> railway network --FEUP

Diogo Gomes -up201905991;
Luís Sousa -up202005832;
Rui Silveira -up202108878;
DA2023_PRJ1_G04_3;

> problema

- O programa tem como objetivo auxiliar e retirar informações sobre as linhas ferroviárias de Portugal.
- Para isso, foi fornecido dados reais sobre as mesmas (*network.csv* e *stations.csv*).

> funcionalidades

> menu.h

```
menu();
void mainMenu();
void back();
static void error(const std::string &s);
```

> Station.h

```
Station();
Station(std::string name, std::string district,
std::string municipality, std::string township,
std::string line);
Station(std::string n);
```

> fileReader.h

```
fileReader();
fileReader(std::string dir);
std::string formatStr(std::string in);
std::string unformatStr(std::string in);
void readStations(std::string dir)
void readNetwork(std::string dir);
```

> funcionalidades

> overloads.h

```
std::ostream& operator<<(std::ostream& o, Station s);
std::ostream& operator<<(std::ostream& o, Vertex* v);
std::ostream& operator<<(std::ostream& o, std::vector<Vertex*> v);
std::ostream& operator<<(std::ostream& o, std::vector<Edge*> v);
std::ostream& operator<<(std::ostream& o, Edge* e);
std::ostream& operator<<(std::ostream& o, Graph g);
std::ostream& operator<<(std::ostream& o, std::pair<Vertex*,Vertex*> vp);
std::ostream& operator<<(std::ostream& o, std::vector<std::pair<Vertex*,Vertex*>> v);
std::ostream& operator<<(std::ostream& o, std::pair<int,int> p);
std::ostream& operator<<(std::ostream& o, std::vector<std::pair<Edge*, int>> v);
```

> funcionalidades

> Graph.h

```
~Graph();
Graph();
Graph(Graph* copy);
Vertex *findVertex(const int &id) const;
Vertex *findVertex(const std::string name) const;
bool addVertex(Station s);
bool addEdge(const std::string &sourc, const std::string &dest, double w, std::string type);
bool addBidirectionalEdge(const std::string &sourc, const std::string &dest, double w, std::string
type);
bool addBidirectionalEdge(Station s1, Station s2, double w, std::string type);
std::vector<Vertex *> getVertexSet() const;
```

> funcionalidades

> Graph.h

```
void fordFulkerson(std::string src, std::string dest);
void fordFulkerson(Vertex* src, Vertex* dest);
bool dfs(Vertex* src, Vertex* dest);
bool dfs(std::string src, std::string dest);
void removePaths();
void removeFlow();
void removeVisited();
double maxInPath(std::string src, std::string dest);
double maxInPath(Vertex* src, Vertex* dest);
std::vector<std::pair<Vertex*,Vertex*>> maxPairs();
std::unordered_map<std::string, int> stList;
```

> funcionalidades

> Graph.h

```
void sortTopList();
std::vector<std::pair<std::string,double>> topDistSorted, topMunSorted, topDistOnlySameSorted,
topMunOnlySameSorted;
std::pair<int,int> costOptimization(std::string src, std::string dest);
std::pair<int,int> costOptimization(Vertex* src, Vertex* dest);
int djikstra(Vertex* src, Vertex* dest);
void removeEdge(std::string a, std::string b);
void removeEdge(Vertex* a, Vertex* b);
void removeEdge(Edge* e);
void removeVertex(Vertex* a);
void removeVertex(std::string n);
```

> funcionalidades

> Graph.h

```
void removeVertex(int n);
std::vector<std::pair<Edge*,int>> getDiffs(Graph* g, int n);
int findVertexIdx(const int &id) const;
int getNumVertex() const;
void deleteMatrix(int **m, int n);
void deleteMatrix(double **m, int n);
```

> funcionalidades

> VertexEdge.h

```
Vertex(int id, Station s);
bool operator<(Vertex & vertex) const;
int getId() const;
std::vector<Edge *> getAdj();
bool isVisited() const;
bool isProcesssing() const;
unsigned int getIndegree() const;
double getDist() const;
Edge *getPath() const;
std::vector<Edge *> getIncoming() const;
void setId(int info);
```

> VertexEdge.h

```
void setVisited(bool visited);
void setProcesssing(bool processing);
void setIndegree(unsigned int indegree);
void setDist(double dist);
void setPath(Edge *path);
Edge * addEdge(Vertex *dest, double w,
std::string type);
bool findAdj(std::string dest);
bool removeEdge(int destID);
```

