 	 64

6 File Index

## **Chapter 4**

## **Class Documentation**

## 4.1 Airline Class Reference

Represents an airline with specific attributes such as code, name, callsign, and country.

```
#include <Airline.hpp>
```

#### **Public Member Functions**

• Airline (const std::string code, const std::string name, const std::string callsign, const std::string country)

Constructor for creating an Airline object with specified attributes.

• Airline (const std::string &code)

Constructor for creating an Airline object with only the code specified.

• std::string getCode () const

Gets the airline code.

void setCode (const std::string &c)

Sets the airline code.

• std::string getName () const

Gets the name of the airline.

• void setName (const std::string &n)

Sets the name of the airline.

• std::string getCallsign () const

Gets the callsign of the airline.

void setCallsign (const std::string &c)

Sets the callsign of the airline.

• std::string getCountry () const

Gets the country associated with the airline.

void setCountry (const std::string &c)

Sets the country associated with the airline.

#### **Private Attributes**

- · std::string code
- std::string name
- · std::string callsign
- std::string country

## 4.1.1 Detailed Description

Represents an airline with specific attributes such as code, name, callsign, and country.

## 4.1.2 Constructor & Destructor Documentation

## 4.1.2.1 Airline() [1/2]

```
Airline::Airline (

const std::string code,

const std::string name,

const std::string callsign,

const std::string country)
```

Constructor for creating an Airline object with specified attributes.

#### **Parameters**

code	The airline code.
name	The name of the airline.
callsign	The callsign of the airline.
country	The country associated with the airline.

## 4.1.2.2 Airline() [2/2]

Constructor for creating an Airline object with only the code specified.

## **Parameters**

code The airline code
-----------------------

#### 4.1.3 Member Function Documentation

## 4.1.3.1 getCallsign()

```
string Airline::getCallsign ( ) const
```

Gets the callsign of the airline.

## Returns

The callsign of the airline.

4.1 Airline Class Reference 9

## 4.1.3.2 getCode()

```
string Airline::getCode ( ) const
```

Gets the airline code.

Returns

The airline code.

## 4.1.3.3 getCountry()

```
string Airline::getCountry ( ) const
```

Gets the country associated with the airline.

Returns

The country associated with the airline.

## 4.1.3.4 getName()

```
string Airline::getName ( ) const
```

Gets the name of the airline.

Returns

The name of the airline.

## 4.1.3.5 setCallsign()

```
void Airline::setCallsign ( {\tt const\ std::string\ \&\ c\ )}
```

Sets the callsign of the airline.

**Parameters** 

c The new callsign of the airline.

## 4.1.3.6 setCode()

```
void Airline::setCode ( {\tt const\ std::string\ \&\ c\ )}
```

Sets the airline code.

#### **Parameters**

c The new airline code.

## 4.1.3.7 setCountry()

```
void Airline::setCountry ( {\tt const\ std::string\ \&\ c\ )}
```

Sets the country associated with the airline.

#### **Parameters**

c The new country associated with the airline.

## 4.1.3.8 setName()

```
void Airline::setName ( const std::string & n )
```

Sets the name of the airline.

#### **Parameters**

n The new name of the airline.

## 4.1.4 Member Data Documentation

## 4.1.4.1 callsign

```
std::string Airline::callsign [private]
```

The callsign of the airline.

## 4.1.4.2 code

```
std::string Airline::code [private]
```

The airline code.

## 4.1.4.3 country

```
std::string Airline::country [private]
```

The country associated with the airline.

#### 4.1.4.4 name

```
std::string Airline::name [private]
```

The name of the airline.

The documentation for this class was generated from the following files:

- · inc/Airline.hpp
- src/Airline.cpp

## 4.2 Airport Class Reference

Represents an airport with attributes such as code, name, city, country, and position.

```
#include <Airport.hpp>
```

#### **Public Member Functions**

• Airport (const std::string &code, const std::string &name, const std::string &city, const std::string &country, const std::pair< float, float > &position)

Constructor for creating an Airport object with specified attributes.

Airport (const std::string &code)

Constructor for creating an Airport object with only the code specified.

• std::string getCode () const

Gets the airport code.

void setCode (const std::string &c)

Sets the airport code.

• std::string getName () const

Gets the name of the airport.

void setName (const std::string &n)

Sets the name of the airport.

std::string getCity () const

Gets the city where the airport is located.

void setCity (const std::string &c)

Sets the city where the airport is located.

• std::string getCountry () const

Gets the country where the airport is located.

void setCountry (const std::string &c)

Sets the country where the airport is located.

- std::pair < float, float > getPosition () const

Gets the geographical position of the airport.

void setPosition (const std::pair< float, float > &pos)

Sets the geographical position of the airport.

bool operator== (const Airport &other)

Overloaded equality operator to compare airports based on their codes.

bool operator< (const Airport &other)</li>

Overloaded less-than operator to compare airports based on their codes.

## **Private Attributes**

```
std::string code
std::string name
std::string city
std::string country
std::pair< float, float > position
```

## 4.2.1 Detailed Description

Represents an airport with attributes such as code, name, city, country, and position.

## 4.2.2 Constructor & Destructor Documentation

## 4.2.2.1 Airport() [1/2]

Constructor for creating an Airport object with specified attributes.

#### **Parameters**

code	The airport code.
name	The name of the airport.
city	The city where the airport is located.
country	The country where the airport is located.
position	The geographical position of the airport (latitude, longitude).

## 4.2.2.2 Airport() [2/2]

Constructor for creating an Airport object with only the code specified.

## **Parameters**

code The airport code.	
------------------------	--

## 4.2.3 Member Function Documentation

## 4.2.3.1 getCity()

```
string Airport::getCity ( ) const
```

Gets the city where the airport is located.

Returns

The city where the airport is located.

## 4.2.3.2 getCode()

```
string Airport::getCode ( ) const
```

Gets the airport code.

Returns

The airport code.

## 4.2.3.3 getCountry()

```
string Airport::getCountry ( ) const
```

Gets the country where the airport is located.

Returns

The country where the airport is located.

## 4.2.3.4 getName()

```
string Airport::getName ( ) const
```

Gets the name of the airport.

Returns

The name of the airport.

## 4.2.3.5 getPosition()

```
pair< float, float > Airport::getPosition ( ) const
```

Gets the geographical position of the airport.

Returns

The geographical position of the airport (latitude, longitude).

## 4.2.3.6 operator<()

Overloaded less-than operator to compare airports based on their codes.

#### **Parameters**

other The	other airport to compare.
-----------	---------------------------

## Returns

True if the code of this airport is less than the code of the other airport.

## 4.2.3.7 operator==()

Overloaded equality operator to compare airports based on their codes.

#### **Parameters**

#### Returns

True if the airports have the same code, false otherwise.

## 4.2.3.8 setCity()

```
void Airport::setCity ( {\tt const\ std::string\ \&\ c\ )}
```

Sets the city where the airport is located.

#### **Parameters**

c The new city where the airport is located.

## 4.2.3.9 setCode()

```
void Airport::setCode ( {\tt const\ std::string\ \&\ c\ )}
```

Sets the airport code.

## **Parameters**

c The new airport code.

## 4.2.3.10 setCountry()

```
void Airport::setCountry ( {\tt const\ std::string\ \&\ c\ )}
```

Sets the country where the airport is located.

**Parameters** 

c The new country where the airport is located.

## 4.2.3.11 setName()

```
void Airport::setName ( {\tt const \ std::string \ \& \ n \ )}
```

Sets the name of the airport.

#### **Parameters**

n The new name of the airport.

## 4.2.3.12 setPosition()

Sets the geographical position of the airport.

**Parameters** 

pos The new geographical position of the airport (latitude, longitude).

## 4.2.4 Member Data Documentation

#### 4.2.4.1 city

```
std::string Airport::city [private]
```

The city where the airport is located.

## 4.2.4.2 code

```
std::string Airport::code [private]
```

The airport code.

## 4.2.4.3 country

```
std::string Airport::country [private]
```

The country where the airport is located.

#### 4.2.4.4 name

```
std::string Airport::name [private]
```

The name of the airport.

## 4.2.4.5 position

```
std::pair<float, float> Airport::position [private]
```

The geographical position of the airport (latitude, longitude).

The documentation for this class was generated from the following files:

- inc/Airport.hpp
- src/Airport.cpp

## 4.3 App Class Reference

```
#include <App.hpp>
```

## **Public Member Functions**

- App (FlightNetwork &flightnetwork)
- void mainMenu ()
- void statisticsMenu ()
- void bestFlightMenu ()
- void goBackStatisticsMenu ()
- void globalStatistics ()
- void showNumFlights ()
- void numberOfDestinations (Airport &airport)
- void reachableDest (Airport & airport, int stops)

## **Private Attributes**

• FlightNetwork flightnetwork

## 4.3.1 Constructor & Destructor Documentation

## 4.3.1.1 App()

## 4.3.2 Member Function Documentation

## 4.3.2.1 bestFlightMenu()

```
void App::bestFlightMenu ( )
```

## 4.3.2.2 globalStatistics()

```
void App::globalStatistics ( )
```

## 4.3.2.3 goBackStatisticsMenu()

```
void App::goBackStatisticsMenu ( )
```

## 4.3.2.4 mainMenu()

```
void App::mainMenu ( )
```

## 4.3.2.5 numberOfDestinations()

## 4.3.2.6 reachableDest()

## 4.3.2.7 showNumFlights()

```
void App::showNumFlights ( )
```

## 4.3.2.8 statisticsMenu()

```
void App::statisticsMenu ( )
```

## 4.3.3 Member Data Documentation

## 4.3.3.1 flightnetwork

```
FlightNetwork App::flightnetwork [private]
```

The documentation for this class was generated from the following files:

- inc/App.hpp
- src/App.cpp

## 4.4 Edge < T > Class Template Reference

Represents an edge between two vertices in a graph with generic information of type T.

```
#include <Graph.hpp>
```

#### **Public Member Functions**

• Edge (Vertex< T > \*d, std::string in, double w)

Constructor for creating an edge with a specified destination vertex, information, and weight.

Vertex< T > \* getDest () const

Gets the destination vertex of the edge.

void setDest (Vertex< T > \*d)

Sets the destination vertex of the edge.

• std::string getInfo () const

Gets the information associated with the edge.

• void setInfo (std::string in)

Sets the information associated with the edge.

· double getWeight () const

Gets the weight of the edge.

void setWeight (double weight)

Sets the weight of the edge.

#### **Private Attributes**

- Vertex< T > \* dest
- std::string info
- · double weight

#### **Friends**

- class GraphT >
- class Vertex< T >

## 4.4.1 Detailed Description

```
template < class T > class Edge < T >
```

Represents an edge between two vertices in a graph with generic information of type T.

## **Template Parameters**

T | The type of information stored in the edge.

## 4.4.2 Constructor & Destructor Documentation

## 4.4.2.1 Edge()

Constructor for creating an edge with a specified destination vertex, information, and weight.

#### **Parameters**

d	The destination vertex of the edge.
in	The information associated with the edge.
W	The weight of the edge.

## 4.4.3 Member Function Documentation

## 4.4.3.1 getDest()

Gets the destination vertex of the edge.

## Returns

The destination vertex of the edge.

## 4.4.3.2 getInfo()

```
template<class T >
std::string Edge< T >::getInfo ( ) const
```

Gets the information associated with the edge.

#### Returns

The information associated with the edge.

## 4.4.3.3 getWeight()

```
\label{template} $$ $$ $$ template < class T > $$ double Edge < T > :: getWeight ( ) const
```

Gets the weight of the edge.

Returns

The weight of the edge.

## 4.4.3.4 setDest()

Sets the destination vertex of the edge.

#### **Parameters**

d The new destination vertex of the edge.

## 4.4.3.5 setInfo()

```
template<class T >
void Edge< T >::setInfo (
          std::string in )
```

Sets the information associated with the edge.

## **Parameters**

in The new information associated with the edge.

## 4.4.3.6 setWeight()

Sets the weight of the edge.

#### **Parameters**

weight	The new weight of the edge.

## 4.4.4 Friends And Related Symbol Documentation

## 4.4.4.1 Graph < T >

```
template<class T >
friend class Graph< T > [friend]
```

Allow Graph class to access private members of Edge.

#### 4.4.4.2 **Vertex**< T >

```
template<class T >
friend class Vertex< T > [friend]
```

Allow Vertex class to access private members of Edge.

## 4.4.5 Member Data Documentation

#### 4.4.5.1 dest

```
template<class T >
Vertex<T>* Edge< T >::dest [private]
```

The destination vertex of the edge.

#### 4.4.5.2 info

```
template<class T >
std::string Edge< T >::info [private]
```

Information associated with the edge.

#### 4.4.5.3 weight

```
template<class T >
double Edge< T >::weight [private]
```

The weight of the edge.

The documentation for this class was generated from the following file:

• inc/Graph.hpp

## 4.5 FlightNetwork Class Reference

Represents a flight network consisting of airports and flights.

```
#include <FlightNetwork.hpp>
```

#### **Public Member Functions**

FlightNetwork ()

Default constructor for the FlightNetwork class.

Time complexity: O(1).

• FlightNetwork (const std::string &airlines\_filename, const std::string &airports\_filename, const std::string &flights filename)

Parameterized constructor for the FlightNetwork class.

Time complexity: O(V + E), where V is the number of lines in the airports.csv file and E is the number of lines in the flights.csv file.

Graph < Airport > getAiportsGraph ()

Getter function to retrieve the airports graph.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

• int getGlobalNumOfAirports () const

Get the total number of airports in the network.

Time complexity: O(1).

· int getGlobalNumOfFlights () const

Get the total number of flights in the network.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

std::pair< int, int > numFlightsAirport (const Airport & airport)

Get the number of flights departing or arriving at a specific airport.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

int numFlightsCity (const std::string &city) const

Get the total number of flights departing or arriving in a specific city.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

• int numFlightsAirline (Airline &airline) const

Get the total number of flights operated by a specific airline.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

 $\bullet \ \ \mathsf{std} :: \mathsf{set} < \mathsf{std} :: \mathsf{string} > \mathsf{getDiffCountriesAirport} \ (\mathsf{const} \ \mathsf{Airport} \ \& \mathsf{airport}) \ \mathsf{const} \\$ 

Get the set of different countries connected to a specific airport.

Time complexity: O(N \* log(P)), where N is the size of the adjacency list and P is the size of the set.

• std::set< std::string > getDiffCountriesCity (const std::string &city) const

Get the set of different countries connected to airports in a specific city.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

• std::set< std::string > getAirportsDestinations (const Airport & airport) const

Get the set of airport names connected to a specific airport.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

std::set< std::string > getCitiesDestinations (const Airport & airport) const

Get the set of city names connected to a specific airport.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

• std::set< std::string > getCountriesDestinations (const Airport & airport) const

Get the set of country names connected to a specific airport.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

std::set< std::string > getReachableAirports (const Airport & airport, const int & distance)

Get the set of airport names reachable from a specific airport within a given distance.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

• std::set< std::string > getReachableCities (const Airport &airport, const int &distance)

Get the set of city names reachable from a specific airport within a given distance.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

• std::set< std::string > getReachableCountries (const Airport & airport, const int & distance)

Get the set of country names reachable from a specific airport within a given distance.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

int maximumTrip (std::vector< std::pair< std::string, std::string > > &airports)

Find the maximum number of stops for a round-trip connecting the given airports.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

std::set< std::string > getGreatestTraffic (const int &k)

Get the set of airports with the greatest traffic, considering both incoming and outgoing flights. Time complexity:  $O(V^2*E)$ , where V is the number of airports and E is the number of flights.

• std::set< std::string > getEssentialAirports ()

Perform a depth-first search to identify essential airports in the flight network.

Time complexity: O(V\*(V+E)), where V is the number of airports and E is the number of flights.

Airport codeCriteria (const std::string &code) const

Find an airport by its code.

Time Complexity: O(V) where V is the number of vertices in the graph.

Airport nameCriteria (const std::string &name) const

Find an airport by its name.

Time Complexity: O(V) where V is the number of vertices in the graph.

std::vector < Airport > cityCriteria (const std::string &city) const

Find airports in the specified city.

