Water Supply Analysis System

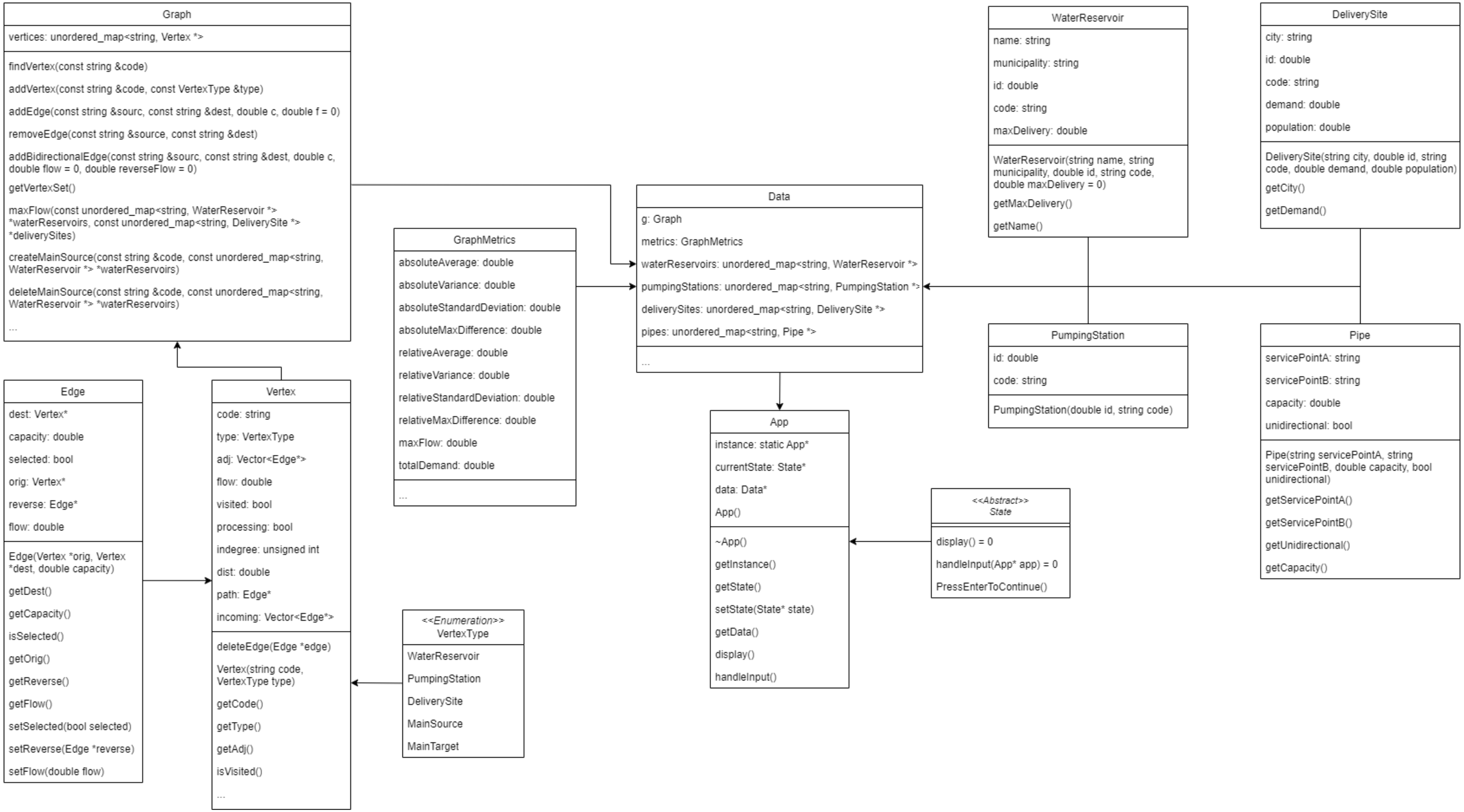
Design of Algorithms (DA) – 2023/24

António Santos – up201705558

Bruno Ferreira – up202207863

Gonçalo Ferros – up202207592

Class Diagram



Project Structure

We opted to implement a state pattern to separate the presentation of content from the underlying algorithms of the different features.

