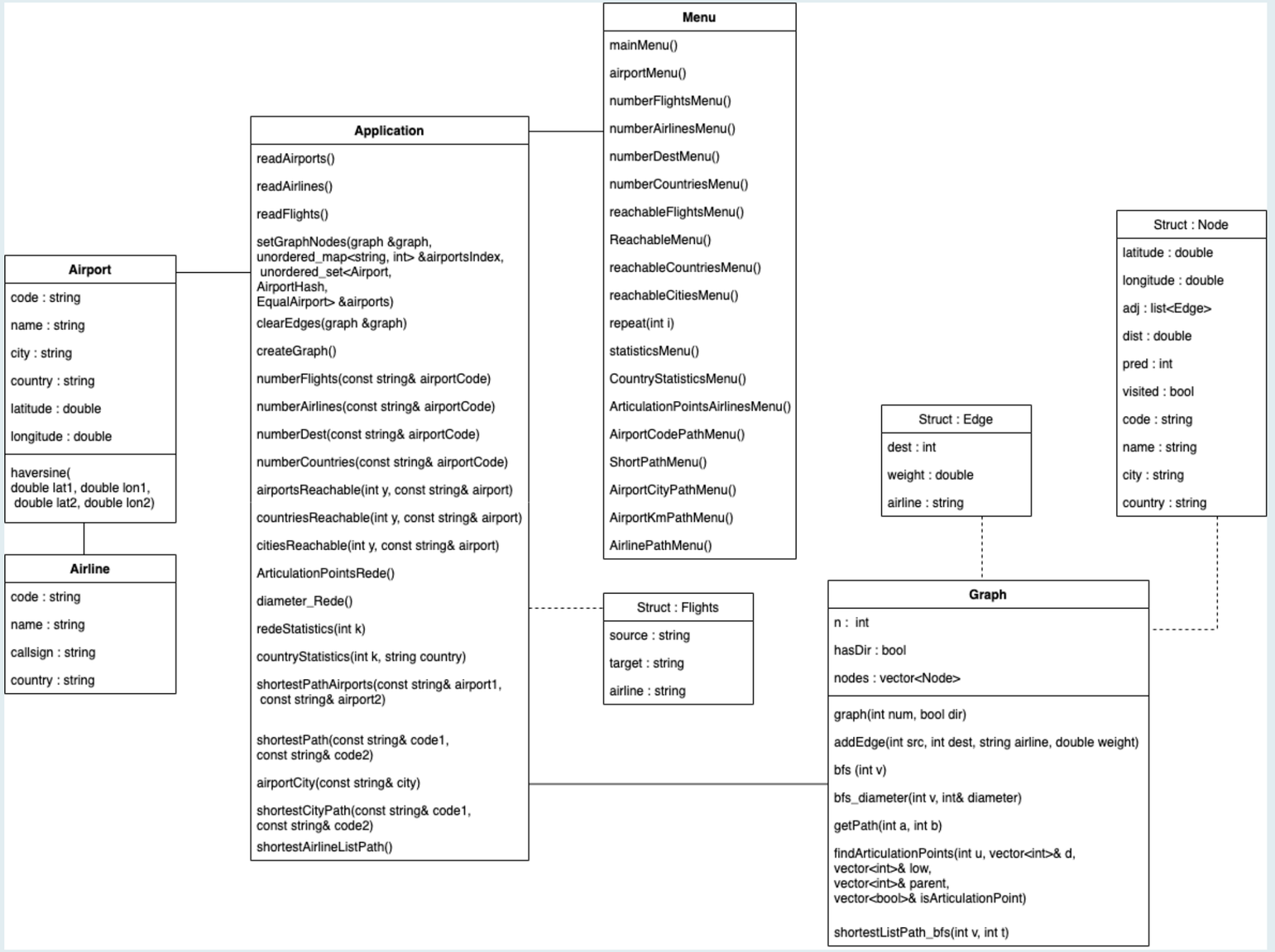
TRANSPORTES AÉREOS

DOMINGOS NETO - UP202108728
INÊS OLIVEIRA - UP202103343
LUÍS CONTREIRAS - UP202108742

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Diagrama de classes



Leitura de ficheiros

Funções de leitura

À semelhança do 1º trabalho, a leitura dos ficheiros é realizada em 3 funções diferentes. Primeiramente, efetuamos a leitura do "airports.csv" para a função readAirports() que retorna um unordered_set com toda a informação dos aeroportos. Em seguida realiza-mos exatamente o mesmo processo para a função readAirlines(), com todas as companhias aéreas.
Por fim, efetuamos a leitura de voos para um vetor, através da função readFlights().