Time Complexity: O(V) where V is the number of vertices in the graph.

• std::vector< Airport > coordinateCriteria (const float &lat, const float &lon) const

Find airports near the specified coordinates.

Time Complexity: O(V) where V is the number of vertices in the graph.

std::vector< vector< Airport >> bestFlight (const Airport &source, const Airport &destination, const set
 string > &allowedAirlines={}, bool minimizeAirlines=false) const

Find the best flight paths between source and destination airports.

Time Complexity: O(V + E) where V is the number of vertices in the graph.

std::vector < std::vector < Airport > > listBestFlights (const int &flag1, const int &flag2, const set < string > &allowedAirlines, bool minimizeAirlines) const

List the best flights based on user input for source and destination.

Time Complexity: O(N \* M \* (V + E)) where V is the number of vertices in the graph, E is the number of flights, N and M is the number of origins and destinations.

std::string airportCodeToName (const std::string &code)

Convert an airport code to its corresponding name.

Time Complexity: O(V) where V is the number of vertices in the graph.

std::string airlineCodeToName (const std::string &code)

Convert an airline code to its corresponding name.

 $\label{thm:complexity:one} \textit{Time Complexity: } \textit{O(N) where N is the number of lines in the airlines.csv.}$ 

#### **Private Attributes**

Graph < Airport > airportsGraph

Graph representing the network of airports and flights.

## 4.5.1 Detailed Description

Represents a flight network consisting of airports and flights.

#### 4.5.2 Constructor & Destructor Documentation

## 4.5.2.1 FlightNetwork() [1/2]

```
FlightNetwork::FlightNetwork ( )
```

Default constructor for the FlightNetwork class.

Time complexity: O(1).

## 4.5.2.2 FlightNetwork() [2/2]

```
FlightNetwork::FlightNetwork (

const std::string & airlines_filename,

const std::string & airports_filename,

const std::string & flights_filename)
```

Parameterized constructor for the FlightNetwork class.

Time complexity: O(V + E), where V is the number of lines in the airports.csv file and E is the number of lines in the flights.csv file.

#### **Parameters**

airlines_filename	Filename for the airlines data.
airports_filename	Filename for the airports data.
flights_filename	Filename for the flights data.

#### 4.5.3 Member Function Documentation

## 4.5.3.1 airlineCodeToName()

Convert an airline code to its corresponding name.

Time Complexity: O(N) where N is the number of lines in the airlines.csv.

This function takes an airline code and searches the dataset to find the corresponding airline name.

#### **Parameters**

code	The unique ICAO code of the airline.

#### Returns

std::string The name of the airline.

#### **Exceptions**

std::runtime_error	if the airline with the given code is not found.
--------------------	--

## 4.5.3.2 airportCodeToName()

Convert an airport code to its corresponding name.

Time Complexity: O(V) where V is the number of vertices in the graph.

This function takes an airport code and searches the dataset to find the corresponding airport name.	

#### **Parameters**

code	The unique IATA code of the airport.
------	--------------------------------------

## Returns

std::string The name of the airport.

## **Exceptions**

	std::runtime_error	if the airport with the given code is not found.	
--	--------------------	--	--

## 4.5.3.3 bestFlight()

Find the best flight paths between source and destination airports.

Time Complexity: O(V + E) where V is the number of vertices in the graph.

#### **Parameters**

source	The source airport.
destination	The destination airport.
allowedAirlines	Set of allowed airlines (empty for any).
minimizeAirlines	Flag to minimize the number of unique airlines in the path.

#### Returns

Vector of vector of airports representing the best flight paths.

## 4.5.3.4 cityCriteria()

Find airports in the specified city.

Time Complexity: O(V) where V is the number of vertices in the graph.

#### **Parameters**

city	The name of the city to search for.
------	-------------------------------------

#### Returns

Vector of airports in the given city.

## **Exceptions**

runtime_error	if no city with the provided name is found.

## 4.5.3.5 codeCriteria()

Find an airport by its code.

Time Complexity: O(V) where V is the number of vertices in the graph.

#### **Parameters**

#### Returns

The airport matching the provided code.

#### **Exceptions**

runtime_error	if no airport is found with the given code.
---------------	---

## 4.5.3.6 coordinateCriteria()

Find airports near the specified coordinates.

Time Complexity: O(V) where V is the number of vertices in the graph.

#### **Parameters**

lat	The latitude of the target location.
lon	The longitude of the target location.

#### Returns

Vector of airports near the specified coordinates.

## **Exceptions**

<i>runtime_error</i> if no airports are found in the specified area.
--

## 4.5.3.7 getAiportsGraph()

```
Graph< Airport > FlightNetwork::getAiportsGraph ( )
```

Getter function to retrieve the airports graph.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### Returns

Graph<Airport> representing the airports and flights.

#### 4.5.3.8 getAirportsDestinations()

Get the set of airport names connected to a specific airport.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### **Parameters**

```
airport The target airport.
```

#### Returns

Set of airport names.

#### 4.5.3.9 getCitiesDestinations()

Get the set of city names connected to a specific airport.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### **Parameters**

airport	The target airport.

## Returns

Set of city names.

#### 4.5.3.10 getCountriesDestinations()

Get the set of country names connected to a specific airport.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### **Parameters**

airport	The target airport.
---------	---------------------

#### Returns

Set of country names.

#### 4.5.3.11 getDiffCountriesAirport()

Get the set of different countries connected to a specific airport.

Time complexity: O(N \* log(P)), where N is the size of the adjacency list and P is the size of the set.

#### **Parameters**

```
airport The target airport.
```

#### Returns

Set of countries.

#### 4.5.3.12 getDiffCountriesCity()

Get the set of different countries connected to airports in a specific city.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### **Parameters**

```
city The target city.
```

#### Returns

Set of countries.

#### 4.5.3.13 getEssentialAirports()

```
\verb|set<| \verb|string|| > \verb|FlightNetwork::getEssentialAirports|| ( )
```

Perform a depth-first search to identify essential airports in the flight network.

Time complexity: O(V\*(V+E)), where V is the number of airports and E is the number of flights.

#### Returns

Set of essential airport names.

#### 4.5.3.14 getGlobalNumOfAirports()

```
\verb|int FlightNetwork::getGlobalNumOfAirports ()| const|\\
```

Get the total number of airports in the network.

Time complexity: O(1).

#### Returns

Total number of airports.

#### 4.5.3.15 getGlobalNumOfFlights()

```
\verb|int FlightNetwork::getGlobalNumOfFlights ()| const|\\
```

Get the total number of flights in the network.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### Returns

Total number of flights.

## 4.5.3.16 getGreatestTraffic()

```
set< string > FlightNetwork::getGreatestTraffic ( const int & k )
```

Get the set of airports with the greatest traffic, considering both incoming and outgoing flights. Time complexity:  $O(V^2*E)$ , where V is the number of airports and E is the number of flights.

#### **Parameters**

*k* The number of airports to retrieve.

#### Returns

Set of airport names with the greatest traffic.

#### 4.5.3.17 getReachableAirports()

Get the set of airport names reachable from a specific airport within a given distance. Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### **Parameters**

airport	The source airport.
distance	The maximum distance to consider.

#### Returns

Set of reachable airport names.

#### 4.5.3.18 getReachableCities()

Get the set of city names reachable from a specific airport within a given distance. Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### **Parameters**

airport	The source airport.
distance	The maximum distance to consider.

#### Returns

Set of reachable city names.

#### 4.5.3.19 getReachableCountries()

Get the set of country names reachable from a specific airport within a given distance. Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### **Parameters**

airport	The source airport.
distance	The maximum distance to consider.

#### Returns

Set of reachable country names.

## 4.5.3.20 listBestFlights()

List the best flights based on user input for source and destination.

Time Complexity: O(N \* M \* (V + E)) where V is the number of vertices in the graph, E is the number of flights, N and M is the number of origins and destinations.

#### **Parameters**

flag1	Type of input for the source (1: code, 2: name, 3: city, 4: coordinate).
flag2	Type of input for the destination (1: code, 2: name, 3: city, 4: coordinate).
allowedAirlines	Set of allowed airlines (empty for any).
minimizeAirlines	Flag to minimize the number of unique airlines in the path.

#### Returns

Vector of vector of airports representing the best flight paths.

## 4.5.3.21 maximumTrip()

Find the maximum number of stops for a round-trip connecting the given airports.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### **Parameters**

airports	A vector of pairs representing airport codes for the round-trip.

#### Returns

The maximum number of stops for the round-trip.

### 4.5.3.22 nameCriteria()

Find an airport by its name.

Time Complexity: O(V) where V is the number of vertices in the graph.

#### **Parameters**

name The name of the ta	rget airport.
-------------------------	---------------

#### Returns

The airport matching the provided name.

#### **Exceptions**

runtime_error	if no airport is found with the given name.
---------------	---

#### 4.5.3.23 numFlightsAirline()

Get the total number of flights operated by a specific airline.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### **Parameters**

airline	The target airline.

#### Returns

Total number of flights.

#### 4.5.3.24 numFlightsAirport()

Get the number of flights departing or arriving at a specific airport.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### **Parameters**

airport	The target airport.

#### Returns

A pair containing the total number of flights and the number of different airlines.

#### 4.5.3.25 numFlightsCity()

Get the total number of flights departing or arriving in a specific city.

Time complexity: O(V + E), where V is the number of airports and E is the number of flights.

#### **Parameters**

```
city The target city.
```

#### Returns

Total number of flights.

#### 4.5.4 Member Data Documentation

#### 4.5.4.1 airportsGraph

```
Graph<Airport> FlightNetwork::airportsGraph [private]
```

Graph representing the network of airports and flights.

The documentation for this class was generated from the following files:

- inc/FlightNetwork.hpp
- src/FlightNetwork.cpp

## 4.6 Graph < T > Class Template Reference

Represents a generic graph with vertices of type T.

```
#include <Graph.hpp>
```

#### **Public Member Functions**

Vertex< T > \* findVertex (const T &in) const

Finds a vertex with a given information in the graph.

• int getNumVertex () const

Gets the number of vertices in the graph.

• bool addVertex (const T &in)

Adds a vertex with the given information to the graph.

bool removeVertex (const T &in)

Removes the vertex with the given information from the graph.

• bool addEdge (const T &sourc, const T &dest, const std::string &in, double w)

Adds an edge between vertices with source and destination information.

bool removeEdge (const T &sourc, const T &dest)

Removes the edge between vertices with source and destination information.

std::vector< Vertex< T > \* > getVertexSet () const

Gets the vector of vertices in the graph.

void dfsVisit (Vertex< T > \*v, std::vector< T > &res) const

Helper function for depth-first search traversal of the graph.

• std::vector< T > dfs () const

Performs depth-first search traversal of the graph.

std::vector< T > dfs (const T &source) const

Performs depth-first search traversal of the graph starting from a specific vertex.

std::vector< T > nodesAtDistanceDFS (const T &source, int k)

Finds nodes at a specific distance from a source vertex using depth-first search.

std::vector< Edge< T >> EdgesAtDistanceDFS (const T &source, int k)

Finds edges at a specific distance from a source vertex using depth-first search.

std::vector< T > bfs (const T &source) const

Performs breadth-first search traversal of the graph starting from a specific vertex.

• std::vector< std::pair< int, T >> bfsDistance (Vertex< T > \*source)

Performs breadth-first search traversal of the graph starting from a specific vertex and returns a vector of pairs containing the distance and information of each vertex.

void inDegree (Vertex< T > \*source)

Determines the in-degree of a specific vertex in the graph.

#### **Private Attributes**

std::vector< Vertex< T > \* > vertexSet

#### 4.6.1 Detailed Description

```
template<class T> class Graph< T>
```

Represents a generic graph with vertices of type T.

**Template Parameters** 

The type of information stored in the vertices of the graph.

## 4.6.2 Member Function Documentation

#### 4.6.2.1 addEdge()

Adds an edge between vertices with source and destination information.

#### **Parameters**

sourc	The information of the source vertex.
dest	The information of the destination vertex.
in	The information associated with the edge.
W	The weight of the edge.

#### Returns

True if the edge is added successfully, false if the source or destination vertex is not found.

## 4.6.2.2 addVertex()

Adds a vertex with the given information to the graph.

#### **Parameters**

```
in The information to be stored in the new vertex.
```

#### Returns

True if the vertex is added successfully, false if the vertex already exists.

## 4.6.2.3 bfs()

Performs breadth-first search traversal of the graph starting from a specific vertex.

#### **Parameters**

source	The information of the starting vertex.
--------	---

#### Returns

A vector containing the information of vertices in the order they are visited.

#### 4.6.2.4 bfsDistance()

Performs breadth-first search traversal of the graph starting from a specific vertex and returns a vector of pairs containing the distance and information of each vertex.

#### **Parameters**

source	The pointer to the starting vertex.
--------	-------------------------------------

#### Returns

A vector of pairs containing the distance and information of each vertex.

#### 4.6.2.5 dfs() [1/2]

```
template<class T > std::vector< T > Graph< T >::dfs ( ) const
```

Performs depth-first search traversal of the graph.

#### Returns

A vector containing the information of vertices in the order they are visited.

## 4.6.2.6 dfs() [2/2]

Performs depth-first search traversal of the graph starting from a specific vertex.

#### **Parameters**

source	The information of the starting vertex.
--------	---

#### Returns

A vector containing the information of vertices in the order they are visited.

#### 4.6.2.7 dfsVisit()

Helper function for depth-first search traversal of the graph.

#### **Parameters**

V	The pointer to the current vertex being visited.
res	The vector to store the information of vertices in the order they are visited.

#### 4.6.2.8 EdgesAtDistanceDFS()

```
template<class T > std::vector< Edge< T > > Graph< T >::EdgesAtDistanceDFS ( const T & source, int k )
```

Finds edges at a specific distance from a source vertex using depth-first search.

#### **Parameters**

source	The information of the source vertex.	
k	The distance from the source vertex.	

#### Returns

A vector containing the edges at the specified distance.

## 4.6.2.9 findVertex()

Finds a vertex with a given information in the graph.

#### **Parameters**

in The information to search for in the vertices.

#### Returns

Pointer to the vertex with the given information, or nullptr if not found.

#### 4.6.2.10 getNumVertex()

```
template<class T >
int Graph< T >::getNumVertex ( ) const
```

Gets the number of vertices in the graph.

#### Returns

The number of vertices in the graph.

#### 4.6.2.11 getVertexSet()

Gets the vector of vertices in the graph.

#### Returns

The vector of vertices in the graph.

#### 4.6.2.12 inDegree()

Determines the in-degree of a specific vertex in the graph.

#### **Parameters**

```
source The pointer to the vertex for which the in-degree is calculated.
```

#### 4.6.2.13 nodesAtDistanceDFS()

Finds nodes at a specific distance from a source vertex using depth-first search.

#### **Parameters**

source	The information of the source vertex.	
k	The distance from the source vertex.	

#### Returns

A vector containing the information of vertices at the specified distance.

### 4.6.2.14 removeEdge()

Removes the edge between vertices with source and destination information.

#### **Parameters**

sourc	The information of the source vertex.	
dest	The information of the destination vertex.	

#### Returns

True if the edge is removed successfully, false if the source or destination vertex is not found.

#### 4.6.2.15 removeVertex()

Removes the vertex with the given information from the graph.

#### **Parameters**

```
in The information of the vertex to be removed.
```

#### Returns

True if the vertex is removed successfully, false if the vertex is not found.

#### 4.6.3 Member Data Documentation

#### 4.6.3.1 vertexSet

```
template<class T >
std::vector<Vertex<T> *> Graph< T >::vertexSet [private]
```

The vector of vertices in the graph.

The documentation for this class was generated from the following file:

• inc/Graph.hpp

## 4.7 Vertex < T > Class Template Reference

Represents a vertex in a graph with generic information of type T.

```
#include <Graph.hpp>
```

#### **Public Member Functions**

• Vertex (T in)

Constructor for creating a vertex with the given information.

• T getInfo () const

Gets the information stored in the vertex.

· void setInfo (T in)

Sets the information stored in the vertex.