> funcionalidades

> VertexEdge.h

```
Edge(Vertex *orig, Vertex *dest, double w, std::string type);
Vertex * getDest() const;
double getWeight() const;
bool isSelected() const;
Vertex * getOrig() const;
Edge *getReverse() const;
double getFlow() const;
std::string getType() const;
void setSelected(bool selected);
void setReverse(Edge *reverse);
void setFlow(double flow);
```

> interface

- O menu principal mostra as várias funcionalidades do programa, exibidas em tópicos de escolha.

```
||----------------------------||
||       RAILWAY NETWORK      ||
||----------------------------||

Choose one topic:
[1] The maximum amount of trains that can simultaneously travel between two stations.
[2] The station pairs that require the most amount of trains when taking full advantage of the existing network capacity.
[3] Where management should assign larger budgets for the purchasing and maintenance of trains.
[4] The maximum number of trains that can simultaneously travel between two stations with minimum cost.

[5] Functions with a new graph:
 - The maximum quantity of trains that can simultaneously travel between two stations in a network of reduced connectivity.
 - Provide a report on the stations that are the most affected by each segment failure.

[0] Quit.
>
```

> interface

- Duas dessas funções trabalham com a exclusão de *stations* e *edges* retiradas pelo usuário.

```
||----------------------------||
||       RAILWAY NETWORK      ||
||----------------------------||

Choose one topic:
[1] The maximum amount of trains that can simultaneously travel between two stations.
[2] The station pairs that require the most amount of trains when taking full advantage of the existing network capacity.
[3] Where management should assign larger budgets for the purchasing and maintenance of trains.
[4] The maximum number of trains that can simultaneously travel between two stations with minimum cost.

[5] Functions with a new graph:
 - The maximum quantity of trains that can simultaneously travel between two stations in a network of reduced connectivity.
 - Provide a report on the stations that are the most affected by each segment failure.

[0] Quit.
> 
```

> destaque de funcionalidades {2.1}

```
Write the name of two stations:
> Pombal
> Faro

The maximum number of trains is 2.

[1] Back to Menu.
[0] Quit.
>
```

> destaque de funcionalidades {2.2}

```
The pairs of stations are (wait a moment):
 - Entroncamento => Santar├®m
The maximum flow in all these stations is 22 trains.

[1] Back to Menu.
[0] Quit.
> 
```

> destaque de funcionalidades {2.3}

```
Choose one topic:
[1] Districts (counting interdistrict travel)
[2] Municipalities (counting interdistrict travel)
[3] Districts (not counting interdistrict travel)
[4] Municipalities (not counting interdistrict travel)
> 1

How many results?
> 2
> LISBOA => 0
> LEIRIA => 0

[1] Back to Menu.
[0] Quit.
>
```

> destaque de funcionalidades {3.1}

```
Write the name of two stations:
> Pombal
> Faro

The cost of the trip is 96, (you will use 2 trains).

[1] Back to Menu.
[0] Quit.
>
```

> destaque de funcionalidades {4.1 && 4.2}

```
Do you want to remove something:
[1] Yes, remove
[2] No
> 1

What do you want removed:
[1] Station
[2] Edge
[3] I don't want to remove
> 1

Write the name of the station:
> Pombal

What do you want removed:
[1] Station
[2] Edge
[3] I don't want to remove
> 3
```

> destaque de funcionalidades {4.1}

```
Choose one topic:
[1] The maximum amount of trains that can simultaneously travel between two stations.
[2] Provide a report on the stations that are the most affected by each segment failure.
[3] Go back.
> 1

Write the name of two stations:
> Nine
> Faro

The maximum number of trains is 2.
```

> destaque de funcionalidades {4.2}

```
Choose one topic:
[1] The maximum amount of trains that can simultaneously travel between two stations.
[2] Provide a report on the stations that are the most affected by each segment failure.
[3] Go back.
> 2

How many results?
> 3
 - Santana Cartaxo => Reguengo - Vale da Pedra - Pont├®vel (changed by 2).
 - Vila Pouca do Campo => Casais (changed by 2).
 - Vila Pouca do Campo => Ameal (changed by 2).
```

> informação adicional

- Ao iniciar o programa é dada a opção de escolha de *datas* (o data set fornecido e o data de teste).

```
Choose one topic:
[1] Real data.
[2] Test data.
> 
```

- Ao longo de todo o *menu* existe sempre a opção de voltar ao *main menu* e de fechar o programa.

```
[1] Back to Menu.
[0] Quit.
> 
```