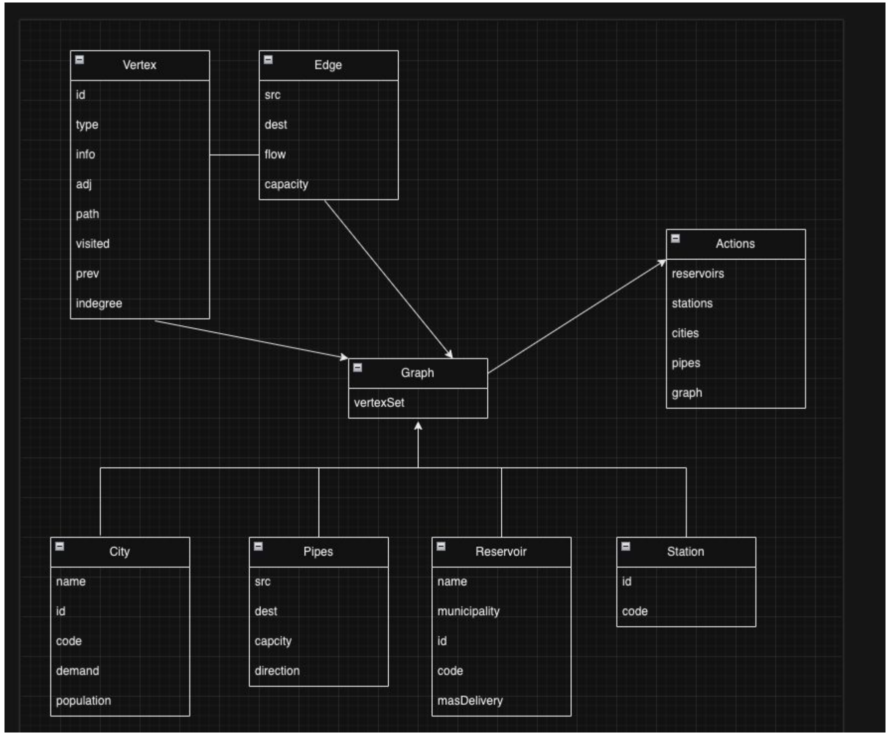
Water Supply Management Analysis Tool

Project nº1 for the course of Algorithms Design

by:

- Guilherme Santos up202105090
- Alexandre Lopes up202207015
- Tomás Silva up202108698

Class Diagram

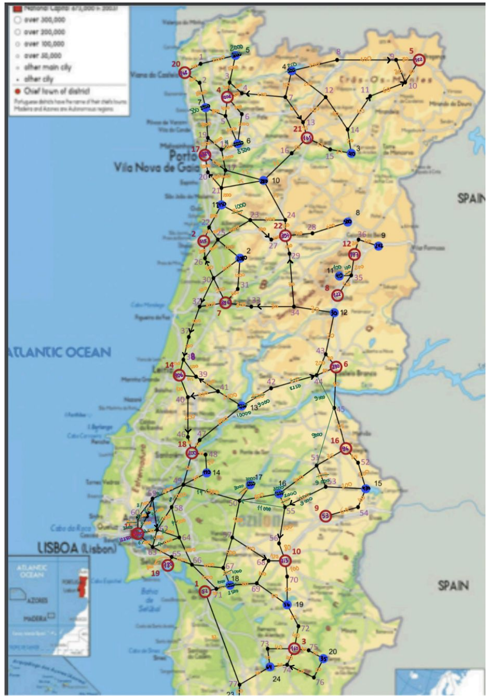


Reading each Dataset

To read and parse each dataset we implemented 4 different functions that would each parse every Reservoir, Pipe, Station and City. We also created a function that would Map each city code to the correspondent city name, this will only be used to make the outputs more user friendly. To parse each dataset we start by creating a new vector where we will store every different object of the same type for example a Reservoir. After that we open the corresponding file and start creating the objects line by line. In the end we return the vector with all of the different objects.