• bool isVisited () const

Checks if the vertex has been visited during graph traversal.

void setVisited (bool v)

Sets the visited status of the vertex.

bool isProcessing () const

Checks if the vertex is currently being processed during traversal.

void setProcessing (bool p)

Sets the processing status of the vertex.

• void setInDegree (int i)

Sets the indegree of the vertex.

• int getInDegree ()

Gets the indegree of the vertex.

• int getNum () const

Gets the numeric identifier of the vertex.

void setNum (int num)

Sets the numeric identifier of the vertex.

• int getLow () const

Gets the low value of the vertex.

void setLow (int low)

Sets the low value of the vertex.

• int getDistance () const

Gets the distance of the vertex.

void setDistance (int distance)

Sets the distance of the vertex.

void addEdge (Vertex< T > \*d, std::string in, double w)

Adds an edge from this vertex to the specified destination vertex with a given weight.

bool removeEdgeTo (Vertex< T > \*d)

Removes the edge from this vertex to the specified destination vertex.

const std::vector< Edge< T >> & getAdj () const

Gets the vector of edges adjacent to this vertex.

void setAdj (const std::vector< Edge< T >> &adj\_vec)

Sets the vector of edges adjacent to this vertex.

#### **Private Attributes**

- T info
- std::vector< Edge< T > > adj
- · bool visited
- bool processing
- · int indegree
- int num
- int low
- int distance

#### **Friends**

class GraphT >

## 4.7.1 Detailed Description

```
\label{template} \begin{split} \text{template} &< \text{class T}> \\ \text{class Vertex} &< \text{T}> \end{split}
```

Represents a vertex in a graph with generic information of type T.

**Template Parameters** 

T | The type of information stored in the vertex.

## 4.7.2 Constructor & Destructor Documentation

## 4.7.2.1 Vertex()

Constructor for creating a vertex with the given information.

#### **Parameters**

in The information to be stored in the vertex.

## 4.7.3 Member Function Documentation

#### 4.7.3.1 addEdge()

```
template < class T >
void Vertex < T >::addEdge (
```

```
Vertex< T > * d,
std::string in,
double w )
```

Adds an edge from this vertex to the specified destination vertex with a given weight.

#### **Parameters**

d	The destination vertex.
in	The information associated with the edge.
W	The weight of the edge.

#### 4.7.3.2 getAdj()

```
template<class T > const std::vector< Edge< T > > & Vertex< T >::getAdj ( ) const
```

Gets the vector of edges adjacent to this vertex.

#### Returns

The vector of edges adjacent to this vertex.

## 4.7.3.3 getDistance()

```
template<class T >
int Vertex< T >::getDistance ( ) const
```

Gets the distance of the vertex.

#### Returns

The distance of the vertex.

#### 4.7.3.4 getInDegree()

```
template<class T >
int Vertex< T >::getInDegree ( )
```

Gets the indegree of the vertex.

#### Returns

The indegree of the vertex.

#### 4.7.3.5 getInfo()

```
template<class T > T Vertex< T >::getInfo ( ) const
```

Gets the information stored in the vertex.

Returns

The information stored in the vertex.

#### 4.7.3.6 getLow()

```
template<class T >
int Vertex< T >::getLow ( ) const
```

Gets the low value of the vertex.

**Returns** 

The low value of the vertex.

#### 4.7.3.7 getNum()

```
template<class T >
int Vertex< T >::getNum ( ) const
```

Gets the numeric identifier of the vertex.

Returns

The numeric identifier of the vertex.

#### 4.7.3.8 isProcessing()

```
template<class T > bool Vertex< T >:: isProcessing ( ) const
```

Checks if the vertex is currently being processed during traversal.

Returns

True if the vertex is being processed, false otherwise.

#### 4.7.3.9 isVisited()

```
template<class T >
bool Vertex< T >::isVisited ( ) const
```

Checks if the vertex has been visited during graph traversal.

Returns

True if the vertex has been visited, false otherwise.

## 4.7.3.10 removeEdgeTo()

Removes the edge from this vertex to the specified destination vertex.

#### **Parameters**

d The destination vertex.

#### Returns

True if the edge was successfully removed, false otherwise.

#### 4.7.3.11 setAdj()

Sets the vector of edges adjacent to this vertex.

#### **Parameters**

adj\_vec | The new vector of edges adjacent to this vertex.

## 4.7.3.12 setDistance()

Sets the distance of the vertex.

#### **Parameters**

distance The new distance of the vertex.

#### 4.7.3.13 setInDegree()

```
\label{template} $$\operatorname{template}<\operatorname{class} T>$$ $\operatorname{void} \ \operatorname{Vertex}< T>::\operatorname{setInDegree} \ ($$ \inf i )$$
```

Sets the indegree of the vertex.

#### **Parameters**

*i* The new indegree of the vertex.

#### 4.7.3.14 setInfo()

Sets the information stored in the vertex.

#### Parameters

in The new information to be stored in the vertex.

## 4.7.3.15 setLow()

```
template<class T >
void Vertex< T >::setLow (
    int low )
```

Sets the low value of the vertex.

#### **Parameters**

low The new low value.

## 4.7.3.16 setNum()

Sets the numeric identifier of the vertex.

#### **Parameters**

num The new numeric identifier.

## 4.7.3.17 setProcessing()

Sets the processing status of the vertex.

#### **Parameters**

p The new processing status.

#### 4.7.3.18 setVisited()

```
template<class T > void Vertex< T >::setVisited ( bool v )
```

Sets the visited status of the vertex.

#### **Parameters**

```
v The new visited status.
```

## 4.7.4 Friends And Related Symbol Documentation

## 4.7.4.1 Graph < T >

#### 4.7.5 Member Data Documentation

#### 4.7.5.1 adj

```
template<class T >
std::vector<Edge<T> > Vertex< T >::adj [private]
```

The vector of edges adjacent to this vertex.

#### 4.7.5.2 distance

```
template<class T >
int Vertex< T >::distance [private]
```

Distance of the vertex in certain graph traversal algorithms.

#### 4.7.5.3 indegree

```
template<class T >
int Vertex< T >::indegree [private]
```

The indegree of the vertex in a directed graph.

#### 4.7.5.4 info

```
template<class T >
T Vertex< T >::info [private]
```

The information stored in the vertex.

#### 4.7.5.5 low

```
template<class T >
int Vertex< T >::low [private]
```

Low value used in Tarjan's algorithm for finding strongly connected components.

#### 4.7.5.6 num

```
template<class T >
int Vertex< T >::num [private]
```

Numeric identifier for the vertex.

## 4.7.5.7 processing

```
template < class T >
bool Vertex < T >::processing [private]
```

Flag indicating if the vertex is currently being processed during traversal.

#### 4.7.5.8 visited

```
template<class T >
bool Vertex< T >::visited [private]
```

Flag indicating if the vertex has been visited during graph traversal.

The documentation for this class was generated from the following file:

• inc/Graph.hpp

# **Chapter 5**

# **File Documentation**

## 5.1 inc/Airline.hpp File Reference

```
#include <string>
```

#### Classes

· class Airline

Represents an airline with specific attributes such as code, name, callsign, and country.

## 5.2 Airline.hpp

#### Go to the documentation of this file.

```
00001 #ifndef AIRLINE_H
00002 #define AIRLINE_H
00003
00004 #include <string>
00005
00009 class Airline
00010 {
00012
          std::string code;
00013
00015
          std::string name;
00016
00018
          std::string callsign;
00019
00021
          std::string country;
00022
00023 public:
00031
         Airline(const std::string code, const std::string name, const std::string callsign, const
     std::string country);
00032
00037
          Airline(const std::string &code);
00038
00043
          std::string getCode() const;
00044
00049
          void setCode(const std::string &c);
00050
00055
          std::string getName() const;
00056
00061
          void setName(const std::string &n);
00062
00067
          std::string getCallsign() const;
00068
00073
          void setCallsign(const std::string &c);
00074
00079
          std::string getCountry() const;
08000
00085
          void setCountry(const std::string &c);
00086 };
00087
00088 #endif
```

## 5.3 inc/Airport.hpp File Reference

```
#include <string>
#include <utility>
```

#### Classes

· class Airport

Represents an airport with attributes such as code, name, city, country, and position.

## 5.4 Airport.hpp

## Go to the documentation of this file.

```
00001 #ifndef AIRPORT_H
00002 #define AIRPORT_H
00003
00004 #include <string>
00005 #include <utility>
00006
00010 class Airport
00011 {
00013
          std::string code;
00014
00016
          std::string name;
00017
00019
          std::string city;
00020
00022
          std::string country;
00023
00025
          std::pair<float, float> position;
00026
00027 public:
00036
          Airport(const std::string &code, const std::string &name, const std::string &city,
00037
                  const std::string &country, const std::pair<float, float> &position);
00038
00043
          Airport (const std::string &code);
00044
00049
          std::string getCode() const;
00050
00055
          void setCode(const std::string &c);
00056
00061
          std::string getName() const;
00062
00067
          void setName(const std::string &n);
00068
00073
          std::string getCity() const;
00074
00079
          void setCity(const std::string &c);
08000
00085
          std::string getCountry() const;
00086
00091
          void setCountry(const std::string &c);
00092
00097
          std::pair<float, float> getPosition() const;
00098
00103
          void setPosition(const std::pair<float, float> &pos);
00104
          bool operator==(const Airport &other);
00110
00111
00117
          bool operator<(const Airport &other);</pre>
00118 };
00119
00120 #endif
```

## 5.5 inc/App.hpp File Reference

```
#include "FlightNetwork.hpp"
#include <cstdlib>
```

5.6 App.hpp 51

#### **Classes**

class App

#### **Functions**

• void clearScreen ()

#### 5.5.1 Function Documentation

#### 5.5.1.1 clearScreen()

```
void clearScreen ( )
```

# 5.6 App.hpp

#### Go to the documentation of this file.

```
00001 #ifndef APP_H
00002 #define APP_H
00003
00004 #include "FlightNetwork.hpp"
00005 #include <cstdlib>
00006
00007 void clearScreen();
80000
00009 class App
00010 {
          FlightNetwork flightnetwork;
00012
00013 public:
        App(FlightNetwork &flightnetwork);
00014
00015
          void mainMenu();
00016
00017
          void statisticsMenu();
00018
          void bestFlightMenu();
00019
          void goBackStatisticsMenu();
00020
          void globalStatistics();
          void showNumFlights();
void numberOfDestinations(Airport &airport);
00021
00022
00023
          void reachableDest(Airport & airport, int stops);
00024
00025 };
00026
00027 #endif
```

## 5.7 inc/FlightNetwork.hpp File Reference

```
#include "Airline.hpp"
#include "Airport.hpp"
#include "Graph.hpp"
#include <iostream>
#include <fstream>
#include <sstream>
#include <set>
#include <cmath>
#include <tuple>
#include <functional>
#include #include #include #include 
#include <functional>
```

#### Classes

class FlightNetwork

Represents a flight network consisting of airports and flights.

#### **Functions**

```
    void dfs_art (Graph < Airport > &g, Vertex < Airport > *v, set < string > &l, int &i)
    Depth-first search for identifying articulation points in the airport graph.
    Time Complexity: O(V + E) where V is the number of vertices in the graph.
```

• double haversineDistance (double lat1, double lon1, double lat2, double lon2)

Calculates the Haversine distance between two geographical points. Time Complexity: O(1)

#### 5.7.1 Function Documentation

#### 5.7.1.1 dfs\_art()

Depth-first search for identifying articulation points in the airport graph.

Time Complexity: O(V + E) where V is the number of vertices in the graph.

#### **Parameters**

g	The airport graph.  The current vertex in the DFS traversal.	
V		
1	Set to store the identified articulation points.	
i	Reference to the current DFS iteration number.	

#### 5.7.1.2 haversineDistance()

Calculates the Haversine distance between two geographical points.

Time Complexity: O(1)

#### **Parameters**

lat1	Latitude of the first point.
lon1	Longitude of the first point.
lat2	Latitude of the second point.
lon2	Longitude of the second point.

5.8 FlightNetwork.hpp 53

#### Returns

Haversine distance between the two points.

## 5.8 FlightNetwork.hpp

#### Go to the documentation of this file.

```
00001 #ifndef FLIGHT_NETWORK_H
00002 #define FLIGHT_NETWORK_H
00003
00004 #include "Airline.hpp'
00005 #include "Airport.hpp"
00006 #include "Graph.hpp"
00007 #include <iostream>
00008 #include <fstream>
00009 #include <sstream>
00010 #include <set>
00011 #include <stack>
00012 #include <cmath>
00013 #include <tuple>
00014 #include <functional>
00015 #include <limits>
00016
00020 class FlightNetwork
00021 {
00025
          Graph<Airport> airportsGraph;
00026
00027 public:
00032
          FlightNetwork();
          FlightNetwork(const std::string &airlines_filename, const std::string &airports_filename, const
00042
     std::string &flights_filename);
00043
00050
          Graph<Airport> getAiportsGraph();
00051
00058
          int getGlobalNumOfAirports() const; // 3) i.
00059
00066
          int getGlobalNumOfFlights() const; // 3) i.
00067
00075
          std::pair<int, int> numFlightsAirport(const Airport &airport); // 3) ii.
00076
00084
          int numFlightsCity(const std::string &city) const; // 3) iii.
00085
00093
          int numFlightsAirline(Airline &airline) const; // 3) iii.
00094
00102
          std::set<std::string> getDiffCountriesAirport(const Airport &airport) const; // 3) iv.
00103
          std::set<std::string> getDiffCountriesCity(const std::string &city) const; // 3) iv.
00111
00112
00120
          std::set<std::string> getAirportsDestinations(const Airport &airport) const; // 3) v.
00121
00129
          std::set<std::string> getCitiesDestinations(const Airport &airport) const; // 3) v.
00130
00138
          std::set<std::string> getCountriesDestinations(const Airport &airport) const; // 3) v.
00139
00148
          std::set<std::string> getReachableAirports(const Airport & airport, const int & distance); // 3) vi.
00149
00158
          std::set<std::string> getReachableCities(const Airport & airport, const int & distance); // 3) vi.
00159
00168
          std::set<std::string> getReachableCountries(const Airport & airport, const int & distance); // 3)
00169
00177
          int maximumTrip(std::vector<std::pair<std::string, std::string» &airports); // 3) vii.
00178
          std::set<std::string> getGreatestTraffic(const int &k); // 3) viii.
00186
00187
00194
          std::set<std::string> getEssentialAirports(); // 3) ix.
00195
00204
          Airport codeCriteria(const std::string &code) const; // 4) i.
00205
00214
          Airport nameCriteria(const std::string &name) const; // 4) i.
00215
00224
          std::vector<Airport> cityCriteria(const std::string &city) const; // 4) ii.
00225
00235
          std::vector<Airport> coordinateCriteria(const float &lat, const float &lon) const; // 4) iii.
00236
          std::vector<vector<Airport» bestFlight(const Airport &source, const Airport &destination, const
00247
      set<string> &allowedAirlines = {}, bool minimizeAirlines = false) const; // 4)
00248
          std::vector<std::vector<Airport» listBestFlights(const int &flag1, const int &flag2, const
      set<string> &allowedAirlines, bool minimizeAirlines) const;
```

```
00260
00271    std::string airportCodeToName(const std::string &code);
00272
00283    std::string airlineCodeToName(const std::string &code);
00284 };
00285
00295 void dfs_art(Graph<Airport> &g, Vertex<Airport> *v, set<string> &l, int &i);
00296
00306 double haversineDistance(double lat1, double lon1, double lat2, double lon2);
00307
00308 #endif
```

## 5.9 inc/Graph.hpp File Reference

```
#include <string>
#include <vector>
#include <queue>
#include <unordered_map>
```

#### **Classes**

class Vertex< T >

Represents a vertex in a graph with generic information of type T.

class Edge< T >

Represents an edge between two vertices in a graph with generic information of type T.

class GraphT >

Represents a generic graph with vertices of type T.