This approach allows for clearer comprehension and modular design. By decoupling the display aspect, it becomes easier to modify or replace the user interface without impacting the core functionality. Additionally, this separation promotes code reusability and enhances maintainability, as modifications to the display do not necessitate changes to the underlying algorithms.

Overall, this design decision improves the overall structure and flexibility of the system.

Dataset reading

The dataset reading is done by the function readFiles() that calls 4 other functions, one for each file (reservoirs, pumping stations, cities and pipes). Files are processed line by line and objects of each type are created and stored in an unordered_map.

```cpp
void Data::readFileStations(ifstream &file) {
    string line;
    getline(file, line);

    while(getline(file, line)) {
        // Remove carriage return characters if present
        line.erase(std::remove(line.begin(), line.end(), '\r'), line.end());

        string code;
        double id;
        stringstream ss(line);
        ss >> id;
        ss.ignore();
        getline(ss, code, ',');

        if(code == "") continue;

        PumpingStation* ps = new PumpingStation(id, code);
        g.addVertex(code, VertexType::PumpingStation);
        this->pumpingStations.insert({code, ps});
    }
}
```

```cpp
unordered_map<string, WaterReservoir *> waterReservoirs;
unordered_map<string, PumpingStation *> pumpingStations;
unordered_map<string, DeliverySite *> deliverySites;
unordered_map<string, Pipe *> pipes;
```

Dataset reading

Special feature: The user simply needs to indicate the path to the files (either absolute or relative path), and everything else is handled by the readFiles function. This improves the user experience, as the user is not required to specify the path to each individual file. Additionally, the user can load a new network whenever desired.

```
void Data::readFiles(const filesystem::path &dir_path) {
    filesystem::path reservoirPath;
    filesystem::path stationsPath;
    filesystem::path citiesPath;
    filesystem::path pipesPath;

    try {
        for (const auto& entry : filesystem::directory_iterator( dir_path)) {

            if (entry.is_regular_file()) {
                string filename = entry.path().filename().string();

                if (filename.find( "Reservoir") != string::npos) {
                    if (!reservoirPath.empty()) throw runtime_error("Error: Multiple Reservoir files found.");

                    reservoirPath = dir_path / filename;

                } else if (filename.find( "Stations") != string::npos) {
                    if (!stationsPath.empty()) throw runtime_error("Error: Multiple Stations files found.");

                    stationsPath = dir_path / filename;

                } else if (filename.find( "Cities") != string::npos) {
                    if (!citiesPath.empty()) throw runtime_error("Error: Multiple Cities files found.");

                    citiesPath = dir_path / filename;

                } else if (filename.find( "Pipes") != string::npos) {
                    if (!pipesPath.empty()) throw runtime_error("Error: Multiple Pipes files found.");

                    pipesPath = dir_path / filename;
                }
            }
        }
```

```
> Loaded Network: Project1DataSetSmall
========== MAIN MENU ==========
   1. Load Network
   2. Find Max Water Flow
   3. Verify Water Supply
   4. Load Optimization
   5. Reservoir Impact
   6. Pumping Station Impact
   7. Pipeline Failure Impact

   q. Exit
------------------------------
Enter your choice: 1
Insert path to the files (Ex: "./dataset/Project1DataSetSmall"): ./dataset/Project1LargeDataSet
```

The Graph

Vertices (Water Reservoirs, Pumping Stations, Delivery Stations):

- Each vertex in the graph represents either a Water Reservoir, Pumping Station, or Delivery Station.
- We have created an enumerator to distinguish between each possible type. For each vertex, we store its type and code.