Graph

The graph we used is based on the map of all the reservoirs, pipes, pumping stations and cities. Each pipe is represented by an edge or two depending on the direction of the edge and it's weight is the capacity of the pipe. There are 3 types of vertices, pumping stations, cities and reservoirs. Pumping stations have a code and can have incoming and outgoing edges. Cities have their code too as well as the demand and can only have incoming edges. Reservoirs also have a code and it's maximum delivery and only have outgoing edges.



Determine the maximum amount of water that can reach each or a specific city.

To find out the maximum amount that can reach each city we used the Edmonds-Karp algorithm. First we add a super-sink and a super-source to the graph, the super-sink is connected to all cities and the super-source connected to all reservoirs, we then use the algorithm for those two vertices. In the end we remove those two vertices so that the graph returns to its original state. We then return the maxFlow that can reach the city.

For all cities we just iterate through all cities and perform the same algorithm.

Time complexity: O(VE^2)

Can an existing network configuration meet the water needs of its customer?

First we run the maxFlow for all cities used in the last exercise, after that we compare it to the demand of all cities. If the demand is higher than the maxFlow we output the city because it does not meet the water needs of the customer.

Time complexity: O(VE^2)

Balance the load across the network.

Firstly, we calculate the maxFlow by using the edmonds-karp algorithm. These flow values will be used to calculate the initial metrics, which will be outputted to the user. After that step, we then call an heuristic evaluation function which is divided in two main steps. The first step gathers all the cities where the demand is being met, after that we take those cities incoming edges and stores them on a vector. After that we take the edges of each city and calculate the best demand that minimizes the difference between all of those edges, which is the cities demand divided by the number of edges incoming. Using that value we balance all that flow through all of the pipes. The only case where this approach does not work is when there are edges before the cities incoming edges with smaller capacity and when there are edges with a smaller value then the max value. The second step, balances the edges immediately after a reservoir by calculating the reservoir residual water and distributing that water to the outgoing edges that are not full.

Time complexity: O(E^2)

Evaluate the network’s resiliency if one specific water reservoir is out of commission.

First we ask which reservoir the user wants to remove, we then calculate the edmonds-karp with the unmodified graph to see what is the maxFlow with that reservoir. After that we put the capacity of all pipes coming out of the reservoir at 0 and it basically means that the reservoir was removed. After that we calculate the edmonds-karp without that reservoir and get a new maxFlow. In the end we check if there is an impact in removing the reservoir, if there is an impact we output the city code, the old maxFlow and the new maxFlow and it’s difference. If there isn’t an impact the output says that there is no impact.

Time complexity: O(VE^2)

Check if any pumping station can be temporarily taken out of service without affecting the delivery capacity.

First we start by calculating again the maxFlow for all cities, then we iterate through each station and create a temporary graph, for each edge adjacent to a station we change its capacity and flow to 0. After that we calculate the maxFlow for the temporary graph. Then we iterate through each city, in case the flow has been affected we store in a vector its code and respective deficit. After we finish iterating through every city we store in a map this vector and the key will be the removed station code.

Time complexity: O(VE^2+SA), where S=Stations and A=Adjacent edges of the station

Determine which pipelines, if ruptured, would make it impossible to deliver the desired amount of water to a given city.