#### **Functions**

```
    template < class T > void nodesAtDistanceDFSVisit (Vertex < T > *v, int k, std::vector < T > &res)
    template < class T > void nodesAtDistanceDFSVisit (Vertex < T > *v, int k, std::vector < Edge < T > > &res)
```

#### 5.9.1 Function Documentation

#### 5.9.1.1 nodesAtDistanceDFSVisit() [1/2]

#### 5.9.1.2 nodesAtDistanceDFSVisit() [2/2]

5.10 Graph.hpp 55

## 5.10 Graph.hpp

#### Go to the documentation of this file.

```
00001 #ifndef GRAPH_H
00002 #define GRAPH_H
00003
00004 #include <string>
00005 #include <vector>
00006 #include <queue>
00007 #include <unordered_map>
00008 using namespace std;
00009
00010 template <class T>
00011 class Edge;
00012
00013 template <class T>
00014 class Graph;
00015
00016 template <class T>
00017 class Vertex;
00018
00023 template <class T>
00024 class Vertex
00025 {
00027
          T info;
00028
00030
          std::vector<Edge<T» adj;</pre>
00031
00033
          bool visited;
00034
00036
          bool processing;
00037
00039
          int indegree;
00040
00042
          int num;
00043
00045
          int low;
00046
00048
          int distance;
00049
00050 public:
          Vertex(T in);
00055
00056
00061
          T getInfo() const;
00062
00067
          void setInfo(T in);
00068
00073
          bool isVisited() const;
00074
00079
          void setVisited(bool v);
08000
00085
          bool isProcessing() const;
00086
00091
          void setProcessing(bool p);
00092
00097
          void setInDegree(int i);
00098
00103
          int getInDegree();
00104
00109
          int getNum() const;
00110
00115
          void setNum(int num);
00116
00121
          int getLow() const;
00122
00127
          void setLow(int low);
00128
00133
          int getDistance() const;
00134
00139
          void setDistance(int distance);
00140
00147
          void addEdge(Vertex<T> *d, std::string in, double w);
00148
          bool removeEdgeTo(Vertex<T> *d);
00154
00155
          const std::vector<Edge<T> &getAdj() const;
00160
00161
00166
          void setAdj(const std::vector<Edge<T>> &adj_vec);
00167
          friend class Graph<T>; // Allow Graph class to access private members of Vertex.
00168
00169 };
00170
00175 template <class T>
00176 class Edge
00177 {
```

```
00179
          Vertex<T> *dest;
00180
00182
          std::string info;
00183
00185
          double weight;
00186
00187 public:
00194
          Edge(Vertex<T> *d, std::string in, double w);
00195
00200
          Vertex<T> *getDest() const;
00201
00206
          void setDest(Vertex<T> *d);
00207
00212
          std::string getInfo() const;
00213
00218
          void setInfo(std::string in);
00219
00224
          double getWeight() const;
00225
00230
          void setWeight(double weight);
00231
00232
          friend class Graph<T>;
00233
         friend class Vertex<T>;
00234 };
00235
00240 template <class T>
00241 class Graph
00242 {
00244
          std::vector<Vertex<T> *> vertexSet;
00245
00246 public:
00252
          Vertex<T> *findVertex(const T &in) const;
00253
00259
          int getNumVertex() const;
00260
          bool addVertex(const T &in);
00266
00267
00273
          bool removeVertex(const T &in);
00274
00283
          bool addEdge(const T &sourc, const T &dest, const std::string &in, double w);
00284
00291
          bool removeEdge (const T &sourc, const T &dest);
00292
00297
          std::vector<Vertex<T> *> getVertexSet() const;
00298
00304
          void dfsVisit(Vertex<T> *v, std::vector<T> &res) const;
00305
00310
          std::vector<T> dfs() const;
00311
00317
          std::vector<T> dfs(const T &source) const;
00318
00325
          std::vector<T> nodesAtDistanceDFS(const T &source, int k);
00326
00333
          std::vector<Edge<T> EdgesAtDistanceDFS(const T &source, int k);
00334
00340
          std::vector<T> bfs(const T &source) const;
00341
00348
          std::vector<std::pair<int, T> bfsDistance(Vertex<T> *source);
00349
00354
          void inDegree(Vertex<T> *source);
00355 };
00356
00357 /*
00358
         Vertex functions
00359 */
00360
00361 template <class T>
00362 Vertex<T>::Vertex(T in) : info(in), visited(false), processing(false) {}
00363
00364 template <class T>
00365 T Vertex<T>::getInfo() const
00366 {
00367
          return info;
00368 }
00369
00370 template <class T>
00371 void Vertex<T>::setInfo(T in)
00372 {
00373
          info = in:
00374 }
00375
00376 template <class T>
00377 bool Vertex<T>::isVisited() const
00378 {
00379
          return visited;
00380 }
00381
```

5.10 Graph.hpp 57

```
00382 template <class T>
00383 void Vertex<T>::setVisited(bool v)
00384 {
00385
         visited = v;
00386 }
00387
00388 template <class T>
00389 bool Vertex<T>::isProcessing() const
00390 {
00391
          return processing;
00392 }
00393
00394 template <class T>
00395 void Vertex<T>::setProcessing(bool p)
00396 {
00397
         processing = p;
00398 }
00399
00400 template <class T>
00401 void Vertex<T>::setInDegree(int i)
00402 {
00403
          indegree = i;
00404 }
00405
00406 template <class T>
00407 int Vertex<T>::getInDegree()
00408 {
00409
          return indegree;
00410 }
00411
00412 template <class T>
00413 int Vertex<T>::getNum() const
00414 {
00415
          return num;
00416 }
00417
00418 template <class T>
00419 void Vertex<T>::setNum(int num)
00420 {
00421
         Vertex::num = num;
00422 }
00423
00424 template <class T>
00425 int Vertex<T>::getLow() const
00426 {
00427
          return low;
00428 }
00429
00430 template <class T>
00431 void Vertex<T>::setDistance(int distance)
00432 {
00433
         Vertex::distance = distance;
00434 }
00435
00436 template <class T>
00437 int Vertex<T>::getDistance() const
00438 {
00439
         return distance;
00440 }
00441
00442 template <class T>
00443 void Vertex<T>::setLow(int low)
00444 {
00445
          Vertex::low = low;
00446 }
00447
00448 template <class T>
00449 void Vertex<T>::addEdge(Vertex<T> *d, std::string in, double w)
00450 {
00451
         adj.push_back(Edge<T>(d, in, w));
00452 }
00453
00454 template <class T>
00455 bool Vertex<T>::removeEdgeTo(Vertex<T> *d)
00456 {
00457
          for (auto it = adj.begin(); it != adj.end(); it++)
00458
              if (it->dest == d)
00459
00460
                  adj.erase(it);
00461
                  return true;
00462
             }
00463
         return false;
00464 }
00465
00466 template <class T>
00467 const std::vector<Edge<T» &Vertex<T>::getAdj() const
00468 {
```

```
00469
         return adj;
00470 }
00471
00472 template <class T>
00473 void Vertex<T>::setAdj(const std::vector<Edge<T>> &adj_vec)
00474 {
00475
          adj = adj_vec;
00476 }
00477
00478 /*
00479
         Edge functions
00480 */
00481
00482 template <class T>
00483 Edge<T>::Edge(Vertex<T> *d, std::string in, double w) : dest(d), info(in), weight(w) {}
00484
00485 template <class T>
00486 Vertex<T> *Edge<T>::getDest() const
00487 {
00488
         return dest;
00489 }
00490
00491 template <class T>
00492 void Edge<T>::setDest(Vertex<T> *d)
00493 {
00494
          dest = d;
00495 }
00496
00497 template <class T>
00498 std::string Edge<T>::getInfo() const
00499 {
00500
          return info;
00501 }
00502
00503 template <class T>
00504 void Edge<T>::setInfo(std::string in)
00505 {
         info = in;
00507 }
00508
00509 template <class T>
00510 double Edge<T>::getWeight() const
00511 {
00512
         return weight;
00513 }
00514
00515 template <class T>
00516 void Edge<T>::setWeight(double weight)
00517 {
00518
          Edge::weight = weight;
00519 }
00520
00521 /*
00522
         Graph functions
00523 */
00524
00525 template <class T>
00526 Vertex<T> *Graph<T>::findVertex(const T &in) const
00527 {
00528
          for (auto v : vertexSet)
           if (v->info == in)
00529
00530
                  return v;
00531
         return NULL;
00532 }
00533
00534 template <class T>
00535 int Graph<T>::getNumVertex() const
00536 {
00537
          return vertexSet.size();
00538 }
00539
00540 template <class T>
00541 bool Graph<T>::addVertex(const T &in)
00542 {
00543
          if (findVertex(in) != NULL)
00544
              return false;
00545
         vertexSet.push_back(new Vertex<T>(in));
00546
         return true;
00547 }
00548
00549 template <class T>
00550 bool Graph<T>::removeVertex(const T &in)
00551 {
00552
          for (auto it = vertexSet.begin(); it != vertexSet.end(); it++)
00553
              if ((*it)->info == in)
00554
              {
00555
                  auto v = *it:
```

5.10 Graph.hpp 59

```
vertexSet.erase(it);
00557
                  for (auto u : vertexSet)
00558
                      u->removeEdgeTo(v);
                  delete v;
00559
00560
                  return true;
00561
00562
         return false;
00563 }
00564
00565 template <class T>
00566 bool Graph<T>::addEdge(const T &sourc, const T &dest, const std::string &in, double w)
00567 {
00568
         auto v1 = findVertex(sourc);
         auto v2 = findVertex(dest);
00569
00570
         if (v1 == NULL || v2 == NULL)
              return false;
00571
00572
         v1->addEdge(v2, in, w);
00573
         return true;
00574 }
00575
00576 template <class T>
00577 bool Graph<T>:::removeEdge(const T &sourc, const T &dest)
00578 {
00579
         auto v1 = findVertex(sourc);
00580
         auto v2 = findVertex(dest);
00581
         if (v1 == NULL | | v2 == NULL)
00582
              return false;
00583
         return v1->removeEdgeTo(v2);
00584 }
00585
00586 template <class T>
00587 std::vector<Vertex<T> *> Graph<T>::getVertexSet() const
00588 {
00589
          return vertexSet;
00590 }
00591
00592 template <class T>
00593 void Graph<T>::dfsVisit(Vertex<T> *v, std::vector<T> &res) const
00594 {
00595
          v->setVisited(true);
00596
         res.push_back(v->getInfo());
00597
00598
          for (const Edge<T> &edge : v->getAdj())
00599
00600
              Vertex<T> *neighbor = edge.getDest();
00601
00602
              if (!neighbor->isVisited())
00603
              {
00604
                  dfsVisit(neighbor, res);
00605
00606
          }
00607 }
00608
00609 template <class T>
00610 std::vector<T> Graph<T>::dfs() const
00611 {
00612
          std::vector<T> res;
00613
00614
          for (Vertex<T> *v : vertexSet)
00615
              if (!v->isVisited())
00616
00617
              {
00618
                  dfsVisit(v, res);
00619
00620
          }
00621
          for (Vertex<T> *v : vertexSet)
00622
00623
00624
             v->setVisited(false);
00625
         }
00626
00627
          return res;
00628 }
00629
00630 template <class T>
00631 std::vector<T> Graph<T>::dfs(const T &source) const
00632 {
00633
          std::vector<T> res;
00634
         res.push_back(source);
00635
00636
          Vertex<T> *source vertex = findVertex(source);
00637
          source_vertex->setVisited(true);
00638
00639
          for (const Edge<T> &e : source_vertex->getAdj())
00640
             Vertex<T> *neighbor = e.getDest();
00641
00642
```

```
if (!neighbor->isVisited())
00644
                  dfsVisit(neighbor, res);
00645
          }
00646
00647
          for (Vertex<T> *vtx : vertexSet)
00648
              vtx->setVisited(false);
00649
00650
00651
00652
          return res;
00653 }
00654
00655 template <class T>
00656 std::vector<T> Graph<T>::nodesAtDistanceDFS(const T &source, int k)
00657 {
00658
          std::vector<T> res;
00659
          Vertex<T> *aux;
00660
00661
          for (auto v : vertexSet)
00662
          {
00663
              v->setVisited(false);
00664
00665
00666
          aux = this->findVertex(source);
00667
00668
          nodesAtDistanceDFSVisit(aux, k, res);
00669
00670
          return res;
00671 }
00672
00673 template <class T>
00674 void nodesAtDistanceDFSVisit(Vertex<T> *v, int k, std::vector<T> &res)
00675 {
00676
          v->setVisited(true);
00677
          if (k == 0)
00678
00679
              res.push_back(v->getInfo());
00680
              return;
00681
00682
          for (Edge<T> e : v->getAdj())
00683
00684
              auto w = e.getDest();
              if (!w->isVisited())
00685
00686
              {
00687
                  nodesAtDistanceDFSVisit(w, k - 1, res);
00688
00689
          }
00690 }
00691
00692 template <class T>
00693 void nodesAtDistanceDFSVisit(Vertex<T> *v, int k, std::vector<Edge<T>> &res)
00694 {
00695
          v->setVisited(true);
00696
          if (k == 0)
00697
00698
              for (auto aux : v->getAdj())
00699
00700
                  res.push_back(aux);
00701
00702
              return;
00703
00704
          for (Edge<T> e : v->getAdj())
00705
00706
              auto w = e.getDest();
00707
              if (!w->isVisited())
00708
              {
00709
                  nodesAtDistanceDFSVisit(w, k - 1, res);
00710
              }
00711
          }
00712 }
00713
00714 template <class T>
00715 std::vector<Edge<T» Graph<T>::EdgesAtDistanceDFS(const T &source, int k)
00716 {
00717
          std::vector<Edge<T» res;
00718
          Vertex<T> *aux;
00719
00720
          for (auto v : vertexSet)
00721
00722
              v->setVisited(false):
00723
00724
00725
          aux = this->findVertex(source);
00726
00727
          nodesAtDistanceDFSVisit(aux, k, res);
00728
00729
          return res;
```

5.10 Graph.hpp 61

```
00730 }
00731
00732 template <class T>
00733 std::vector<T> Graph<T>::bfs(const T &source) const
00734 {
00735
          std::vector<T> res;
00736
          std::queue<Vertex<T> *> aux;
00737
00738
          for (Vertex<T> *v : vertexSet)
00739
              v->setVisited(false);
00740
00741
          Vertex<T> *source vertex = findVertex(source);
00742
          source_vertex->setVisited(true);
00743
          aux.push(source_vertex);
00744
00745
          while (!aux.empty())
00746
00747
              Vertex<T> *curr = aux.front();
00748
              aux.pop();
00749
              res.push_back(curr->getInfo());
00750
00751
              for (const Edge<T> &e : curr->getAdj())
00752
00753
                  Vertex<T> *neighbor = e.getDest();
00754
                  if (!neighbor->isVisited())
00755
                      neighbor->setVisited(true);
00756
00757
                      aux.push(neighbor);
00758
                  }
00759
              }
00760
          }
00761
00762
          return res;
00763 }
00764
00765 template <class T>
00766 std::vector<std::pair<int, T» Graph<T>::bfsDistance(Vertex<T> *source)
00768
          std::vector<std::pair<int, T» res;
00769
          std::queue<Vertex<T> *> aux;
00770
00771
          for (Vertex<T> *v : vertexSet)
00772
00773
              v->setVisited(false);
00774
              v->setDistance(10000);
00775
00776
00777
          source->setVisited(true);
00778
          source->setDistance(0);
00779
          aux.push(source);
00780
00781
          while (!aux.empty())
00782
00783
              Vertex<T> *curr = aux.front();
00784
              aux.pop();
00785
              res.push_back({curr->getDistance(), curr->getInfo()});
00786
00787
              for (const Edge<T> &e : curr->getAdj())
00788
00789
                  Vertex<T> *neighbor = e.getDest();
00790
                  if (!neighbor->isVisited())
00791
                  {
00792
                      neighbor->setVisited(true);
00793
                      neighbor->setDistance(curr->getDistance() + 1);
00794
                      aux.push(neighbor);
00795
                  }
00796
              }
00797
          }
00798
00799
          return res;
00800 }
00801 template <class T>
00802 void Graph<T>::inDegree(Vertex<T> *source)
00803 {
00804
          int res = 0;
00805
          if (source == NULL)
00806
00807
00808
          for (auto it = vertexSet.begin(); it != vertexSet.end(); ++it)
00809
          {
              Vertex<T> *aux = *it;
00810
00811
              std::vector<Edge<T> adj = aux->getAdj();
00812
              for (auto ed : adj)
00813
              {
00814
                  if (ed.getDest() == source)
00815
                      res++;
00816
              }
```

```
00817    }
00818    source->setInDegree(res);
00819 }
00820
00821 #endif
```

## 5.11 README.md File Reference

## 5.12 src/Airline.cpp File Reference

```
#include "../inc/Airline.hpp"
```

## 5.13 src/Airport.cpp File Reference

```
#include "../inc/Airport.hpp"
```

## 5.14 src/App.cpp File Reference

```
#include "../inc/App.hpp"
```

#### **Functions**

• void clearScreen ()

### 5.14.1 Function Documentation

## 5.14.1.1 clearScreen()

```
void clearScreen ( )
```

## 5.15 src/FlightNetwork.cpp File Reference

```
#include "../inc/FlightNetwork.hpp"
#include <limits.h>
```

#### **Functions**

• double haversineDistance (double lat1, double lon1, double lat2, double lon2)

Calculates the Haversine distance between two geographical points. Time Complexity: O(1)

void dfs\_art (Graph< Airport > &g, Vertex< Airport > \*v, set< string > &I, int &i)

Depth-first search for identifying articulation points in the airport graph. Time Complexity: O(V + E) where V is the number of vertices in the graph.