Edges (Pipelines):

- Each edge represents a pipeline.
- For each edge, we store the capacity and the current water flow passing through it.

```
class Vertex {
private:
    string code;            // code of the node
    VertexType type;        // type of the node
    vector<Edge *> adj;  // outgoing edges

    double flow = 0;

    // auxiliary fields
    bool visited = false; // used by DFS, BFS, Prim ..
    double dist = 0;
    Edge *path = nullptr;

    vector<Edge *> incoming; // incoming edges
```

```
class Edge {
private:
    Vertex *dest; // destination vertex
    double capacity; // edge capacity

    // used for bidirectional edges
    Vertex *orig;
    Edge *reverse = nullptr;

    double flow{}; // for flow-related problems
```

```
class Graph {
private:
    unordered_map<string, Vertex *> vertices;
    string mainSourceCode = "mainSource";
    string mainTargetCode = "mainTarget";
```

Implemented features (Max Water Flow)

- **A Specific City:** O(1)
- **All Cities:** O(n)
- **Verify Water Supply:** O(n)

This functions are this efficient because the Edmonds-Karp algorithm is executed only once (after reading the files).

```
>> All Cities Max Flow:
City                    Code      Demand      Flow Value

Calheta                 C_10      76          76
Ponta do Sol            C_9       59          59
Funchal                 C_6       740         664
Santa Cruz              C_5       295         295
Ribeira Brava           C_8       89          89
Machico                 C_4       137         137
Santana                 C_3       46          46
Câmara de Lobos        C_7       225         225
São Vicente            C_2       34          34
Porto Moniz             C_1       18          18

Max Flow: 1643 m3/s

>> Output file is at: ./output/Project1DataSetSmall/max_flow.csv
---------------------------------------------------
Press ENTER to continue...
```

```
Insert city code (Ex: C_1): C_1
------------------------------------
>> Specific City Max Flow:
City name: Porto Moniz
Code: C_1
Demand: 18 m3/sec
Flow value: 18 m3/sec
------------------------------------
Press ENTER to continue...
```

```
>> Cities lacking desired water rate level:
City                    Code      Deficit Value

Funchal                 C_6       76

Total Demand: 1719 m3/s
Total Water Supplied: 1643 m3/s
The network cannot meet the water needs!

>> Output file is at: ./output/Project1DataSetSmall/verify_water_supply.csv
---------------------------------------------------
Press ENTER to continue...
```

Implemented features (Load Optimization)

The worst-case time complexity is **O(n(E * (V + E))),** where **n** is the number of iterations.

Initially, it computes the graph's initial. It then iteratively performs load optimization until convergence criteria are satisfied or the maximum number of iterations is reached. During each iteration, it sorts edges based on the difference between capacity and flow. For each edge, it identifies paths between its source and destination vertices, selecting the path with the maximum minimum residual capacity and adjusting flow along that path. After each iteration, it updates the final metrics and checks for convergence. This process continues until convergence or until the maximum number of iterations is reached. Finally, it updates the flow values for all vertices in the graph.

```
>> Load Optimization:
(initial metrics / final metrics)

> Absolute:
   Average:            173.41176 / 75.07843
   Max Difference:     750.00000 / 493.00000
   Variance:           59522.97042 / 16333.26774
   Standard deviation: 243.97330 / 127.80167

> Relative:
   Average:            0.40480 / 0.24696
   Max Difference:     1.00000 / 1.00000
   Variance:           0.20193 / 0.11281
   Standard deviation: 0.44936 / 0.33588

> Total Max Flow:      1643 / 1643
----------------------------------------------------
Press ENTER to continue...
```

Implemented features (Reservoir Impact)

- **Not Essential:** O(n * V * (E^2)
- **A Specific City:** O(V * (E^2))
- **All Cities:** O(n * V * (E^2))

Where n is the number of reservoirs.