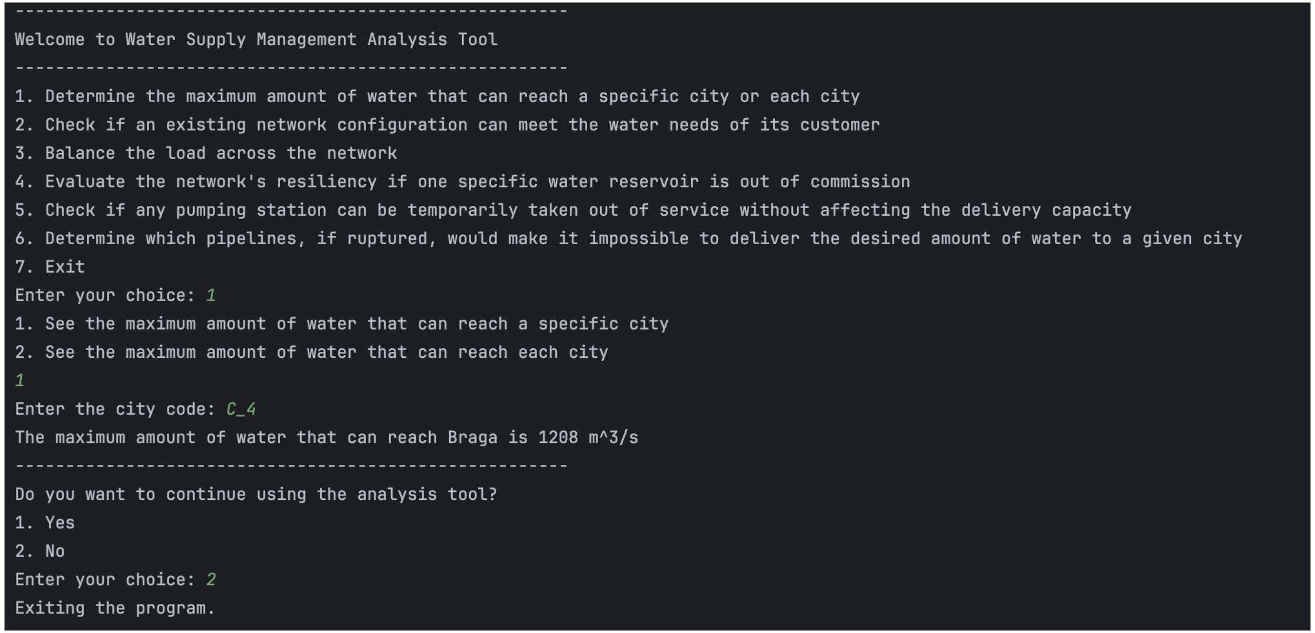
First we calculate again the maxFlow for the graph, then we iterate through all pipes and we execute an algorithm to handle differently if the pipe is uni or bi directional. In that algorithm we store the original flow and capacity of the pipe, after that we put the flow and capacity at 0 and calculate the maxFlow, after that we restore the original values. After that we iterate through all cities and if the flow is affected we calculate the deficit.

Time complexity: O(VE^2 + P * (VE + VE^2 +C))

The user interface

We implemented a simple user interface that lets the user choose which functionality he wants to use for the problem he wants to solve. The menu first asks the user for a input from 1 to 7 to choose the functionality or exit the program. After the user chooses the functionality he wants he can get new prompts, such as codes that are needed to solve the problem. The user will then get the output in the console and will be asked if he wants to continue using the program.

Example of user interface for exercise 2.1



```
-------------------------------------------------------
Welcome to Water Supply Management Analysis Tool
-------------------------------------------------------
1. Determine the maximum amount of water that can reach a specific city or each city
2. Check if an existing network configuration can meet the water needs of its customer
3. Balance the load across the network
4. Evaluate the network's resiliency if one specific water reservoir is out of commission
5. Check if any pumping station can be temporarily taken out of service without affecting the delivery capacity
6. Determine which pipelines, if ruptured, would make it impossible to deliver the desired amount of water to a given city
7. Exit
Enter your choice: 1
1. See the maximum amount of water that can reach a specific city
2. See the maximum amount of water that can reach each city
1
Enter the city code: C_4
The maximum amount of water that can reach Braga is 1208 m^3/s
-------------------------------------------------------
Do you want to continue using the analysis tool?
1. Yes
2. No
Enter your choice: 2
Exiting the program.
```

Highlights

We consider that the most relevant part of our work was the balancing of our network, as it has a good impact in efficiency in the use of the water supplies. It also was what we put the most work into.

Main difficulties and participation

Our main difficulty was basically finding the right solution to solve each individual problem in the most efficient way.

Participation:

- Guilherme Santos: 1/3
- Alexandre Lopes: 1/3
- Tomás Silva: 1/3