#### 5.15.1 Function Documentation

#### 5.15.1.1 dfs\_art()

Depth-first search for identifying articulation points in the airport graph.

Time Complexity: O(V + E) where V is the number of vertices in the graph.

#### **Parameters**

g	The airport graph.	
V	The current vertex in the DFS traversal.	
1	Set to store the identified articulation points.	
i	Reference to the current DFS iteration number.	

#### 5.15.1.2 haversineDistance()

Calculates the Haversine distance between two geographical points.

Time Complexity: O(1)

#### **Parameters**

lat1	Latitude of the first point.
lon1	Longitude of the first point.
lat2	Latitude of the second point.
lon2	Longitude of the second point.

#### Returns

Haversine distance between the two points.

## 5.16 src/main.cpp File Reference

```
#include "../inc/Airline.hpp"
#include "../inc/Airport.hpp"
#include "../inc/Graph.hpp"
#include "../inc/FlightNetwork.hpp"
#include "../inc/App.hpp"
```

#### **Functions**

• int main ()

## 5.16.1 Function Documentation

## 5.16.1.1 main()

```
int main ( )
```

# Index

```
addEdge
                                                              bestFlightMenu, 17
     Graph< T >, 36
                                                              flightnetwork, 18
     Vertex< T>, 42
                                                              globalStatistics, 17
addVertex
                                                              goBackStatisticsMenu, 17
    Graph < T >, 36
                                                              mainMenu, 17
                                                              numberOfDestinations, 17
adj
     Vertex< T>, 47
                                                              reachableDest, 17
Air Travel Flight Management System, 1
                                                              showNumFlights, 17
Airline, 7
                                                              statisticsMenu, 17
    Airline, 8
                                                         App.cpp
    callsign, 10
                                                              clearScreen, 62
    code, 10
                                                         App.hpp
                                                              clearScreen, 51
    country, 10
    getCallsign, 8
                                                         bestFlight
    getCode, 8
                                                              FlightNetwork, 26
    getCountry, 9
                                                         bestFlightMenu
    getName, 9
                                                              App, 17
    name, 10
                                                         bfs
    setCallsign, 9
                                                              Graph < T >, 36
    setCode, 9
                                                         bfsDistance
    setCountry, 10
                                                              Graph < T >, 37
    setName, 10
airlineCodeToName
                                                         callsign
    FlightNetwork, 24
                                                              Airline, 10
Airport, 11
                                                         city
    Airport, 12
                                                              Airport, 15
    city, 15
                                                         cityCriteria
    code, 15
                                                              FlightNetwork, 26
    country, 15
                                                         clearScreen
    getCity, 13
                                                              App.cpp, 62
    getCode, 13
                                                              App.hpp, 51
    getCountry, 13
                                                         code
    getName, 13
                                                              Airline, 10
    getPosition, 13
                                                              Airport, 15
    name, 16
                                                         codeCriteria
    operator<, 13
                                                              FlightNetwork, 27
    operator==, 14
                                                         coordinateCriteria
    position, 16
                                                              FlightNetwork, 27
    setCity, 14
                                                         country
    setCode, 14
                                                              Airline, 10
    setCountry, 14
                                                              Airport, 15
    setName, 15
    setPosition, 15
                                                         dest
airportCodeToName
                                                              Edge< T >, 21
     FlightNetwork, 24
                                                         dfs
airportsGraph
                                                              Graph < T >, 37
    FlightNetwork, 34
                                                         dfs art
App, 16
                                                              FlightNetwork.cpp, 63
     App, 17
                                                              FlightNetwork.hpp, 52
```