Descrição do grafo utilizado

```
void Application::createGraph() {
    readAirports();
    this->graph1 = new graph(3019, true);
    setGraphNodes(*graph1, airportIndex, airportSet);
    flightsVector = readFlights();

    for (auto & i : flightsVector) {
        int src1 = 0;
        int trg1 = 0;
        string airline = i.airline;

        src1 = airportIndex[i.source];
        trg1 = airportIndex[i.target];

        Airport airport2 = Airport(i.source);
        auto itr = airportSet.find(airport2);
        double lon1 = itr->getLongitude();
        double lat1 = itr->getLatitude();

        Airport airport3 = Airport(i.target);
        auto itr2 = airportSet.find(airport3);
        double lon2 = itr2->getLongitude();
        double lat2 = itr2->getLatitude();

        double distance = haversine(lon1, lat1, lat2, lon2);
        graph1->addEdge(src1, trg1, airline, distance);
    }
}
```

Criação do grafo

Com a finalidade de armazenar os dados do dataset, criamos um grafo com todos os aeroportos e voos. Assim, cada node representa um aeroporto, com o código, nome, cidade, país, latitude e longitude e cada edge representa um voo, com o código do destino e o peso da viagem.

Descrição das funcionalidades implementadas e algoritmos associados

Melhor caminho

Utilização de Pesquisa em Largura (bfs()) para encontrar o caminho com menos voos de um local a outro.

Time Complexity - O(|V| + |E|)

```
vector<string> graph::shortestPath_bfs(int v, int t){
    for (int i=1; i<=n; i++) {
        nodes[i].visited = false;
        nodes[i].dist = -1;
    }
    queue<int> q;
    q.push( x: v);
    nodes[v].dist = 0;
    nodes[v].visited = true;
    nodes[v].pred = -1;
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (const auto& e :const Edge & : nodes[u].adj) {
            int w = e.dest;
            if (!nodes[w].visited) {
                q.push( x: w);
                nodes[w].visited = true;
                nodes[w].dist = nodes[u].dist + 1;
                nodes[w].pred = u;
            }
        }
    }
```

```
    vector<string> airport_route;
    airport_route.push_back(nodes[t].code);
    int current = t;
    while(current !=v) {
        current = nodes[current].pred;
        airport_route.push_back(nodes[current].code);
    }
    reverse( first: airport_route.begin(), last: airport_route.end());
    return airport_route;
}
```

Descrição das funcionalidades implementadas e algoritmos associados

Melhor caminho: aeroportos

```
vector<string> Application::shortestPathAirports(const string& airport1, const string& airport2) {
    int s = airportIndex[airport1];
    int t = airportIndex[airport2];
    vector<string> airport_route = graph1->shortestPath_bfs( v: s,t);

    return airport_route;
}
```

```
void Application::shortestPath(const string& code1, const string& code2) {
    vector<string>airport_route = shortestPathAirports( airport1: code1,  airport2: code2);
    cout << "Shortest path: " << endl;
    string airline;
    for(int i=1; i<airport_route.size(); i++) {
        string target = airport_route[i];
        string source = airport_route[i-1];
        for(auto x : Flights  : flightsVector) {
            if(x.source == source && x.target == target)
                airline = x.airline;
        }
        cout << source << " -> " << target << ": " << airline << endl;
    }
    cout << endl;
}
```

```
Introduce the departure airport's code:JFK
Introduce the arrival airport's code:BQS
Shortest path:
JFK -> BRU: UAL
BRU -> DME: DAT
DME -> BQS: TSO
```

Descrição das funcionalidades implementadas e algoritmos associados

Melhor caminho: cidades/distancias

```
Introduce the departure airport's city:New York
Introduce the arrival airport's code:BQS
Shortest path:
JFK -> BRU: UAL
BRU -> DME: DAT
DME -> BQS: TSO

Shortest path:
LGA -> IAH: UAL
IAH -> DME: TSO
DME -> BQS: TSO
```

```
Introduce the longitude:-70
Introduce the latitude:41
Introduce the arrival airport's code:BQS
Enter the maximum distance yo
u wish to travel to an airport:500
Shortest path:
JFK -> BRU: UAL
BRU -> DME: DAT
DME -> BQS: TSO

Shortest path:
LGA -> IAH: UAL
IAH -> DME: TSO
DME -> BQS: TSO

Shortest path:
BOS -> FRA: UAL
FRA -> DME: SBI
DME -> BQS: TSO

Shortest path:
EWR -> BRU: UAL
BRU -> DME: DAT
DME -> BQS: TSO

Shortest path:
ACY -> IAH: UAL
IAH -> DME: TSO
DME -> BQS: TSO
```

Descrição das funcionalidades implementadas e algoritmos associados

Informações sobre um aeroporto

1- Quantos voos existem a partir de um dado aeroporto?

