Transportes Aéreos

Algoritmo e Estruturas de Dados

Grupo G117:

Luiz Queiroz 202102362

Raphael Gonçalves 202103338

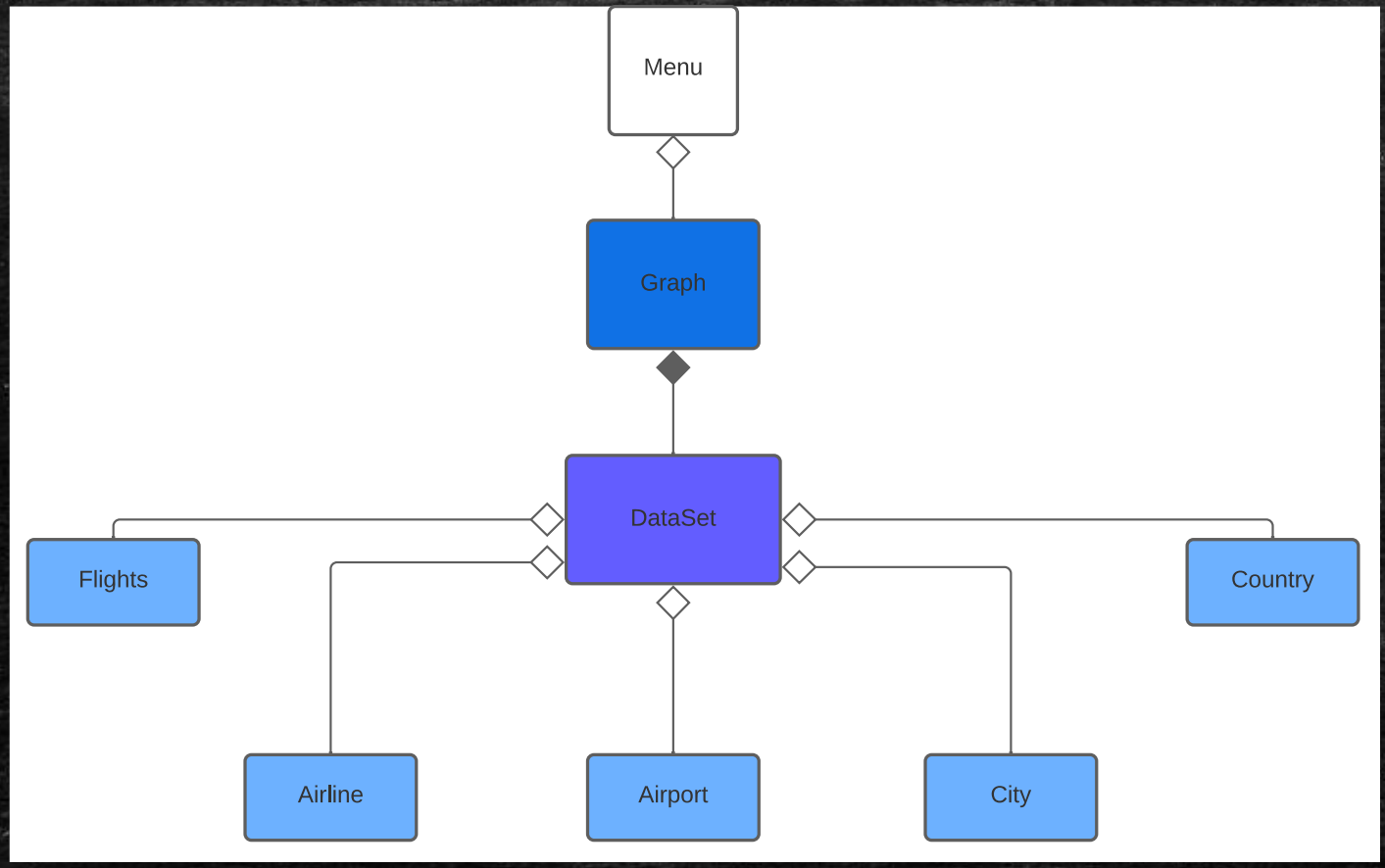
Victor Matos 202102358

Objetivo do Projeto

- Implementar um sistema capaz de providenciar ajuda para quem quer usar a rede de voos das companhias aéreas de todo o mundo.

- Por meio de um menu, as funcionalidades são implementadas nesse sistema de forma amigável.

Diagrama de Classes Utilizadas no Sistema



Leitura do Dataset

- Os arquivos airlines.csv, airports.csv e flights.csv foram lidos por meio dos métodos "readAirports", "readAirlines" e "readFlights" presentes na classe DataSet.

```
void readAirports(string filename);
void readAirlines(string filename);
void readFlights(string filename);
```

- Os dados foram guardados em tabelas hash (unordered_map), para auxilio na implementação do grafo e pesquisas mais rápidas.

```
unordered_map < string , Country* > CountriesMap;
unordered_map < string , City* > CitiesMap;
unordered_map < string , Airport* > AirportsMap;
unordered_map < string , Airline* > AirlinesMap;
unordered_map < string , int > AirportsIndex;
unordered_map < int , string > IndexAirports;
```

Grafo usado para representar o dataset

- O grafo está representado na classe Graph.
- Foi usado o grafo em forma de matriz adjacente (adjMatrix). Além disso, foi usado um array dinâmico de vetores (adj) para fazer uma conexão entre o source e o target, principalmente para auxiliar a implementação da bfs.
- O grafo serviu de suporte para obter as informações necessárias para implementar as funcionalidades do programa.

```
int numVertices;
DataSet* dataset;
vector<vector<int>> paths;
vector<int> path;
vector<int> *parent;
Flight *** adjMatrix;
vector<int>* adj;
void bfs(int start);
void find_paths(int u);
```

