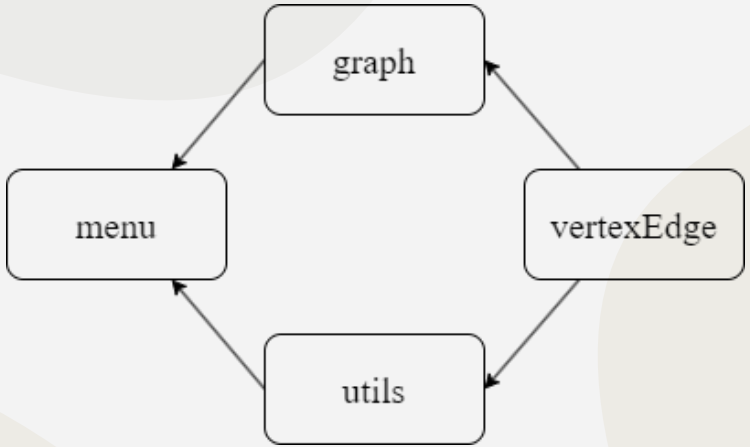
RAILWAY NETWORK

MANAGEMENT & ANALYSIS

João Alves – up202108670
Eduardo Sousa – up202103342

Grupo 53

CLASS DIAGRAM



utils: used for dataset reading

vertexEdge: representation of the stations and networks

menu: simple way to use the functions

graph: where all the main algorithms are

DATASET REPRESENTATION

- In order to represent the provided data set, we decided to create only one bidirectional graph
- For the topics that required a residual graph, we decided to momentarily remove the edges (networks) required by the user, and at the end reset them all
- Pointers were used at lot in order to have more efiency
- All the algorithms applied to the data set are in the graph class as well as their time complexities

APPROACH TO TOPICS

- [T2.1] approach

Algorithms: edmondsKarp and findAugmentingPath

Description: basic max-flow calculation between a source station and target station using the Edmonds-Karp algorithm

- [T2.2] approach

Algorithms: findMostTrainsRequired

Description: calculate the flow between all, non-repeated stations, using an unordered_map to save the intermediate results in order to save the max flow possible stations at every moment of the execution of the function, and at the end return them in a vector for simpler use

APPROACH TO TOPICS

- [T2.3] approach

Algorithms: getMaxFlowPerMunicipality and getMaxFlowPerDistrict

Description: use an unordered_map to calculate the sum of flows to each the district/municipality, then place them in vector to sort by the flows

Approach: the more flow the district/municipality have the more priority they will have to have more investment

- [T2.4] approach

Algorithms: findMostStationTrains

Description: search the graph source nodes (the ones that have only one adjacente vertex) create a super_source station with infinite capacity, connect it with those source nodes, calculate the maximum flow between that super_source and the wanted station, and at the end remove everything related to the super_source node, so the graph is left as it was

APPROACH TO TOPICS

- [T3.1] approach

Algorithms: maxTrainsMinCost

Description: while there's an augmenting path from source station to target station: a normal dijkstra combined with setting paths and analysing flows and capacity, will run; then it will discover the bottleneck to the path discovered in the first part; and finally it will calculate the cost. That way the lowest cost for the maximum flow between the station will be calculated

Approach: we tried an approach to the max-flow min-cost theorem, keeping the maximum flow with Edmonds-Karp algorithm and minimum cost with Dijkstra algorithm working together

APPROACH TO TOPICS

- [T4.1] approach

Algorithms: reducedConnectivity and askForRemovedEdge

Description: ask for networks that the user wants to get removed due to some reason. Save them in a list and remove them from the graph, generating a subgraph. Then run edmondsKarp algorithm used in [T2.1] to get the maximum flow between the source station and the target station with reduced connectivity. At the end add the removed networks, that are currently in the list, back to the graph.

- [T4.2] approach

Algorithms: topKMostAffected and askForRemovedEdge

Description: ask for networks that the user wants to get removed due to some reason. Calculate the maximum flow of all stations before the networks removal and save it in a vector, and then do the same after the removal (this uses the [T2.4] fuction). Then calculate the difference between the results obtained before and after the removal and show the results in form of affected flow percentage or flow difference itself. At the end add the removed networks, that are currently in the list, back to the graph.

INTERFACE

- Simple and easy to use menu was also implemented
- All the topic previously referred are represented in order in the menu, being the [1] option referred to [T2.1], the [2] option referred to [T2.2] … and the[7] option referred to [T4.2]
- All options of each topic are presented after selection one topic
- Menu is very text sensitive, so it is important to write everything as it is in the dataset or how it is asked in the menu

MAIN ALGORITHM

The [T3.1] has our most laborious and complex algorithm as it englobes the ideas from the Dijkstra and Edmonds-Karp algorithms in order to follow the min-cost max-flow theorem, so the result will always be the minimum cost for the most amount of trains that can simultaneously between two stations