DESIGN OF ALGORITHMS – FIRST PROJECT

Done by:

-Carlos Daniel Lopes Rebelo

-Diogo Tomás Valente Fernandes

-Jaime Francisco Rodrigues Fonseca

CLASSES

- The main classes created for this project were:
  - **Station** – Represents the nodes in our graph
  - **Track** – Represents the edges in our graph
  - **Network** – Graph class

  There are also some classes such as **Menu** (responsible for exposing our projects functionalities to the user in a human-friendly way); **Utils** (has some utility functions used throughout the whole project) ; **NetworkManager** (used to temporarily create subgraphs necessary for some of the functionalities implemented).

FUNCTIONALITIES

- We are going to do a brief explanation on the way we interpreted each of the points presented on the project description as well as how we implemented its solution.

Point 2.1 (“Calculate the maximum … “)

Clearly a max-flow problem. Therefore we just implemented a simple Edmonds-karp algorithm that takes a source and a destination as arguments.

Point 2.2

Which pair(s) of nodes have the most amount of flow in the entire graph. We run Edmonds-karp with all the possible pair combinations on the graph keeping track which one(s) have the highest amount of flow. In the end we return the pairs calculated.

Point 2.3

This implementation is very similar to the previous one. The only difference is that we keep track of the flow on each district or municipality using a vector of pairs called "res". Every time we run the Edmonds-Karp algorithm on a pair of nodes, we update the flow in res. For example, if we run Edmonds-Karp between Aveiro-Campanhã and get a max-flow of 5, we add 5 to Aveiro's total flow. We do the same for Porto.

We return the results in non-increasing order. We interpret "top transportation needs" as meaning that we should increase our budget in the districts or municipalities with the highest flow. This is because in a real-life situation, they would need more resources and maintenance. If we were to return the results in increasing order, the project description should specify that we are looking for where to invest more.

Point 2.4

In this point, we need to consider the entire railway as the source but only one specified sink (given as an argument). We realized that running Edmonds-Karp with all the nodes in the graph as sources and one sink would result in the sum of the edge capacities leading to our sink. This would be a meaningless solution. In order to have a more interesting solution, we run a DFS on the incoming edges of our sink. Whenever the DFS finds an unvisited node, it adds that node to a vector. We use this vector to calculate the "original sources in the graph".

Next, we create a super source that is connected to all of those original sources, and then we run Edmonds-Karp between the super source and our sink producing a much more useful result in the end.

Point 3.1

The thought process behind this step involved initially saving all the possible flows and associated costs. However, in some cases, a maximum flow solution may involve multiple paths. In such instances, we needed to identify the path with the lower cost. To achieve this, we utilized a pair "flowCost" which was saved in "all_flowCosts" and computed using the function "findMinResidualCostAlongPath". This function not only calculates the flow as per the standard procedure, but also adds the cost of service, which is 2 for standard and 4 for "alfa". Due to the cost being calculated per segment rather than per track, it is not feasible to simply accumulate the cost along the track. Instead, we must multiply the cost of each segment by the flow of the path, which represents the number of trains that traverse that path. This is crucial for accurately computing the total cost of the railway network. We saved all the flowCosts that were possible and subsequently sorted them based on the previously defined parameters.

Point 4.1

A temporary sub-graph is created based on the original graph through the networkManager class. Then we run the Edmonds Karp algorithm on this new sub-graph created.

Point 4.2

Initially, we implemented the Edmonds-Karp algorithm for all possible pairs of stations in the original graph. Then, we proceeded to repeat the same procedure on a new subgraph, which was constructed using the same method as in the previous step. Subsequently, we compared the values obtained from each pair and calculated the difference between these two values, namely the maximum flow in the original graph and the maximum flow in the subgraph. Lastly, we associated each city with its corresponding difference, and presented to the user the cities with the most significant differences.

PARTICIPATION

33%	33%	33%
Carlos Rebelo	Diogo Fernandes	Jaime Fonseca