Rail Network Management

Projeto nº 1

- António Rodolfo de Almeida Seara Ferreira up202108834
- João Afonso Oliveira Teixeira Santos up202108805
- João Pedro Seabra Pedrosa Botelho de Sousa up202106996

Leitura dos dados fornecidos

A leitura dos dados fornecidos nos ficheiros csv realizou-se através de 3 funções pertencentes à classe manager que conseguiam ler os dados através da função getline, armazenando estes em vetores.

Cada vez que o programa é iniciado, a função manager é chamada dando load a todos os dados sendo fácil e simples o seu acesso e utilização para diversos fins.

ENUMERAÇÃO DE CLASSES

- *Classe Graph (Criação e implementação do grafo)*
- *Classe Network (Dados sobre as networks)*
- *Classe Station (Dados sobre as estações)*

Classe Graph

```cpp
class Graph {
public:
    Graph();
    void addStation(const Station& station); // node
    void addNetwork(const Network& network); // edge
    vector<Station> getAdjacentStations(const string& stationName) const;
    int getNetworkCapacity(const string& station_A, const string& station_B) const;
    void setResidualCapacity(const string& station_A, const string& station_B, int flow);
    int getResidualCapacity(const string& station_A, const string& station_B) const;
    bool bfs(const string& source, const string& destination, unordered_map<string, string>& parent);
    int maxFlow(const string& source, const string& destination); // 2.1
    void findMostTrainsRequired(); // 2.2
    int maxNumOfTrainsArrivingAt(const string& station); // 2.4
    void topTransportationNeedsDistrict(int k); // 2.3
    void topTransportationNeedsMunicipality(int k); // 2.3
    static Graph createSubGraph(const Graph& graph, const string& line, int num); // 4.1
    int maxFlowMinCost(const string& source, const string& destination); // 3.1
private:
    unordered_map<string, Station> stations;
    unordered_map<string, vector<Network>> stationNetworks;
    unordered_map<string, unordered_map<string, int>> initialCapacities;
};
```

Menu principal

```
Select what you pretend to do:

1: Service Metrics
2: Calculate the maximum amount of trains that can simultaneously travel between two specific stations with minimum cost for the company
3: Reliability and sensitivity to line failures
4: Exit

>>
```

Menu principal com três opções que levam a sub-menus onde se vão realizar as várias operações descritas no enunciado do problema!

Service Metrics

```
Select what you pretend to do:

1: Find maximum number of trains between 2 stations
2: Find station pairs with highest max flow
3: Top-k municipalities and districts, regarding transportation needs
4: Find the maximum number of trains that can simultaneously arrive at a given station

>>
```

- Depois de escolher a opção “Service Metrics” no Menu Principal é se redirecionado para este sub-menu
- Este sub-menu tem vários algoritmos para responder aos requesitos de 2.1-2.4 todos usando o algoritmo de Edmons-Karps

Service Metrics

Exemplos de execução:

```
Select what you pretend to do:

1: Find maximum number of trains between 2 stations
2: Find station pairs with highest max flow
3: Top-k municipalities and districts, regarding transportation needs
4: Find the maximum number of trains that can simultaneously arrive at a given station

>> 1

Enter the source station name: Faro

Enter the destination station name: Pombal

Maximum number of trains that can simultaneously travel: 4
```

Operation Cost Optimization

```
Select what you pretend to do:

1: Service Metrics
2: Calculate the maximum amount of trains that can simultaneously travel between two specific stations with minimum cost for the company
3: Reliability and sensitivity to line failures
4: Exit

>> 2

Enter the source station name: Lisboa Oriente

Enter the destination station name: Braga
```

- Depois de escolher a segunda opção no Menu Principal é se redirecionado para este sub-menu
- Este sub-menu calcula o número máximo de comboios que podem viajar entre duas estações pelo menor preço para a companhia
- Esta funcionalidade usa o algoritmo de Edmons-Karps e o algoritmo de Dijkstra

Reliability/Sensivity to line failures

```
>> 3

Select what you pretend to do:

1: Maximum number of trains that can simultaneously travel between two specific stations in a network of reduced connectivity
2: Report on the stations that are the most affected by each segment failure
```

- Depois de escolher a terceira opção no Menu Principal é se redirecionado para este sub-menu
- Este sub-menu contém duas opções para realizar operações com linhas de capacidade reduzida

Exemplos de funcionalidades

- *Criação de uma interface simples "user-friendly"*

- *Leitura dos dados presentes nos ficheiros .csv*

4.1. Basic Service Metrics
The management team wishes to analyze the number of trains that it should purchase in order to serve the railway network fully. To support their needs, you should implement efficient algorithms to help them in responding to the following company requirements:

[T2.1: 3.5 points] Calculate the maximum number of trains that can simultaneously travel between two specific stations. Note that your implementation should take any valid source and destination stations as input;

[T2.2: 2.0 points] Determine, from all pairs of stations, which ones (if more than one) require the most amount of trains when taking full advantage of the existing network capacity;

[T2.3: 1.5 points] Indicate where management should assign larger budgets for the purchasing and maintenance of trains. That is, your implementation should be able to report the top-k municipalities and districts, regarding their transportation needs;

[T2.4: 1 point] Report the maximum number of trains that can simultaneously arrive at a given station, taking into consideration the entire railway grid.

4.2. Operation Cost Optimization
Management has settled on two types of services to be made available to the public: STANDARD and ALFA (PENDULAR). Each service requires different trains and has distinct train connections, which results in different costs for the company. For the purposes of your work, you can consider that the STANDARD service in a segment of the line has a fixed cost of 2€ per train while the ALFA service will cost the company 4€ per train in the same segment. Note that this is the cost of operation for the company, not the fare price for the customers, and that this cost is per train and per segment. Naturally, management would like to understand how to allocate trains to each segment in order to minimize their costs, while maintaining the same level of service. To this effect, you should provide an "optimization" analysis that takes these restrictions into consideration:

[T3.1: 2.0 points] Calculate the maximum amount of trains that can simultaneously travel between two specific stations with minimum cost for the company. Note that your system should also take any valid source and destination stations as input;

4.3. Reliability and Sensitivity to Line Failures
To enhance the reliability of the railway network, management needs to make periodic maintenance and occasional repairs. This is a complex activity and one of its key aspect is forecasting of failures and repair needs. To help management, you are to implement a functionality to assess the sensitivity of some segments to failures as they may occasionally be unavailable due to a wide number of reasons (repairs, services, disasters, etc.)/ Specifically, your implementation should provide the following data:

[T4.1: 2.5 points] Calculate the maximum number of trains that can simultaneously travel between two specific stations in a network of reduced connectivity. Reduced connectivity is understood as being a **subgraph** (generated by your system) of the original railway network. Note that your system should also take any valid source and destination stations as input;

[T4.2: 1.5 points] Provide a report on the stations that are the most affected by each segment failure, i.e., the top-k most affected stations for each segment to be considered.