66 INDEX

dfs/sit         dfs_art, 52           Graph< T > , 38         haversineDistance, 52           distance         Vertex< T > , 43           Edge         getAdj           Edge < T > , 19         getAiportsGraph           Edge, 19         FlightNetwork, 28           getBe, 19         getCallsign           getUbest, 19         getCallsign           getWeight, 19         getCitiesDestinations           Graph< T > , 21         FlightNetwork, 28           getCity         Airline, 8           getCitiesDestinations         FlightNetwork, 28           getCode         Airline, 8           getCode         Airline, 8           getCode         Airport, 13           getCode         Airine, 8           setWeight, 20         Airport, 13           Vertex< T > , 21         getCode           weight, 21         FlightNetwork, 28           getCountries Code         getCountry           Graph< T > , 38         FlightNetwork, 29           getDest         Graph< T > , 38           FlightNetwork, 21         getDiffCountriesAirport           airportSdraph, 34         getDiffCountriesAirport           getDiffCountriesAirport, 29         getSidobalNumOfAirports         getSid		
distance         Vertex< T > , 47         getAdj         Vertex< T > , 43         getAjportsGraph         FlightNetwork, 28         getAjportsGraph         FlightNetwork, 28         getAiportsDestinations         FlightNetwork, 28         getAiportsDestinations         FlightNetwork, 28         getAiportsDestinations         FlightNetwork, 28         getCallsign         Airline, 8         getCallsign         Airline, 8         getCitiesDestinations         FlightNetwork, 28         getCitiesDestinations         FlightNetwork, 28         getCoty         Airline, 8         Airport, 13         getCode         Airline, 8         Airport, 13         getCode         Airline, 8         Airport, 13         getCode         Airline, 8         Airport, 13         getCountriesDestinations         FlightNetwork, 28         getCountriesDestinations         FlightNetwork, 28         getCountry         Airport, 13         getCountry         Airline, 9         Airport, 13         getDest         Edge< T >, 19         getDiffCountriesAirport         FlightNetwork, 29         getDiffCountriesAirport         FlightNetwork, 29         getDiffCountriesAirport         FlightNetwork, 29         getDiffCountriesAirports         FlightNetwork, 29         getDistance         Vertex < T >, 43         getLow         Yertex < T >, 43         getLow         FlightNetwork, 30         getGlobalNumOfFlights         FlightNetwork, 30         getInDegree         Yertex < T >,	dfsVisit	dfs_art, <mark>52</mark>
Edge	Graph < T >, 38	haversineDistance, 52
Edge CT >, 19 Edge CT >, 18 Edge Sedje T >, 19 Edge Sedje T >, 21 Edge Sedje T >, 38 Edge Sedje T >, 38 Edge Sedje T >, 38 ElightNetwork, 21 Edge Sedje T >, 38 ElightNetwork, 29 Edge Sedje T >, 39 Edge Sedje T >, 43 Edge T >, 43 Edge Sedje T >, 43 Edge T	distance	
Edge T >, 19 Edge T >, 19 Edge T >, 18 Gest, 21 Edge, 19 getDest, 19 getDest, 19 getDest, 19 getIno, 19 getWeight, 19 Graph < T >, 21 info, 21 setDest, 20 setInfo, 20 setWeight, 20 Vertex < T >, 21 weight, 21 EdgesAlDistanceDFS Graph < T >, 38 FlightNetwork, 28 getCountries Destinations FlightNetwork, 28 getCountry Reflection of the state of	Vertex $<$ T $>$ , 47	<del>-</del> -
Edge < T >, 19  Edge < T >, 18  dest, 21  Edge, 19  getDest, 19  getDest, 19  getDest, 19  getWeight, 19  Graph < T >, 21  info, 21  setDest, 20  setInfo, 20  setInfo, 20  setInfo, 20  setNeight, 21  EdgesAlDistanceDFS  Graph < T >, 38  FlightNetwork, 28  getCountry  Graph < T >, 38  FlightNetwork, 29  getCountry  Airport, 13  getDest  Graph < T >, 38  FlightNetwork, 29  getDest  Graph < T >, 38  FlightNetwork, 29  getDest  Graph < T >, 38  FlightNetwork, 29  getDiffCountriesAirport  getDiftCountriesAirport  FlightNetwork, 29  getDiffCountriesAirport  getDistance  Vertex < T >, 43  getDistance  Vertex < T >, 43  getEssentialAirports, 29  getGlobalNumOfAirports Septinations, 28  getCitiesDestinations, 28  getCountriesCity, 29  getEssentialAirports, 29  getEssentialAirports, 29  getGlobalNumOfFlights, 30  getGeatestTraffic, 30  getReachableCountries, 31  listBestFlights, 32  maximumTrip, 32  nameCriteria, 32  numFlightsAirline, 33  numFlightsAirline, 33  numFlightsAirline, 33  haversineDistance, 63  FlightNetwork, 21  getReachableAirports, 31  getReachableAirport, 33  numFlightsAirline, 33  haversineDistance, 63  FlightNetwork, 31  getReachableAirport  getReachableAirport  getPosition  Airjort, 13  getReachableAirport  getInDegree  Vertex < T >, 43  getLow  Vertex < T >, 43  getNum  Vertex < T >, 44  getNumVertex  Graph < T >, 39  getNum  Vertex < T >, 44  getNumVertex  Graph < T >, 39  getReachableAirports, 31  getReachableAirport, 33  numFlightsAirline, 33  haversineDistance, 63		
Edge < T >, 18 dest, 21 Edge, 19 getDest, 19 getInfo, 19 getInfo, 19 getWeight, 19 Graph < T >, 21 info, 21 setDest, 20 setInfo, 20 setWeight, 20 vertex < T >, 21 weight, 21 EdgesAtDistanceDFS Graph < T >, 38 FlightNetwork, 28 getCountry Graph < T >, 38 FlightNetwork, 21 airlineCodeToName, 24 airportGoraph, 28 destFlight, 26 clyCriteria, 26 clyCriteria, 27 FlightNetwork, 23 getAirportsDestinations, 28 getAirportsDestinations, 28 getCitiesDestinations, 28 getCountriesAirport, 29 getBiffCountriesCity, 29 getClobalNumOfAirports solutions getReachableCities, 31 getReachableCities  Airjort, 13 getReachableCities FlightNetwork, 30 getInDegree Vertex < T >, 43 getLow Vertex < T >, 44 getName Airjort, 13 getNam Vertex < T >, 44 getName Airjort, 13 getNam Vertex < T >, 44 getName Airjort, 13 getNam Vertex < T >, 34 getName Airjort, 13 getNam Vertex < T >, 34 getName Airjort, 13 getName Airjort, 29 getDition Airjort, 13 getName Airjort, 13 getName Airjort, 29 getDition Airjort, 13 getName Airjort, 13 getName Vertex < T	Edge	getAiportsGraph
dest, 21 Edge, 19 getDest, 19 getDest, 19 getInfo, 19 getWeight, 19 Graph < T >, 21 info, 21 setDest, 20 setInfo, 20 setInfo, 20 setWeight, 20 Vertex < T >, 21 weight, 21 EdgesAlDistanceDFS Graph < T >, 38 FlightNetwork, 28 getCountry Airline, 8 Airport, 13 getCountriesDestinations FlightNetwork, 28 getCountry Airline, 8 Airport, 13 getCountriesDestinations FlightNetwork, 28 getCountry Airline, 9 Airport, 13 getDest EdgesAlDistanceDFS Graph < T >, 38 FlightNetwork, 21 airlineCodeToName, 24 airportCodeToName, 24 airportCodeToName, 24 airportGodeToName, 24 airportGodeToName, 24 airportGodeToName, 24 airportGodeToName, 24 airportGodeToName, 24 getDiffCountriesAirport FlightNetwork, 29 getDistance Vertex < T >, 43 getCountriesCity FlightNetwork, 29 getCountriesDestinations, 28 getAiportsDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getCountriesAirport, 29 getBiffCountriesCity, 29 getBiffCountriesCity, 29 getBiffCountriesCity, 29 getClitesDestinations, 28 getCountriesCity, 29 getGlobalNumOfFlights FlightNetwork, 30 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 qetReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirine, 33 numFlightsAirine, 33 numFlightsAirine, 33 numFlightsAirine, 33 numFlightsAirport, 31 getReachableAirports App, 18 FlightNetwork App, 18 FlightNetwork App, 18 FlightNetwork, 31 getReachableCities FlightNetwork, 31 getReachableCities	Edge < T >, 19	FlightNetwork, 28
dest, 21 Edge, 19 getDest, 19 getDest, 19 getInfo, 19 getWeight, 19 Graph < T >, 21 setDest, 20 setInfo, 20 setWeight, 20 Vertex < T >, 21 weight, 21 EdgesAtDistanceDFS Graph < T >, 38 FlightNetwork, 21 airlineCodeToName, 24 airportCodeToName, 24 airportCodeToName, 24 airportCodeToName, 24 airportGodeToName, 24 airportScraph, 34 bestFlight, 26 codeCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAjportsDestinations, 28 getAjportsDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 29 getDiffCountriesAirport getDiffCountriesCity, 29 getDiffCountriesCity, 29 getDiffCountriesCity, 29 getDiffCountriesCity, 29 getDiffCountriesCity, 29 getDiffCountriesCity, 29 getGlobalNumOfAirports, 30 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirport, 33 flightnetwork App, 18 FlightNetwork, 29 getReachableCities of the control of th	Edge < T >, 18	getAirportsDestinations
Edge, 19 getDest, 19 getIDest, 19 getWeight, 19 getWeight, 19 Graph< T >, 21 setDest, 20 setInfo, 20 setWeight, 20 Vertex< T >, 21 weight, 21 EdgesAtDistanceDFS Graph< T >, 38 FlightNetwork, 28 getCountriesDestinations FlightNetwork, 28 getCountriesDestinations FlightNetwork, 28 getCountriesDestinations FlightNetwork, 28 getDost Graph< T >, 38 FlightNetwork, 21 airlineCodeToName, 24 airportGodeToName, 24 getDiffCountriesAirport FlightNetwork, 29 getDiffCountriesCity FlightNetwork, 29 getDiffCountriesCity FlightNetwork, 29 getSesentialAirports FlightNetwork, 30 getGlobalNumOfAirports, 30 getGlobalNumOfAirports, 30 getGlobalNumOfAirports, 30 getReachableCities, 31 getReachableCities, 31 getReachableCities, 31 getReachableCountries, 31 listBestFlights, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 35 numFlightsAi	_	FlightNetwork, 28
getDest, 19 getHoft, 19 getWeight, 19 Graph < T >, 21 info, 21 setDest, 20 setUneight, 20 Vertex < T >, 21 getCode Airline, 8 Airport, 13 getCountriesDestinations FlightNetwork, 28 getCountry Airline, 9 Airport, 13 getDest Edges T >, 19 getDiffCountriesAirport FlightNetwork, 29 getDiffCountriesCity FlightNetwork, 29 getGlobalNumOfAirports FlightNetwork, 30 getGlobalNumOfAirports FlightNetwork, 30 getGlobalNumOfFlights GetDestinations, 28 getCountriesAirport, 29 getEssentialAirports, 29 getEssentialAirports, 29 getEssentialAirports, 29 getEssentialAirports, 29 getEssentialAirports, 29 getEssentialAirports, 31 getReachableCities, 31 getReachableCity, 34 flightnetwork App, 18 FlightNetwork, 39 getReachableAirports Graph < T >, 44 getNum Vertex < T >, 44 getNum Vertex < T >, 44 getNumVertex Graph < T >, 39 getPosition Airport, 13 getReachableAirports Graph < T >, 39 getPosition Airport, 13 getReachableAirports Graph < T >, 39 getReachableAirports Graph < T >, 49 getPosition Airport, 13 getReachableAirports Graph < T >, 49 getNum Vertex < T >, 44 getNum Vertex < T >, 44 getNum Vertex < T >, 43 getReachableAirports Graph < T >, 44 getNum Vertex < T >, 43 getReachableAirports Graph < T >, 44 getNum Vertex < T >, 43 getReachableAirports Graph < T >, 44 getNum Vertex < T >, 44 getNum Vertex	Edge, 19	getCallsign
getInfo, 19 getWeight, 19 Graph < T >, 21 info, 21 setDest, 20 setInfo, 20 setWeight, 20 Vertex < T >, 21 getCountriesDestinations FlightNetwork, 28 getCountry Graph < T >, 38 FlightNetwork, 28 getCountry Graph < T >, 38 FlightNetwork, 21 getCountry Graph < T >, 38 FlightNetwork, 21 getDest Graph < T >, 38 FlightNetwork, 21 getDest Graph < T >, 38 FlightNetwork, 21 getDiffCountriesAirport griprica, 26 codeCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getCountry getDiffCountriesCity FlightNetwork, 29 getSistance Vertex < T >, 43 getEssentialAirports FlightNetwork, 29 getGlobalNumOfFlights getCitesDestinations, 28 getCitesDestinations, 28 getCitesDestinations, 28 getCountriesCity, 29 getGlobalNumOfFlights getCountriesCity, 29 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getReachableCountries, 31 getReachableCities, 31 getReachableCities, 31 getReachableCity, 34 flightNetwork, 29 getSitinn getReachableCity, 34 flightNetwork, 29 getSitinn Airport, 13 getReachableCity, 34 flightNetwork, 30 getReachableCity, 34 flightNetwork, 29 getSition Airport, 13 getReachableCity, 34 flightNetwork, 30 getReachableCities App, 18 FlightNetwork, 29 getReachableCities flightNetwork, 30 getReachableCities flightNetwork, 30 getReachableCities flightNetwork, 30 getReachableCity, 34 flightnetwork App, 18 FlightNetwork, 39 getReachableCities		Airline, 8
getWeight, 19 Graph < T > , 21 info, 21 setDest, 20 setInto, 20 setWeight, 20 Vertex < T > , 21 weight, 21 EdgesAtDistanceDFS Graph < T > , 38 FlightNetwork, 21 airport, 13 getCountriesDestinations FlightNetwork, 28 getCountry Graph < T > , 38 FlightNetwork, 21 airlineCodeToName, 24 airportCodeToName, 24 airportCodeToName, 24 airportGraph, 34 bestFlight, 26 cityCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAiportsGraph, 28 getAiportsGraph, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getCountriesAirport, 29 getEssentialAirports, 29 getEssentialAirports, 29 getGlobalNumOfAirports, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getReachableCities, 31 getReachableCountries, 31 listBestFlights, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsAirport, 34 flightNetwork, 29 getPosition Airport, 13 getReachableAirports, 31 getReachableCity, 34 flightnetwork App, 18 FlightNetwork, 29 getReachableAirports getPosition Airport, 13 getReachableAirports getPosition Airport, 13 getReachableCities getPosition Airport, 13 getReachableAirports getPosition Airport, 13 getReachableAirports getPosition Airport, 13 getReachableAirports getPosition Airport, 13 getReachableAirports getPosition Airport, 31 getReachableAirports getPosition Airport, 31 getReachableAirports getPosition Airport, 31 getReachableAirports getPosition Airport, 13 getReachableAirports getPosition Airport, 13 getReachableAirports getPosition Airport, 13 getReachableAirports getPosition Airport, 13 getReachableAirports getReachableAirports getPosition Airport, 13 getReachableAirports getReachableCities	_	getCitiesDestinations
Graph < T >, 21 into, 21 setDest, 20 setInfo, 20 setWeight, 20 Vertex < T >, 21 weight, 21 EdgesAtDistanceDFS Graph < T >, 38 FlightNetwork, 21 airlineCodeToName, 24 airportCodeToName, 24 getDiffCountriesCity FlightNetwork, 29 getDistance cityCriteria, 26 codeCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAirportsDestinations, 28 getAirportsDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getDiffCountriesAirport, 29 getEssentialAirports, 29 getGlobalNumOfAirports, 30 getGreatestTraffic, 30 getReachableCountries, 31 listBestFlights, 32 numFlightsAirine, 33 numFlightsAirine, 33 numFlightsAirine, 33 numFlightsAirine, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork. 39 getReachableCities	_	_
info, 21 setDest, 20 setInfo, 20 setWeight, 20 Vertex < T >, 21 weight, 21 EdgesAtDistanceDFS Graph < T >, 38 FlightNetwork, 23 airjortCodeToName, 24 airportGodeToName, 24 getDiffCountriesCity FlightNetwork, 29 getDiffCountriesCity FlightNetwork, 29 getDistance Vertex < T >, 43 getEssentialAirports FlightNetwork, 29 getBissentialAirports getAirportsDestinations, 28 getCitiesDestinations, 28 getClitiesDestinations, 28 getClitiesDestinations, 28 getGlobalNumOfAirports, 29 getEssentialAirports, 29 getEssentialAirports, 29 getEssentialAirports, 29 getGlobalNumOfFlights getGlobalNumOfFlights getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getReachableCities, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsAirport, 33 numFlightsAirline, 33 numFlightsAirport, 33 numFlightsAirport, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63		
setDest, 20 setInfo, 20 setWeight, 20 Vertex< T >, 21 weight, 21 EdgesAtDistanceDFS Graph< T >, 38 FlightNetwork, 21 airportCodeToName, 24 airportCodeToName, 24 airportCodeToName, 24 airportCodeToName, 24 airportCodeToName, 24 airportSgraph, 34 bestFlight, 26 cityCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAiportsGraph, 28 getAiportsGraph, 28 getAiportsGraph, 28 getAiportsGraph, 29 getBiffCountriesAirports FlightNetwork, 29 getBiffCountriesCity, 29 getGl	•	
setInfo, 20 setWeight, 20 Vertex < T >, 21 weight, 21 EdgesAtDistanceDFS Graph < T >, 38 FlightNetwork, 28 Graph < T >, 38 FlightNetwork, 21 getCountry Airline, 9 Airport, 13 getCountry Airline, 9 Airport, 13 getCountry Airline, 9 Airport, 13 getDest Edge < T >, 19 getDiffCountriesAirport FlightNetwork, 21 airlineCodeToName, 24 airportCodeToName, 24 getDiffCountriesCity FlightNetwork, 29 getDistance Vertex < T >, 43 getEssentialAirports FlightNetwork, 29 getDistance Vertex < T >, 43 getEssentialAirports FlightNetwork, 29 getGlobalNumOfAirports getAirportsDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getCountriesCity, 29 getGlobalNumOfFiights FlightNetwork, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getReachableAirports, 31 getReachableCountries, 31 iistBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsAirport, 33 numFlightsAirline, 35 numFlightsAirline, 36 numFlightsAirline, 37 numFlightsAirline, 39 numFlightsAirline, 39 numFlightsAirline, 39 numFlights		·
setWeight, 20 Vertex < T >, 21 weight, 21 EdgesAtDistanceDFS Graph < T >, 38 FlightNetwork, 28 getCountry Airline, 9 Airport, 13 getDest Edge < T >, 19 getDiffCountriesAirport FlightNetwork, 29 getDiffCountriesCity FlightNetwork, 29 getGlobalNumOfAirports FlightNetwork, 30 getGlobalNumOfAirports FlightNetwork, 30 getGlobalNumOfAirports, 30 getGlobalNumOfAirports, 30 getGlobalNumOfAirports, 30 getGlobalNumOfAirports, 31 getReachableAirports, 31 getReachableAirports, 31 getReachableAirports, 31 getReachableAirports IslightSetFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsCity, 34 flightNetwork, 29 getReachableAirports App, 18 FlightNetwork, 29 getReachableCities Graph < T >, 43 getReachableAirports Graph < T >, 43 getNum Airline, 9 Airport, 13 getNum Vertex < T >, 43 getNum Vertex < T >, 44 getNume Airline, 9 Airport, 13 getNumVertex Graph < T >, 39 getPosition Airport, 13 getReachableAirports App, 18 FlightNetwork, 31 getReachableCities FlightNetwork, 31 getReachableCities FlightNetwork, 31 getReachableCities		<del>-</del>
Vertex < T >, 21 weight, 21 EdgesAtDistanceDFS Graph < T >, 38 findVertex Graph < T >, 38 findVertex Graph < T >, 38 FlightNetwork, 21 airlineCodeToName, 24 airportCodeToName, 24 airportGaraph, 34 bestFlight, 26 cityCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAirportsDestinations, 28 getAirportsDestinations, 28 getCountry getDistCountriesAirport FlightNetwork, 29 getDistance Vertex < T >, 43 getEssentialAirports FlightNetwork, 29 getGlobalNumOfAirports FlightNetwork, 30 getGreatestTraffic getTiesDestinations, 28 getDiffCountriesAirport, 29 getEssentialAirports, 30 getGreatestTraffic getTountriesAirport, 30 getGreatestTraffic getReachableAirports, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsCountries App, 18 FlightNetwork, 21 getCountriesDestinations FlightNetwork, 28 getCountry Airline, 9 getDiftCountriesAirport FlightNetwork, 29 getBitfCountriesCity FlightNetwork, 30 getGlobalNumOfFlights FlightNetwork, 30 getInDegree Vertex < T >, 43 getLow Vertex < T >, 43 getLow Vertex < T >, 44 getName Airline, 9 Airport, 13 getNum Vertex < T >, 44 getName Airline, 9 Airport, 13 getNum Vertex < T >, 44 getNumVertex Graph < T >, 39 getPosition Airport, 13 getNum Vertex < T >, 44 getNumVertex Graph < T >, 39 getPosition Airport, 13 getReachableAirports FlightNetwork, 31 getReachableCitites FlightNetwork, 31 getReachableCitites		
weight, 21 EdgesAtDistanceDFS Graph < T > , 38 Graph < T > , 38 FlightNetwork, 21 getDest Graph < T > , 38 FlightNetwork, 21 grilineCodeToName, 24 airportCodeToName, 24 airportGaraph, 34 bestFlight, 26 cityCriteria, 26 codeCriteria, 27 FlightNetwork, 23 getAiportsGraph, 28 getAiportsGraph, 28 getAiportsGraph, 28 getAiportsDestinations, 28 getCountriesDestinations, 28 getDiffCountriesCity, 29 getBostance Vertex < T > , 43 getBesentialAirports FlightNetwork, 29 getGlobalNumOfAirports FlightNetwork, 30 getGreatestTraffic FlightNetwork, 30 getGreatestTraffic getDiffCountriesCity, 29 getBesentialAirports, 30 getGreatestTraffic getBostNumOfFlights, 30 getGreatestTraffic, 30 getBost Vertex < T > , 43 getInDegree Vertex < T > , 43 getInDegree Vertex < T > , 43 getLow Vertex < T > , 43 getLow Vertex < T > , 44 getName Airline, 9 Airport, 13 getLow Vertex < T > , 44 getName Airline, 9 Airport, 13 getNum Vertex < T > , 44 getName Airline, 9 Airport, 13 getNum Vertex < T > , 44 getName Airline, 9 Airport, 13 getNum Vertex < T > , 44 getName Airline, 9 Airport, 13 getNum Vertex < T > , 44 getName Airline, 9 Airport, 13 getNum Vertex < T > , 44 getName Airline, 9 Airport, 13 getNum Vertex < T > , 44 getName Airline, 9 Airport, 13 getNum Vertex < T > , 44 getName Airline, 9 Airport, 13 getNum Vertex < T > , 44 getName Airline, 9 Airport, 13 getNum Vertex < T > , 44 getName Airline, 9 Airport, 13 getNum Vertex < T > , 44 getName Airline, 9 Airport, 13 getNumVertex Graph < T > , 39 getPosition Airport, 13 getReachableAirports FlightNetwork, 31 getReachableCities FlightNetwork, 31 getReachableCities		•
EdgesAtDistanceDFS Graph< T >, 38  findVertex Graph< T >, 38  FlightNetwork, 21 airlineCodeToName, 24 airportCodeToName, 24 airportSGraph, 34 bestFlight, 26 cityCriteria, 26 codeCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAiportsDestinations, 28 getAiprortsDestinations, 28 getCountriesDestinations, 28 getCountriesCity, 29 getGlobalNumOfAirports getDiffCountriesCity, 29 getGlobalNumOfAirports FlightNetwork, 29 getGlobalNumOfAirports FlightNetwork, 30 getGlobalNumOfFlights FlightNetwork, 30 getGlobalNumOfFlights GetGlobalNumOfAirports FlightNetwork, 30 getGlobalNumOfFlights FlightNetwork, 30 getGlobalNumOfFlights FlightNetwork, 30 getGlobalNumOfAirports GetGlobalNumOfFlights FlightNetwork, 30 getGlobalNumOfFlights FlightNetwork, 30 getGlobalNumOfFlights FlightNetwork, 30 getGlobalNumOfFlights FlightNetwork, 30 getGlobalNumOfAirports GetGlobalNumOfFlights FlightNetwork, 30 getInDegree Vertex< T >, 43 getInDegree Vertex< T >, 44 getNumVertex Graph< T >, 39 getNumVertex Graph< T >, 39 getNumVertex Graph< T >, 39 getPosition Airport, 13 getReachableAirports Graph< T >, 39 getReachableAirports Graph< T >, 39 getReachableAirports Graph< T >, 39 getReachableAirports FlightNetwork, 31 getReachableCities		_