Funcionalidades Implementadas

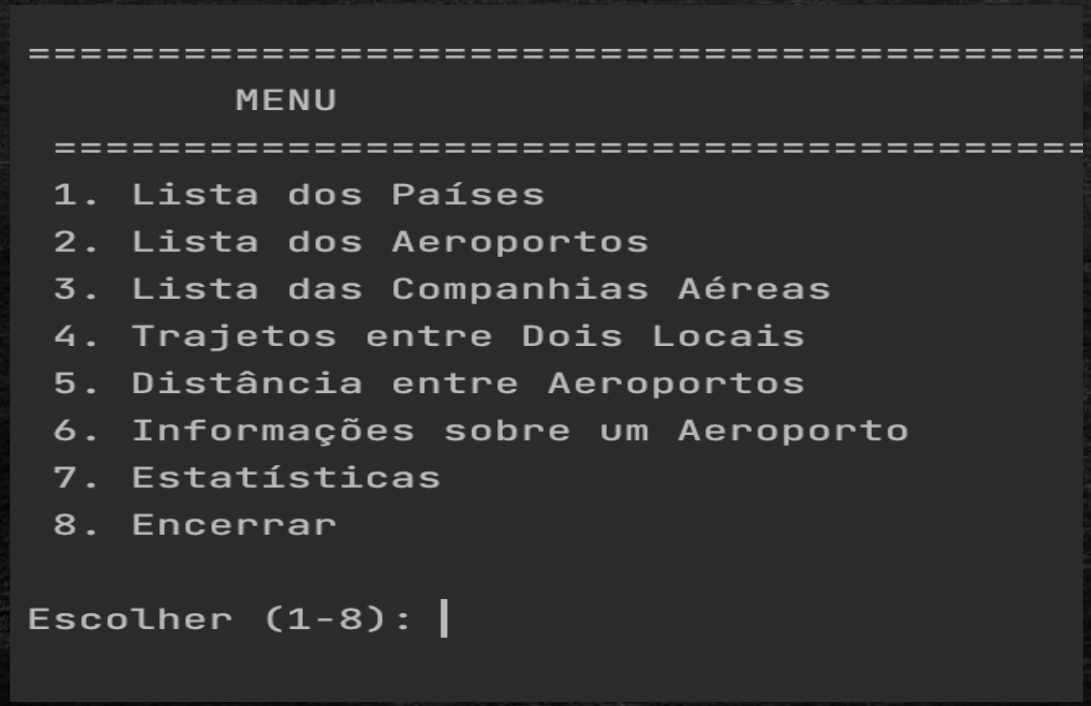
- Melhor maneira de voar de um local para o outro
- Quantas companhias aéreas diferentes em um aeroporto
- Quantos países diferentes
- Todos os destinos possíveis
- Implementação da pesquisa em largura (BSF)
- Cálculo da distância por meio da fórmula de Haversine
- Estatísticas globais de rede, de um país e de uma companhia para número de aeroportos, números de voos, números de companhias aéreas
- Top K aeroportos com mais voos ou companhias aéreas

Trajeto de um local para outro

- Essa funcionalidade permite que o usuário veja o trajeto entre dois aeroportos especificados ou entre duas cidades. No segundo caso, todos os trajetos existentes entre os aeroportos que existam nas duas cidades são considerados.
- Essa funcionalidade é implementada por meio do algoritmo BFS(Breadth First Search) que faz com que seja possível descobrir a melhor maneira de voar, já que calcula a distância mínima relacionada ao menor número de voo.

Interface do Utilizador

- O usuário interage com o sistema por meio de um Menu
- No Menu Inicial são disponibilizadas as seguintes opções para o usuário:



Interface do Utilizador

- Ao longo da utilização do Menu, irão existir Sub-Menus, totalizando 4, que irão fornecer ao utilizador as opções correspondentes as funcionalidades implementadas no programa.
- Exemplos de Sub-Menus:

```
Escolher (1-8): 7

 1. Número de Aeroportos
 2. Número de Voos
 3. Número de Companhias Aéreas
 4. Top-K de Aeroportos com mais Voos
 5. Top-K de Aeroportos com mais Companhias Aéreas
 6. Retornar ao Menu Principal
```

```
Escolher (1-8):4

 1. Trajeto entre Aeroportos
 2. Trajeto entre Cidades
```

```
1. Estatísticas Globais da Rede
2. Estatísticas de um País
3. Estatísticas de uma Companhia Aérea
```

Funcionalidade Destaque

- A funcionalidade que mais nos deixou orgulhosos foi a implementação do grafo por meio da matriz adjacente. E a organização dos dados nas tabelas hash, que permitiram a busca das informações mais facilmente.

Principais Dificuldades Encontradas

- Uma das dificuldades envolve a busca em largura(BFS). O algoritmo parava quando encontrava o trajeto possível. E não foi fácil conseguir todo os trajetos mínimos com o mesmo número de conexões.
- No método add_edge havia o problema que adicionávamos um voo de ida e outro de volta, porém não é sempre que há voos nos dois sentidos. Isso fazia com que, posteriormente, um voo inexistente fosse selecionado para calcular o trajeto, o que fazia o programar crashar.
- As busca por informações serem corretas, já que o usuário pode inserir informações erradas e necessita que isso não cause instabilidade no programa.

Esforço de cada elemento do grupo

- Luiz Queiroz 33.3%
- Raphael Gonçalves 33.3%
- Victor Matos 33.3%