Faculdade de Engenharia da Universidade do Porto

Programming Project I

An Analysis Tool for Railway Network Management

Desenho de Algoritmos, 2022/23

Licenciatura em Engenharia Informática e Computação

Grupo 2 - Turma 2

Bruno Fernandes up202108871

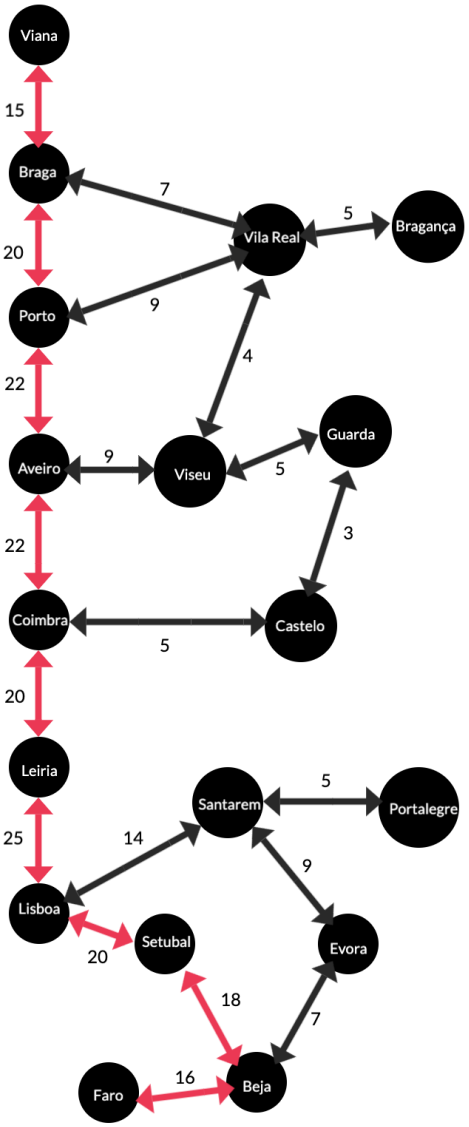
Diana Martins up202108815

Vasco Oliveira up202108881

Index

Introduction

Our system is designed to manage a train network with multiple stations and routes. It can help users to analyze the train network and optimize transportation needs. In this presentation, we will show you how to use the system with demo data.
For this purpose, we drafted two files of data for demonstration purposes and, also, to help us test the correctness of our algorithms: “demoNetwork.csv” and “demoStations.csv”. The “demoNetwork.csv” file contains information about the train network, including the source station, destination station, line capacity and service type of the trains. The “demoStations.csv” file contains details about the stations: name, district, municipality, township, and line.



Demo Graph

Reading the files

We start by reading the files chosen by the user. For that we have the readStations() and readNetworks() functions.

The first function, readStations(), reads a csv file containing information about stations and creates vertices in the network for each unique station name. The function removes any whitespace and accents from the station name, district, municipality, township and line before inserting the station into the stations unordered map and adding it as a vertex in the network.
Time complexity: O(n), where n is the number of stations in the input file.

The second function, readNetwork(), reads a file containing information about connections between stations and creates edges in the network for each connection. The function removes any whitespace and accents from the station names and replaces any whitespace in the service with a hyphen. It then adds a bidirectional edge between the vertices representing station A and station B in the network, with the capacity specified in the file and the service specified in the file.
Time complexity: O(n), where n is the number of lines in the input file.

The interface

Our interface is divided into a start menu and a main menu.

The start menu is to allow users to choose which files they want to read:
1. Read standard files;
2. Read other files;
3. Read demo files.
0. Exit.

The main menu provides users with several options to explore the train network system:
1. Display all stations;
2. Browse stations;
3. Discover the maximum number of trains that can simultaneously travel between two stations;
4. Determine which pairs of stations require the most amount of trains;
5. Discover the top-k municipalities and districts regarding their transportation needs;
6. Discover the maximum number of trains that can simultaneously arrive at a station;
7. Discover the maximum amount of trains that can simultaneously travel between two stations with minimum cost;
8. Discover the maximum number of trains that can simultaneously travel between two stations with just 1 type of service;
9. Discover the top-k most affected stations for each segment.
0. Exit.

To select an option, the user needs to input the corresponding number and press Enter.

System Functions

Option 1: Display all stations

This option displays the name, district and line of all the stations in the network.
Time complexity: O(V), where V is the number of vertices (stations) in the network.

Option 2: Browse stations

This option displays the name, municipality, district, township, and line of a specific station in the network. To use this, the user needs to provide the name of the station he wants to browse.
Time complexity: O(V), where V is the number of vertices in the network.

Option 2.1: Display a station’s connections

This option displays all the stations which are directly connected to the station that the user has provided in option 2.
Time complexity: O(E), where E is the number of edges the station provided in option 2 has.

Option 3: Discover the maximum number of trains that can simultaneously travel between two stations

This option calculates the maximum number of trains that can travel between two stations simultaneously. To use this, the user needs to provide the source and destination stations.
Time complexity: O(V * E^2), where E is the number of edges and V is the number of vertices in the network.