Graph< T >, 38  findVertex  Graph< T >, 38  FlightNetwork, 21  airlineCodeToName, 24  airportCodeToName, 24  airportCodeToName, 24  airportCodeToName, 24  airportCodeToName, 24  airportGoraph, 34  bestFlight, 26  cityCriteria, 26  codeCriteria, 27  coordinateCriteria, 27  FlightNetwork, 29  getSistance  Vertex< T >, 43  getEssentialAirports  FlightNetwork, 29  getGlobalNumOfAirports  getAiportsDestinations, 28  getAirportsDestinations, 28  getCountriesDestinations, 28  getDiffCountriesAirport, 29  getGlobalNumOfFlights  getCitiesDestinations, 28  getAirportsDestinations, 28  getCountriesDestinations, 28  getDiffCountriesAirport, 29  getEssentialAirports  getGlobalNumOfFlights  FlightNetwork, 30  getGlobalNumOfFlights  FlightNetwork, 30  getGreatestTraffic  FlightNetwork, 30  getInDegree  Vertex< T >, 43  getInfo  getGlobalNumOfFlights, 30  getGreatestTraffic, 30  getReachableAirports, 31  getReachableCities, 31  getReachableCities, 31  getReachableCities, 31  getReachableCities, 31  getNum  Airline, 9  Airport, 13  getNum  Vertex< T >, 44  getName  Airline, 9  Airport, 13  getNum  Vertex< T >, 44  getNum  Vertex< T >, 44  getNum  Airline, 9  Airport, 13  getNum  Vertex< T >, 44  getNum  Vertex< T >, 44  getNum  Vertex< T >, 44  getNum  Airline, 9  Airport, 13  getNum  Vertex< T >, 44  getNumVertex  Graph< T >, 39  getPosition  Airport, 13  getPosition  Airport, 13  getReachableAirports  FlightNetwork, 31  getReachableAirports  FlightNetwork, 31  getReachableCities		
findVertex Graph < T > , 38 FlightNetwork, 21 airlineCodeToName, 24 airportCodeToName, 24 airportSdraph, 34 bestFlight, 26 cityCriteria, 26 codeCriteria, 27 FlightNetwork, 23 getAiportSGraph, 28 getAiportSGraph, 29 getGitiesDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getDiffCountriesAirports FlightNetwork, 29 getGlobalNumOfAirports FlightNetwork, 30 getGreatestTraffic getDiffCountriesAirport, 29 getGlobalNumOfAirports, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getReachableAirports, 31 getReachableCities, 31 getReachableCities, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsCity, 34 flightNetwork cpp dfs_art, 63 haversineDistance, 63	_	- ·
findVertex Graph< T>, 38 FlightNetwork, 21 airlineCodeToName, 24 airportCodeToName, 24 airportSgraph, 34 bestFlight, 26 cityCriteria, 27 FlightNetwork, 23 getAiportsGraph, 28 getAiprotsDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getDiffCountriesAirport getBightNetwork, 30 getGlobalNumOfFlights getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 31 getReachableCities, 31 getReachableCountries, 31 listBestFlights, 32 numFlightsAirport, 33 numFlightsAirport, 33 numFlightsAirport, 34 flightNetwork, 29 getReachableAirports getName App, 18 FlightNetwork, 29 getReachableAirports getReachableAirports getName Airport, 13 getReachableAirports getReachableAirports getName Airport, 13 getReachableAirports getReachableAirports getReachableAirport, 33 numFlightsAirport, 33 numFlightsAirport, 33 numFlightsAirport, 34 flightNetwork cpp dfs_art, 63 haversineDistance, 63	Graph $<$ T $>$ , 38	
Graph< T >, 38  FlightNetwork, 21  airlineCodeToName, 24  airportCodeToName, 24  airportSGraph, 34  bestFlight, 26  cityCriteria, 26  codeCriteria, 27  FlightNetwork, 23  getAirportsDestinations, 28  getCountriesDestinations, 28  getDiffCountriesCity, 29  getDistance  vertex< T >, 43  getGlobalNumOfAirports  FlightNetwork, 30  getGlobalNumOfAirports, 29  getBiffCountriesCity  flightNetwork, 29  getGlobalNumOfFlights  getCitiesDestinations, 28  getCountriesDestinations, 28  getDiffCountriesCity, 29  getBlobalNumOfFlights, 30  getGlobalNumOfFlights, 30  getGlobalNumOfFlights, 31  getReachableAirports, 31  getReachableCities, 31  getName  Airline, 9  Airport, 13  getNum  Vertex< T >, 44  getNumVertex  Graph< T >, 39  getNumVertex  Graph< T >, 39  getRoachableAirports  App, 18  FlightNetwork copp  dfs_art, 63  haversineDistance, 63		Airport, 13
FlightNetwork, 21 airlineCodeToName, 24 airportCodeToName, 24 airportCodeToName, 24 airportCodeToName, 24 airportCodeToName, 24 airportCodeToName, 24 airportSGraph, 34 bestFlight, 26 cityCriteria, 26 codeCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAirportSGraph, 28 getAirportSDestinations, 28 getAirportSDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getGlobalNumOfFlights getDiffCountriesCity, 29 getEssentialAirports, 30 getGreatestTraffic FlightNetwork, 30 getGreatestTraffic, 30 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getReachableAirports, 31 getReachableCities, 31 getReachableCountries, 31 listBestFlights, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsCity, 34 FlightNetwork, 31 getReachableAirport, 33 numFlightsCity, 34 FlightNetwork, 31 getReachableAirports getPosition App, 18 FlightNetwork, 31 getReachableAirports flightNetwork, 29 getDiffCountriesAirports flightNetwork, 29 getBistance Vertex < T >, 43 getNum Vertex < T >, 44 getNum Vertex < T >, 43 getNum Vertex < T >, 44 getNum Vertex < T >, 43 getNum Vertex < T >, 44 getNumVertex Graph < T >, 39 getPosition Airport, 13 getReachableAirports FlightNetwork, 31 getReachableAirports FlightNetwork, 31 getReachableCities		getDest
airlineCodeToName, 24 airportCodeToName, 24 airportSGraph, 34 bestFlight, 26 cityCriteria, 26 codeCriteria, 27 FlightNetwork, 29 getDistance cityCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAiportsGraph, 28 getAiportsGraph, 28 getAirportsDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getDiffCountriesAirport, 29 getEssentialAirports, 30 getGlobalNumOfFlights getCitiesDestinations, 28 getObiffCountriesCity, 29 getBiffCountriesAirport, 29 getEssentialAirports, 30 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getReachableAirports, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirport, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork, 29 getBlobalCities getDiffCountriesCity getReachableAirports getReachableAirports getReachableAirports getReachableCountries and FlightNetwork, 29 getBlobalNumOfAirports getReachableCountries getNum Vertex < T >, 44 getNum Vertex < T >, 43 getReachableAirport, 13 getReachableAirport, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63	•	Edge< T >, 19
airportCodeToName, 24 airportsGraph, 34 bestFlight, 26 cityCriteria, 26 cityCriteria, 27 coordinateCriteria, 27 FlightNetwork, 29 getEssentialAirports getAirportsGraph, 28 getAirportsDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getDiffCountriesCity, 29 getGlobalNumOfAirports getBiffCountriesAirport, 29 getGlobalNumOfFlights getCitiesDestinations, 28 getOuntriesDestinations, 28 getGlobalNumOfFlights getCitiesDestinations, 28 getGlobalNumOfFlights getCitiesDestinations, 28 getGlobalNumOfFlights getGlobalNumOfFlights, 30 getGlobalNumOfAirports, 30 getInDegree yetEssentialAirports, 30 getInDegree yetEssentialAirports, 30 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getReachableAirports, 31 getReachableAirports, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirport, 33 numFlightsAirport, 33 numFlightsAirport, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63	FlightNetwork, 21	getDiffCountriesAirport
airportsGraph, 34 bestFlight, 26 cityCriteria, 26 cityCriteria, 27 coordinateCriteria, 27 FlightNetwork, 29 getEssentialAirports FlightNetwork, 29 getGlobalNumOfAirports getAirportsDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getGlobalNumOfFlights getDiffCountriesAirport, 29 getBssentialAirports FlightNetwork, 29 getGlobalNumOfFlights getDiffCountriesAirport, 29 getBssentialAirports, 30 getGreatestTraffic FlightNetwork, 30 getInDegree Vertex< T >, 43 getInfo getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getGreatestTraffic, 31 getReachableAirports, 31 getReachableCountries, 31 getReachableCountries, 31 getReachableCountries, 31 getReachableCountries, 31 getReachableCountries, 32 nameCriteria, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork, 29 getDostance Vertex< T >, 43 getNum Vertex< T >, 44 getNumVertex Graph< T >, 39 getNum Airport, 13 getNumVertex Graph< T >, 39 getPosition Airport, 13 getReachableAirports FlightNetwork, 31 getReachableAirports FlightNetwork, 31 getReachableAirports FlightNetwork, 31 getReachableAirports FlightNetwork, 31 getReachableCities	airlineCodeToName, 24	FlightNetwork, 29
bestFlight, 26 cityCriteria, 26 codeCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAiportsGraph, 28 getAirportsDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getDiffCountriesAirport, 29 getGlobalNumOfFlights getBiffCountriesCity, 29 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 31 getReachableAirports, 31 getReachableCities, 31 getReachableCountries, 31 getReachableCountries, 31 getReachableCountries, 31 getReachableCountries, 31 getReachableCountries, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork, cpp dfs_art, 63 haversineDistance, 63	airportCodeToName, 24	getDiffCountriesCity
cityCriteria, 26 codeCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAiportsGraph, 28 getAirportsDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getDiffCountriesAirport, 29 getBesentialAirports getBesentialAirports getBiffCountriesCity, 29 getBesentialAirports, 29 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 31 getReachableAirports, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork, 27 getReachableCities FlightNetwork, 30 getGlobalNumOfFlights, 30 getInfo Edge < T >, 19 Vertex < T >, 43 getLow Vertex < T >, 44 getName Airline, 9 Airport, 13 getNum Vertex < T >, 44 getNum Vertex < T >, 44 getNumVertex Graph < T >, 39 getReachableAirports, 31 getNumVertex Graph < T >, 39 getPosition Airport, 13 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63	airportsGraph, 34	FlightNetwork, 29
cityCriteria, 26 codeCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAiportsGraph, 28 getAirportsDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getDiffCountriesAirport, 29 getBesentialAirports getBesentialAirports getBiffCountriesCity, 29 getBesentialAirports, 29 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 31 getReachableAirports, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork, 27 getReachableCities FlightNetwork, 30 getGlobalNumOfFlights, 30 getInfo Edge < T >, 19 Vertex < T >, 43 getLow Vertex < T >, 44 getName Airline, 9 Airport, 13 getNum Vertex < T >, 44 getNum Vertex < T >, 44 getNumVertex Graph < T >, 39 getReachableAirports, 31 getNumVertex Graph < T >, 39 getPosition Airport, 13 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63	bestFlight, 26	_
codeCriteria, 27 coordinateCriteria, 27 FlightNetwork, 23 getAiportsGraph, 28 getAirportsDestinations, 28 getCitiesDestinations, 28 getCitiesDestinations, 28 getDiffCountriesAirport, 29 getEssentialAirports getBiptNetwork, 30 getGlobalNumOfFlights FlightNetwork, 30 getGreatestTraffic FlightNetwork, 30 getBiptNetwork, 30 getGreatestTraffic FlightNetwork, 30 getInDegree yetEssentialAirports, 29 getBiptOcuntriesCity, 29 getGlobalNumOfAirports, 30 getGlobalNumOfFlights, 30 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getReachableAirports, 31 getReachableCountries, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirport, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63		<del>-</del>
coordinateCriteria, 27 FlightNetwork, 23 getAiportsGraph, 28 getAiportsDestinations, 28 getAirportsDestinations, 28 getCountriesDestinations, 28 getCountriesAirport, 29 getBlobalNumOfFlights FlightNetwork, 30 getGlobalNumOfFlights FlightNetwork, 30 getGreatestTraffic FlightNetwork, 30 getInDegree Vertex< T >, 43 getInfo getGlobalNumOfAirports, 30 getInfo getGlobalNumOfFlights, 30 getLow getReachableAirports, 31 getLow getReachableCities, 31 getLow getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63	·	
FlightNetwork, 23 getAiportsGraph, 28 getAiportsDestinations, 28 getAirportsDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesAirport, 29 getBiffCountriesCity, 29 getBiffCountriesCity, 29 getGlobalNumOfAirports, 30 getGreatestTraffic FlightNetwork, 30 getInDegree Vertex< T >, 43 getInfo getGlobalNumOfAirports, 30 getInfo getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getGlobalNumOfFlights, 30 getInfo getGlobalNumOfFlights, 31 getLow getReachableAirports, 31 getLow getReachableCities, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsAirline, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63		-
getAiportsGraph, 28 getAirportsDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getGreatestTraffic getDiffCountriesAirport, 29 getBssentialAirports, 29 getEssentialAirports, 29 getGlobalNumOfAirports, 30 getGlobalNumOfAirports, 30 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getGreatestTraffic, 30 getReachableAirports, 31 getLow getReachableCities, 31 getReachableCountries, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirport, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63  FlightNetwork, 30 getGreatestTraffic FlightNetwork, 30 getInfo Edge< T >, 43 getLow Vertex< T >, 44 getLow Vertex< T >, 44 getName Airline, 9 Airport, 13 getNum Vertex< T >, 44 getNumVertex Graph< T >, 39 getNumVertex Graph< T >, 39 getNumVertex Graph< T >, 39 getReachableAirports FlightNetwork, 31 getReachableAirports FlightNetwork, 31 getReachableCities		_
getAirportsDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getCountriesDestinations, 28 getGountriesCity, 29 getDiffCountriesCity, 29 getEssentialAirports, 29 getGlobalNumOfAirports, 30 getInDegree Vertex< T >, 43 getInDegree Vertex< T >, 43 getInfo getGlobalNumOfAirports, 30 getInDegree Vertex< T >, 43 getInfo getGlobalNumOfAirports, 30 getInDegree Vertex< T >, 43 getInfo g		-
getCitiesDestinations, 28 getCountriesDestinations, 28 getGountriesDestinations, 28 getGountriesDestinations, 28 getDiffCountriesCity, 29 getBiffCountriesCity, 29 getEssentialAirports, 29 getBobalNumOfAirports, 30 getGlobalNumOfAirports, 30 getGlobalNumOfFlights, 30 getGreatestTraffic getBiffCountriesCity, 29 getInDegree Vertex< T >, 43 getInfo getGlobalNumOfFlights, 30 getInfo getGlobalNumOfFlights, 30 getReachableAirports, 31 getLow Vertex< T >, 43 getReachableAirports, 31 getLow getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirport, 33 numFlightsAirport, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63		_
getCountriesDestinations, 28 getDiffCountriesAirport, 29 getDiffCountriesCity, 29 getEssentialAirports, 29 getEssentialAirports, 29 getGlobalNumOfAirports, 30 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getReachableAirports, 31 getReachableCities, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsCity, 34 flightNetwork.cpp dfs_art, 63 haversineDistance, 63 getInfo getInfo Edge< T >, 43 getInfo Edge< T >, 19 Vertex< T >, 43 getLow Vertex< T >, 44 getName Airline, 9 Airport, 13 getNum Vertex< T >, 44 getNum Vertex< T >, 44 getNumVertex Graph< T >, 39 getNumVertex Graph< T >, 39 getReachableAirports GetReachableAirports GetReachableAirports GetReachableAirports GetReachableAirports GetReachableAirports GetReachableCities		-
getDiffCountriesAirport, 29 getDiffCountriesCity, 29 getEssentialAirports, 29 getGlobalNumOfAirports, 30 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getReachableAirports, 31 getReachableCities, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsCity, 34 flightNetwork.cpp dfs_art, 63 haversineDistance, 63  FlightNetwork, 30 getInDegree Vertex< T >, 43 getLow Vertex< T >, 44 getName Airline, 9 Airport, 13 getNum Vertex< T >, 44 getNumVertex Graph< T >, 39 getNumVertex Graph< T >, 39 getPosition Airport, 13 getReachableAirports FlightNetwork, 31 getReachableCities		•
getDiffCountriesCity, 29 getEssentialAirports, 29 getGlobalNumOfAirports, 30 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getReachableAirports, 31 getReachableCities, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsCity, 34 flightNetwork App, 18 FlightNetwork.cpp dfs_art, 63 flightNetwork, 31 getBox Vertex T >, 44 getReachableCountries, 31 getNum Vertex T >, 44 getName Airline, 9 Airport, 13 getNum Vertex T >, 44 getNumVertex Graph < T >, 39 getNumVertex Graph < T >, 39 getPosition Airport, 13 getReachableAirports flightNetwork.cpp getReachableAirports flightNetwork, 31 getReachableCities	<del>-</del>	_
getEssentialAirports, 29 getGlobalNumOfAirports, 30 getInfo getGlobalNumOfFlights, 30 getReactstTraffic, 30 getReachableAirports, 31 getReachableCities, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 numFlightsAirline, 33 numFlightsAirport, 33 flightnetwork App, 18 FlightNetwork.cpp dragetReachableCities, 31 getIlnfo getIlnfo getIlnfo getLow Vertex< T >, 44 getName Airline, 9 Airline, 9 Airport, 13 getNum Vertex< T >, 44 getNumVertex Graph< T >, 39 getNumVertex Graph< T >, 39 getPosition Airport, 13 getReachableAirports flightNetwork.cpp dfs_art, 63 haversineDistance, 63	- · · · · · · · · · · · · · · · · · · ·	_
getGlobalNumOfAirports, 30 getGlobalNumOfFlights, 30 getGreatestTraffic, 30 getReachableAirports, 31 getReachableCities, 31 getReachableCountries, 31 listBestFlights, 32 maximumTrip, 32 numFlightsAirline, 33 numFlightsAirport, 33 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63  getInfo Edge < T >, 19 yertex < T >, 43 getLow Vertex < T >, 44 getName Airline, 9 Airport, 13 getNum Vertex < T >, 44 getNum Vertex < T >, 44 getNumVertex Graph < T >, 39 getNumVertex Graph < T >, 39 getPosition Airport, 13 getReachableAirports FlightNetwork, 31 getReachableCities	- · · · · · · · · · · · · · · · · · · ·	-
$\begin{array}{llllllllllllllllllllllllllllllllllll$	-	
$\begin{array}{llllllllllllllllllllllllllllllllllll$		· ·
$\begin{array}{llllllllllllllllllllllllllllllllllll$	<b>3</b> ,	
getReachableCities, 31  getReachableCountries, 31  listBestFlights, 32  maximumTrip, 32  nameCriteria, 32  numFlightsAirline, 33  numFlightsAirport, 33  numFlightsCity, 34  flightnetwork  App, 18  FlightNetwork.cpp  dfs_art, 63  haversineDistance, 63  FlightNetwork, 31  getRachableCities  Vertex < T >, 44  getNum  Vertex < T >, 44  getNumVertex  Graph < T >, 39  getPosition  Airport, 13  getReachableAirports  FlightNetwork, 31  getReachableCities	<del>-</del>	· ·
getReachableCountries, 31  listBestFlights, 32  maximumTrip, 32  nameCriteria, 32  numFlightsAirline, 33  numFlightsAirport, 33  numFlightsCity, 34  flightnetwork  App, 18  FlightNetwork.cpp  dfs_art, 63  haversineDistance, 63  getName  Airline, 9  Airport, 13  getNum  Vertex < T >, 44  getNumVertex  Graph < T >, 39  getPosition  Airport, 13  getReachableAirports  FlightNetwork, 31  getReachableCities	•	getLow
listBestFlights, 32 maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirport, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63  Airline, 9 Airport, 13 getNum Vertex   Vertex < T >, 44 getNumVertex Graph < T >, 39 getPosition Airport, 13 getReachableAirports FlightNetwork, 31 getReachableCities	getReachableCities, 31	Vertex $<$ T $>$ , 44
maximumTrip, 32 nameCriteria, 32 numFlightsAirline, 33 numFlightsAirport, 33 numFlightsAirport, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63  Airport, 13 getRumVertex Graph< T >, 39 getPosition Airport, 13 getReachableAirports FlightNetwork, 31 getReachableCities	getReachableCountries, 31	getName
nameCriteria, 32 numFlightsAirline, 33 numFlightsAirport, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63  getNum Vertex	listBestFlights, 32	Airline, 9
nameCriteria, 32 numFlightsAirline, 33 numFlightsAirport, 33 numFlightsCity, 34 flightnetwork App, 18 FlightNetwork.cpp dfs_art, 63 haversineDistance, 63  getNum Vertex	maximumTrip, 32	Airport, 13
numFlightsAirline, 33  numFlightsAirport, 33  numFlightsCity, 34  flightnetwork  App, 18  FlightNetwork.cpp  dfs_art, 63  haversineDistance, 63  Vertex < T >, 44  getNumVertex  Graph < T >, 39  getPosition  Airport, 13  getReachableAirports  FlightNetwork, 31  getReachableCities	nameCriteria, 32	-
numFlightsAirport, 33 numFlightsCity, 34  Graph < T >, 39  flightnetwork  App, 18  FlightNetwork.cpp  dfs_art, 63 haversineDistance, 63  getNumVertex  Graph < T >, 39  getPosition  Airport, 13  getReachableAirports  FlightNetwork, 31  getReachableCities		_
numFlightsCity, 34  flightnetwork App, 18  FlightNetwork.cpp dfs_art, 63 haversineDistance, 63  Graph< T >, 39 getPosition Airport, 13 getReachableAirports FlightNetwork, 31 getReachableCities		
flightnetwork getPosition App, 18 Airport, 13 FlightNetwork.cpp getReachableAirports dfs_art, 63 FlightNetwork, 31 haversineDistance, 63 getReachableCities	- · · · · · · · · · · · · · · · · · · ·	•
App, 18 Airport, 13 FlightNetwork.cpp getReachableAirports dfs_art, 63 FlightNetwork, 31 haversineDistance, 63 getReachableCities		-
FlightNetwork.cpp getReachableAirports dfs_art, 63 FlightNetwork, 31 haversineDistance, 63 getReachableCities		_
dfs_art, 63 FlightNetwork, 31 haversineDistance, 63 getReachableCities	··	-
haversineDistance, 63 getReachableCities		-
· · · · · · · · · · · · · · · · · · ·		_
FlightNetwork, 31		•
	Filgntivetwork.npp	FlightNetwork, 31