However, in practice, these functions perform better because they never execute the entire Edmonds-Karp algorithm from the beginning.

```
>> Not Essential Reservoirs:

All reservoirs are essential to
maintain the current max flow!

>> Output file is at: ./output/Project1DataSetSmall/not_essential_reservoirs.csv
----------------------------------------------------
>> All Reservoirs Impact:
Reservoir Code > (City Code, Demand, Old Flow, New Flow)

R_4  >  (C_6, 740, 664, 494)   (C_5, 295, 295, 195)   (C_4, 137, 137, 72)   (C_3, 46, 46, 0)
R_3  >  (C_6, 740, 664, 214)   (C_3, 46, 46, 20)   (C_7, 225, 225, 155)   (C_2, 34, 34, 30)
R_2  >  (C_10, 76, 76, 31)   (C_9, 59, 59, 20)   (C_8, 89, 89, 80)   (C_2, 34, 34, 24)
R_1  >  (C_10, 76, 76, 50)   (C_6, 740, 664, 534)   (C_5, 295, 295, 176)   (C_4, 137, 137, 0)

>> Output file is at: ./output/Project1DataSetSmall/reservoirs_impact.csv
```

```
----------------------------------------------------
>> Water Reservoir out of commission:
Code: R_4
Name: Ribeiro Frio
Max Delivery: 385

> Cities with affected water flow:
City                      Code       Demand     Old Flow    New Flow


Funchal                   C_6        740        664         494
Santa Cruz                C_5        295        295         195
Machico                   C_4        137        137         72
Santana                   C_3        46         46          0
São Vicente              C_2        34         34          30


Total Demand: 1719 m3/s
Max Flow: 1643 m3/s
Total Water Supplied: 1258 m3/s
> Without this reservoir the network cannot meet the water needs!
> This reservoir is essential to maintain the current max flow!
----------------------------------------------------
Press ENTER to continue...
```

Implemented features (Pumping Station Impact)

- **Not Essential:** O(n * V * (E^2)
- **A Specific City:** O(V * (E^2))
- **All Cities:** O(n * V * (E^2))

Where n is the number of pumping stations.

However, in practice, these functions perform better because they never execute the entire Edmonds-Karp algorithm from the beginning.

```
>> Not Essential Pumping Stations:

All pumping stations are essential to
maintain the current max flow!

>> Output file is at: ./output/Project1DataSetSmall/not_essential_stations.csv
----------------------------------------------------
>> All Pumping Stations Impact:
Pumping Station Code > (City Code, Demand, Old Flow, New Flow)

PS_12    >  (C_10, 76, 76, 26)   (C_9, 59, 59, 40)
PS_7     >  (C_5, 295, 295, 61)   (C_4, 137, 137, 0)
PS_6     >  (C_5, 295, 295, 111)   (C_4, 137, 137, 0)   (C_3, 46, 46, 40)
PS_10    >  (C_6, 740, 664, 348)   (C_7, 225, 225, 0)
PS_4     >  (C_6, 740, 664, 549)   (C_5, 295, 295, 204)   (C_4, 137, 137, 2)
PS_11    >  (C_9, 59, 59, 20)   (C_6, 740, 664, 740)   (C_5, 295, 295, 219)
PS_8     >  (C_6, 740, 664, 625)   (C_5, 295, 295, 100)   (C_4, 137, 137, 80)
```

```
>> Pumping Station out of commission:
Code: PS_1

> Cities with affected water flow:
City                      Code       Demand     Old Flow    New Flow

Calheta                   C_10       76         76          50
Funchal                   C_6        740        664         658
Santana                   C_3        46         46          20
São Vicente              C_2        34         34          10
Porto Moniz               C_1        18         18          0

Total Demand: 1719 m3/s
Current Max Flow: 1643 m3/s
Total Water Supplied: 1543 m3/s
> Without this pumping station the network cannot meet the water needs!
> This pumping station is essential to maintain the current max flow!
----------------------------------------------------
Press ENTER to continue...
```

Implemented features (Pipeline Impact)

- **Essential:** O(V * (E^3)
- **A Specific Pipeline:** O(V * (E^2))
- **All Pipelines:** O(V * (E^3))