Para calcular o número de voos, apenas é necessário encontrar o aeroporto, com a função find(), e retorna o tamanho do vetor de edges desse node.

Time complexity - O(n)

2- De quantas companhias aéreas diferentes?

Para realizar esta função, repetimos o inicio do método da anterior. Em seguida, percorremos a lista de edges desse aeroporto e adicionamos as airlines a um unordered_set, de moddo a remover os duplicados. Por fim, retornamos o tamanho.

Time complexity - O(|V| + |E|)

Descrição das funcionalidades implementadas e algoritmos associados

Informações sobre um aeroporto

3- Para quantos destinos diferentes? De quantos países diferentes?
A realização destas funções foi realizada com o mesmo método da anterior.
Time complexity - O(n)

4- Quantos aeroportos, cidades ou países são atingíveis usando um máximo de Y voos?
Para realizar estas funções, perguntamos ao utilizador o valor de Y e o código do aeroporto. Utilizamos o algoritmo bfs ("Breadth-First Search") para percorrer o grafo e calcular os aeroportos atingíveis com Y voos. Para calcular os paises e cidades, chamamos a função do numero de aeroportos, que retorna um unordered_set e para cada aeroporto adiciona os países/cidades a outro unordered_set.
Time complexity - airports: O(|V| + |E|) / countries, cities: O(|V| + |E| + n)

Descrição da interface com o utilizador

O programa desenvolvido inicia-se com um menu principal em que são apresentadas, de forma resumida e intuitiva, as principais funcionalidades. Assim, o utilizador tem de selecionar o que pretende fazer, através de um input. De seguida, são apresentados menus semelhantes até imprimir o resultado pretendido.
Em qualquer menu é possível voltar ao menu anterior, sem necessidade de correr o código novamente.

```
==============MAIN MENU==============
1 - Best route
2 - Airport information
3 - Airports/Countries/Cities reachable on a maximum of Y flights
4 - Statistics
5 - Articulation points
6 - Exit
=====================================
Option:
```

Destaque de Funcionalidades/Algoritmos

BFS

O algoritmo de pesquisa em largura foi fundamental para a implementação de diversas funcionalidades, que permitiram a procura das melhores rotas pelo código do aeroporto, cidade ou aeroportos a x Km, a melhor rota a através de uma airline/conjunto de airlines e o número de aeroportos/países/cidades atingíveis num máximo de Y voos.

ShortestPath_bfs(int v, int t)

Funcionalidade de Pesquisa em Largura fundamental para a implementação de diversas funções que permitiram a procura das melhores rotas, em termos de número de voos, mediante a versão de "local" utilizada.

Principais dificuldades encontradas e contribuição de cada elemento

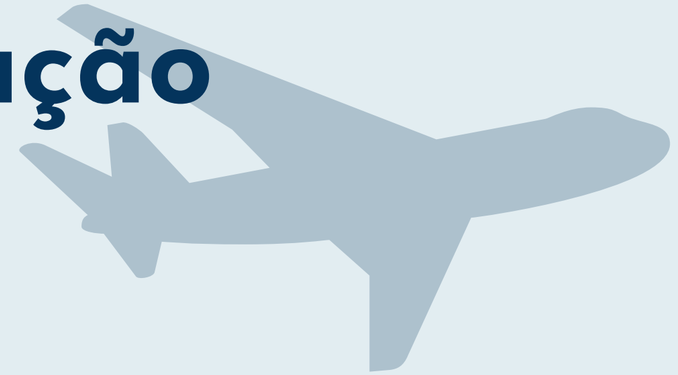
DIFICULDADES

A gestão do tempo devido à entrega ser próxima do segundo mini-teste.

ESFORÇO DO GRUPO

Domingos Neto - 33%
Inês Oliveira - 33%
Luís Contreiras - 33%

Tarefas de valorização



- Estatísticas globais da rede ou de um país

Calculo do número de aeroportos, número de voos, número de companhias, diâmetro, top-k de aeroportos com mais voos e/ou companhias.

- Pontos de articulação

Calculo dos pontos de articulação existentes na rede.

- Caminhos de diferentes Airlines com o mesmo número de voos

Se existir mais do que uma possibilidade com o menor número de voos, quando é introduzida uma cidade de partida, indicamos todas essas possibilidades (incluindo que companhias aéreas se devem usar);