INDEX 67

getReachableCountries	main.cpp
FlightNetwork, 31	main, 64
getVertexSet	mainMenu
Graph < T >, 39	App, 17
getWeight	maximumTrip
Edge< T >, 19	FlightNetwork, 32
globalStatistics	3 , -
App, 17	name
goBackStatisticsMenu	Airline, 10
App, 17	Airport, 16
Graph $< T >$ , 34	nameCriteria
·	FlightNetwork, 32
addEdge, 36	nodesAtDistanceDFS
addVertex, 36	Graph $<$ T $>$ , 39
bfs, 36	nodesAtDistanceDFSVisit
bfsDistance, 37	
dfs, 37	Graph.hpp, 54
dfsVisit, 38	num
Edge $<$ T $>$ , 21	Vertex $<$ T $>$ , 48
EdgesAtDistanceDFS, 38	numberOfDestinations
findVertex, 38	App, 17
getNumVertex, 39	numFlightsAirline
getVertexSet, 39	FlightNetwork, 33
inDegree, 39	numFlightsAirport
nodesAtDistanceDFS, 39	FlightNetwork, 33
removeEdge, 40	numFlightsCity
removeVertex, 40	FlightNetwork, 34
Vertex $<$ T $>$ , 47	
vertexSet, 40	operator<
Graph.hpp	Airport, 13
nodeeAtDietanceDESVieit 5/	operator==
nodesAtDistanceDFSVisit, 54	
	Airport, 14
haversineDistance	
haversineDistance FlightNetwork.cpp, 63	Airport, 14
haversineDistance	Airport, 14 position Airport, 16
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52	Airport, 14  position    Airport, 16  processing
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52 inc/Airline.hpp, 49	Airport, 14 position Airport, 16
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50	Airport, 14  position    Airport, 16  processing
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55	Airport, 14  position    Airport, 16  processing    Vertex < T >, 48  reachableDest    App, 17  README.md, 62
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree	Airport, 14  position    Airport, 16  processing    Vertex < T >, 48  reachableDest    App, 17  README.md, 62 removeEdge
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree Graph< T >, 39	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62 removeEdge    Graph< T >, 40
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree Graph< T >, 39 indegree	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62  removeEdge    Graph< T >, 40  removeEdgeTo
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree Graph< T >, 39 indegree Vertex< T >, 47	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62  removeEdge    Graph< T >, 40  removeEdgeTo    Vertex< T >, 44
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree Graph< T >, 39 indegree Vertex< T >, 47 info	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62 removeEdge    Graph< T >, 40 removeEdgeTo    Vertex< T >, 44 removeVertex
haversineDistance     FlightNetwork.cpp, 63     FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree     Graph< $T>$ , 39 indegree     Vertex< $T>$ , 47 info     Edge< $T>$ , 21	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62  removeEdge    Graph< T >, 40  removeEdgeTo    Vertex< T >, 44
haversineDistance     FlightNetwork.cpp, 63     FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree     Graph< $T>$ , 39 indegree     Vertex< $T>$ , 47 info     Edge< $T>$ , 21     Vertex< $T>$ , 47	Airport, 14  position    Airport, 16  processing    Vertex < T >, 48  reachableDest    App, 17  README.md, 62  removeEdge    Graph < T >, 40  removeEdgeTo    Vertex < T >, 44  removeVertex    Graph < T >, 40
haversineDistance     FlightNetwork.cpp, 63     FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree     Graph< $T>$ , 39 indegree     Vertex< $T>$ , 47 info     Edge< $T>$ , 21	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62  removeEdge    Graph< T >, 40  removeEdgeTo    Vertex< T >, 44  removeVertex    Graph< T >, 40  setAdj
haversineDistance     FlightNetwork.cpp, 63     FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree     Graph< $T>$ , 39 indegree     Vertex< $T>$ , 47 info     Edge< $T>$ , 21     Vertex< $T>$ , 47	position Airport, 16 processing Vertex< T >, 48  reachableDest App, 17 README.md, 62 removeEdge Graph< T >, 40 removeEdgeTo Vertex< T >, 44 removeVertex Graph< T >, 40  setAdj Vertex< T >, 45
haversineDistance     FlightNetwork.cpp, 63     FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree     Graph $<$ T $>$ , 39 indegree     Vertex $<$ T $>$ , 47 info     Edge $<$ T $>$ , 21     Vertex $<$ T $>$ , 47 isProcessing	position Airport, 16 processing Vertex< T >, 48  reachableDest App, 17 README.md, 62 removeEdge Graph< T >, 40 removeEdgeTo Vertex< T >, 44 removeVertex Graph< T >, 40  setAdj Vertex< T >, 45 setCallsign
haversineDistance     FlightNetwork.cpp, 63     FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree     Graph $<$ T $>$ , 39 indegree     Vertex $<$ T $>$ , 47 info     Edge $<$ T $>$ , 21     Vertex $<$ T $>$ , 47 isProcessing     Vertex $<$ T $>$ , 44	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62  removeEdge    Graph< T >, 40  removeEdgeTo    Vertex< T >, 44  removeVertex    Graph< T >, 40  setAdj    Vertex< T >, 45  setCallsign    Airline, 9
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52 inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree Graph $<$ T $>$ , 39 indegree Vertex $<$ T $>$ , 47 info Edge $<$ T $>$ , 47 isProcessing Vertex $<$ T $>$ , 44 isVisited	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62 removeEdge    Graph< T >, 40 removeEdgeTo    Vertex< T >, 44 removeVertex    Graph< T >, 40  setAdj    Vertex< T >, 45 setCallsign    Airline, 9 setCity
haversineDistance     FlightNetwork.cpp, 63     FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree     Graph< $T>$ , 39 indegree     Vertex< $T>$ , 47 info     Edge< $T>$ , 21     Vertex< $T>$ , 47 isProcessing     Vertex< $T>$ , 44 isVisited     Vertex< $T>$ , 44	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62 removeEdge    Graph< T >, 40 removeEdgeTo    Vertex< T >, 44 removeVertex    Graph< T >, 40  setAdj    Vertex< T >, 45 setCallsign    Airline, 9 setCity    Airport, 14
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52 inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree Graph $<$ T $>$ , 39 indegree Vertex $<$ T $>$ , 47 info Edge $<$ T $>$ , 47 isProcessing Vertex $<$ T $>$ , 44 isVisited Vertex $<$ T $>$ , 44	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62  removeEdge    Graph< T >, 40  removeEdgeTo    Vertex< T >, 44  removeVertex    Graph< T >, 40  setAdj    Vertex< T >, 45  setCallsign    Airline, 9  setCity    Airport, 14  setCode
haversineDistance     FlightNetwork.cpp, 63     FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree     Graph< $T>$ , 39 indegree     Vertex< $T>$ , 47 info     Edge< $T>$ , 21     Vertex< $T>$ , 47 isProcessing     Vertex< $T>$ , 44 isVisited     Vertex< $T>$ , 44	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62 removeEdge    Graph< T >, 40 removeEdgeTo    Vertex< T >, 44 removeVertex    Graph< T >, 40  setAdj    Vertex< T >, 45 setCallsign    Airline, 9 setCity    Airport, 14 setCode    Airline, 9
haversineDistance FlightNetwork.cpp, 63 FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree Graph< T >, 39 indegree Vertex< T >, 47 info Edge< T >, 21 Vertex< T >, 47 isProcessing Vertex< T >, 44 isVisited Vertex< T >, 44  listBestFlights FlightNetwork, 32	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62 removeEdge    Graph< T >, 40 removeEdgeTo    Vertex< T >, 44 removeVertex    Graph< T >, 40  setAdj    Vertex< T >, 45 setCallsign    Airline, 9 setCity    Airport, 14 setCode    Airline, 9 Airport, 14
haversineDistance     FlightNetwork.cpp, 63     FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree     Graph < T >, 39 indegree     Vertex < T >, 47 info     Edge < T >, 21     Vertex < T >, 47 isProcessing     Vertex < T >, 44 isVisited     Vertex < T >, 44  listBestFlights     FlightNetwork, 32 low	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62 removeEdge    Graph< T >, 40 removeEdgeTo    Vertex< T >, 44 removeVertex    Graph< T >, 40  setAdj    Vertex< T >, 45 setCallsign    Airline, 9 setCity    Airport, 14 setCode    Airline, 9 Airport, 14
haversineDistance     FlightNetwork.cpp, 63     FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree     Graph < T >, 39 indegree     Vertex < T >, 47 info     Edge < T >, 21     Vertex < T >, 47 isProcessing     Vertex < T >, 44 isVisited     Vertex < T >, 44  listBestFlights     FlightNetwork, 32 low	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62 removeEdge    Graph< T >, 40 removeEdgeTo    Vertex< T >, 44 removeVertex    Graph< T >, 40  setAdj    Vertex< T >, 45 setCallsign    Airline, 9 setCity    Airport, 14 setCode    Airline, 9 Airport, 14
haversineDistance     FlightNetwork.cpp, 63     FlightNetwork.hpp, 52  inc/Airline.hpp, 49 inc/Airport.hpp, 50 inc/App.hpp, 50, 51 inc/FlightNetwork.hpp, 51, 53 inc/Graph.hpp, 54, 55 inDegree     Graph < T >, 39 indegree     Vertex < T >, 47 info     Edge < T >, 21     Vertex < T >, 47 isProcessing     Vertex < T >, 44 isVisited     Vertex < T >, 44  listBestFlights     FlightNetwork, 32 low     Vertex < T >, 47	Airport, 14  position    Airport, 16  processing    Vertex< T >, 48  reachableDest    App, 17  README.md, 62 removeEdge    Graph< T >, 40 removeEdgeTo    Vertex< T >, 44 removeVertex    Graph< T >, 40  setAdj    Vertex< T >, 45 setCallsign    Airline, 9 setCity    Airport, 14 setCode    Airline, 9 Airport, 14

68 INDEX

setDest	setInfo, 45
Edge< T >, 20	setLow, 46
setDistance	setNum, 46
Vertex $<$ T $>$ , 45	setProcessing, 46
setInDegree	setVisited, 46
Vertex $<$ T $>$ , 45	Vertex, 42
setInfo	visited, 48
Edge $<$ T $>$ , 20	vertexSet
Vertex < T >, 45	Graph $<$ T $>$ , 40
setLow	visited
Vertex < T >, 46	Vertex $<$ T $>$ , 48
setName	
Airline, 10	weight
Airport, 15	Edge $<$ T $>$ , 21
setNum	
Vertex< T >, 46	
setPosition	
Airport, 15	
setProcessing Vertex< T >, 46	
setVisited	
Vertex $<$ T $>$ , 46	
setWeight	
Edge $<$ T $>$ , 20	
showNumFlights	
App, 17	
src/Airline.cpp, 62	
src/Airport.cpp, 62	
src/App.cpp, 62	
src/FlightNetwork.cpp, 62	
src/main.cpp, 64	
statisticsMenu	
App, 17	
V. I	
Vertex	
Vertex <t>, 42</t>	
Vertex< T >, 41	
addEdge, 42	
adj, 47 distance, 47	
Edge $<$ T $>$ , 21	
getAdj, 43	
getDistance, 43	
getInDegree, 43	
getInfo, 43	
getLow, 44	
getNum, 44	
Graph $< T >$ , 47	
indegree, 47	
info, 47	
isProcessing, 44	
isVisited, 44	
low, 47	
num, 48	
processing, 48	
removeEdgeTo, 44	
setAdj, 45	
setDistance, 45	
setInDegree, 45	