Let's say we want to know the maximum number of trains that can simultaneously travel between Viana and Vila. We can use the Edmonds Karp algorithm to solve this problem. The output should be "Maximum number of trains between Viana and Vila: 15", with 7 trains traveling from Viana to Braga and then to Vila, and 8 trains traveling from Viana to Porto and then to Vila.

Option 4: Determine which pairs of stations require the most amount of trains

This option determines the pairs of stations that require the most amount of trains. To use this function, the user needs to input the number of pairs of stations to show.
Time complexity: O(V^3 E^2), where V is the number of vertices and E is the number of edges in the graph.

We can use the Edmonds Karp algorithm to solve this problem. The output should be "Max flow: 27. There are 2 stations with max flow: (Braga, Porto) and (Lisboa, Setubal)". The maximum flow between Braga and Porto is achieved by sending 11 trains from Braga to Porto and 16 trains from Porto to Braga. Similarly, the maximum flow between Lisbon and Setubal is achieved by sending 14 trains from Lisbon to Setubal and 13 trains from Setubal to Lisbon.

Option 5: Discover the top-k municipalities and districts regarding their transportation needs

This option discovers the top-k municipalities and districts regarding their transportation needs. To use this function, the user needs to input the number of municipalities and districts to show.
Time complexity: O(V*log(V) + E), where V is the number of vertices and E is the number of edges in the graph.

The algorithm first computes the transportation needs of each district and each municipality by multiplying the stations' cost and capacity, adding them all and then storing the values in a vector. That is then sorted in descending order of transportation needs and selects the top districts and municipalities. If we choose 5 as the value of k, the output should be “Top 5 districts by transportation needs: Lisboa: 208, Aveiro: 194, Porto: 186, Leiria: 180, Coimbra: 178. Top 5 municipalities by transportation needs: Lisboa: 208, Aveiro: 194, Porto: 186, Leiria: 180, Coimbra: 178”.

Option 6: Discover the maximum number of trains that can simultaneously arrive at a station

This option calculates the maximum number of trains that can simultaneously arrive at a station. To use this function, the user needs to provide the station name.
Time complexity: O(V * E^2), where V is the number of vertices and E is the number of edges in the graph.

We can choose Vila as our destination station. The output should be: “Maximum amount of trains that can simultaneously arrive at Vila: 25”. In this case, those trains would be (source station: number of trains): Braga: 7 trains, Porto: 9 trains, Bragança: 5 trains, Viseu: 4 trains, because of the shortest paths’ capacity.

Option 7: Discover the maximum amount of trains that can simultaneously travel between two stations with minimum cost

This option calculates the maximum number of trains that can simultaneously travel between two stations with minimum cost. To use this function, the user needs to provide the source and destination stations.
Time complexity: O(E log(V)), where V is the number of vertices and E is the number of edges in the graph.

Let's say we want to know the maximum number of trains that can simultaneously travel between Viana and Vila with minimum cost. We start by using the Dijkstra algorithm, then we calculate the costs for all the paths resulting from it and choose the one with the minimum cost. The output should be “The maximum number of trains that can travel simultaneously between Viana and Vila is: 7. The path is: Viana -> Braga -> Vila. The minimum cost for the company while maintaining the same level of service is: 42 euros.”

Option 8: Discover the maximum number of trains that can simultaneously travel between two stations with just 1 type of service

This option calculates the maximum number of trains that can simultaneously travel between two stations with just one type of service. To use this function, the user needs to provide the source and destination stations and the service type.
Time complexity: O(V * E^2), where E is the number of edges and V is the number of vertices in the graph.

Let's say we want to know the maximum number of trains that can simultaneously travel between Viana and Vila with standard service. We can use the Edmonds Karp algorithm to solve this problem. The maximum flow returned by this function represents the maximum number of trains that can travel simultaneously between the two stations, considering only the "standard" service. The output should be "There is no path between Viana and Vila using only the STANDARD service", as there are only ALFA-PENDULAR trains leaving Viana.

Option 9: Discover the top-k most affected stations for each segment

This option discovers the top-k most affected stations for each segment. To use this function, the user needs to input the number of stations to show and the stations which to be removed.
Time complexity: O(V^2 * E^2), where V is the number of vertices and E is the number of edges in the graph.

The function works by first finding all the shortest paths between stA (Evora) and stB (Santarem) using Dijkstra's algorithm. Next, we traverse all the nodes on the shortest paths and count the number of times each node appears on the paths. Finally, we return the top-k most affected stations, sorted in descending order of their counts. The output using k=5 should be “Evora Decay of Max Simultaneous Trains: 9; Beja Decay of Max Simultaneous Trains: 7; Santarem Decay of Max Simultaneous Trains: 7; Setubal Decay of Max Simultaneous Trains: 2; Aveiro Decay of Max Simultaneous Trains: 0”.

Students’ effort

The workload was evenly distributed and each member of the team contributed equally.