However, in practice, these functions performs better, because they never execute the whole Edmonds-Karp from the beginning.

```
>> Essential Pipelines for each city:
(City Code, City Name) > (Pipeline Code)

(C_1, Porto Moniz)  >  (PS_1-C_1) (R_1-PS_1)
(C_9, Ponta do Sol)  >  (PS_11-C_9) (PS_12-C_9)
(C_10, Calheta)  >  (PS_1-C_10) (PS_12-C_10) (R_1-PS_1) (R_2-PS_12)
(C_7, Câmara de Lobos)  >  (PS_10-C_7) (PS_11-C_8) (PS_4-PS_10) (R_3-PS_10) (
(C_2, São Vicente)  >  (PS_1-PS_2) (PS_11-C_2) (PS_2-C_2) (PS_2-PS_3) (PS_3-C_
(C_3, Santana)  >  (PS_1-PS_2) (PS_2-PS_3) (PS_5-C_3) (PS_6-C_3) (R_1-PS_2) (
(C_8, Ribeira Brava)  >  (PS_11-C_8) (R_2-PS_11)
(C_4, Machico)  >  (PS_1-C_10) (PS_11-C_9) (PS_2-PS_3) (PS_3-PS_5) (PS_5-PS_6)
(C_5, Santa Cruz)  >  (PS_1-C_10) (PS_11-C_9) (PS_3-C_2) (PS_3-PS_4) (PS_3-PS_
(C_6, Funchal)  >  (PS_1-C_1) (PS_1-C_10) (PS_10-C_6) (PS_10-C_7) (PS_11-C_8)

>> Output file is at: ./output/Project1DataSetSmall/cities_not_essential_pipe
---------------------------------------------------
```

```
>> Pipeline Impact:
Code: PS_1-PS_2
Capacity: 400
Unidirectional

> Cities with affected water flow:
City                     Code       Demand      Old Flow    New Flow

Funchal                  C_6        740         664         658
Santana                  C_3        46          46          20
São Vicente             C_2        34          34          10

Total Demand: 1719 m3/s
Current Max Flow: 1643 m3/s
Total Water Supplied: 1587 m3/s
> Without this pipeline the network cannot meet the water needs!
> This pipeline is essential to maintain the current max flow!
----------------------------------------------------
Press ENTER to continue...
```

Implemented features (Output Files)

The previous functions all output the resulting data into a file. These files are saved in the **"./output/<name_of_network>"** directory.

This setup enables the user to load different networks during program execution without constantly overwriting output files. Consequently, this enhances usability.

User Interface

The user interface was designed to be intuitive and simple. We used numbered options and colors to make navigation easier for the user.

```
==== PUMPING STATION IMPACT ====
  1. Not Essential
  2. Specific Pumping Station
  3. All Pumping Stations

  q. Main Menu
-------------------------------
```

```
> Loaded Network: Project1DataSetSmall
========== MAIN MENU ==========
   1. Load Network
   2. Find Max Water Flow
   3. Verify Water Supply
   4. Load Optimization
   5. Reservoir Impact
   6. Pumping Station Impact
   7. Pipeline Failure Impact

   q. Exit
------------------------------
Enter your choice: 1
Insert path to the files (Ex: "./dataset/Project1DataSetSmall"): ./dataset/Project1LargeDataSet
```

Highlighted features

- **Network Resiliency Testing:** All algorithms demonstrate high efficiency.
- **Load Optimization:** The algorithm significantly enhances metrics in an efficient manner.
- **File Management:** The process of reading network files and saving output files is straightforward yet robust, ensuring a user-friendly operation.

Difficulties and team effort

The most significant challenge was certainly developing the **load optimization** function, given its potential for various approaches. The difficulty was not in finding a solution to the problem, but rather in identifying an efficient one. **This proved to be the primary challenge.**

Additionally, implementing the **resilience** function presented some challenges, as it occasionally entered into an infinite loop due to the initial lack of checks for flow cycles.

The workload was evenly distributed among all team members, with each contributing equally.