Sistema de Gestão de Abastecimento de Água

António Santos (up202205469)
Vanessa Quirós (up202207919)
Leonardo Garcia (up202200041)

Classes Utilizadas

WaterManager

- Contém todos os dados necessários para fazer todas as operações.
- Verifica e processa os inputs do utilizador.
- Classe principal.

Application

- Gere as várias opções do menu e chama as funções respetivas do WaterManager.
- Recebe os inputs do utilizador

WR

- Representa um Water Reservoir

PS

- Representa uma Pumping Station

Graph

- Representa os dados dos ficheiros .csv numa só estrutura de dados

WaterElement

- Representa um WR, PS ou DS

DS

- Representa um Delivery Site

Leitura de Dados

- Leitura feita através dos métodos:
  - parseData() -> processReservoirs(), processPumps(), processCities(), processPipes()

Tipo	Nome	Dados extraídos a partir de
Graph<WaterElement*>	waterNetwork	Reservoirs.csv, Stations.csv, Cities.csv, Pipes.csv
unordered_map	waterReservoirMap	Reservoirs.csv
unordered_map	waterPumpMap	Stations.csv
unordered_map	waterCityMap	Cities.csv

Leitura de Dados

Estruturas de Dados

- Graph<T>

Método	Complexidade
findVertex()	O(V)
addBidirectionalEdge()	O(1)
getVertexSet()	O(1)

- unordered_map

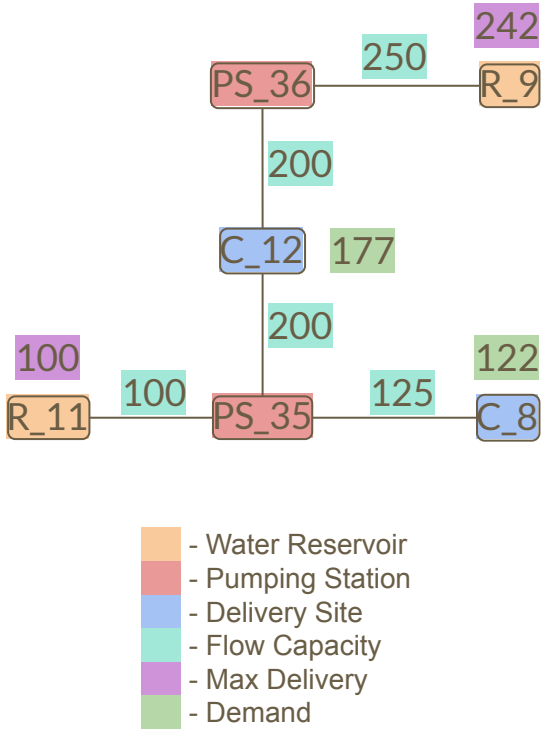
Método	Complexidade
at()	O(1)
size()	O(1)

Grafo Utilizado

`Graph<WaterElement*>`

Atributos utilizados no algoritmo de Edmonds-Karp:

- Vertex:
  - visited
  - path
  - incoming
- Edge:
  - flow
  - weight
- Graph:
  - vertexSet

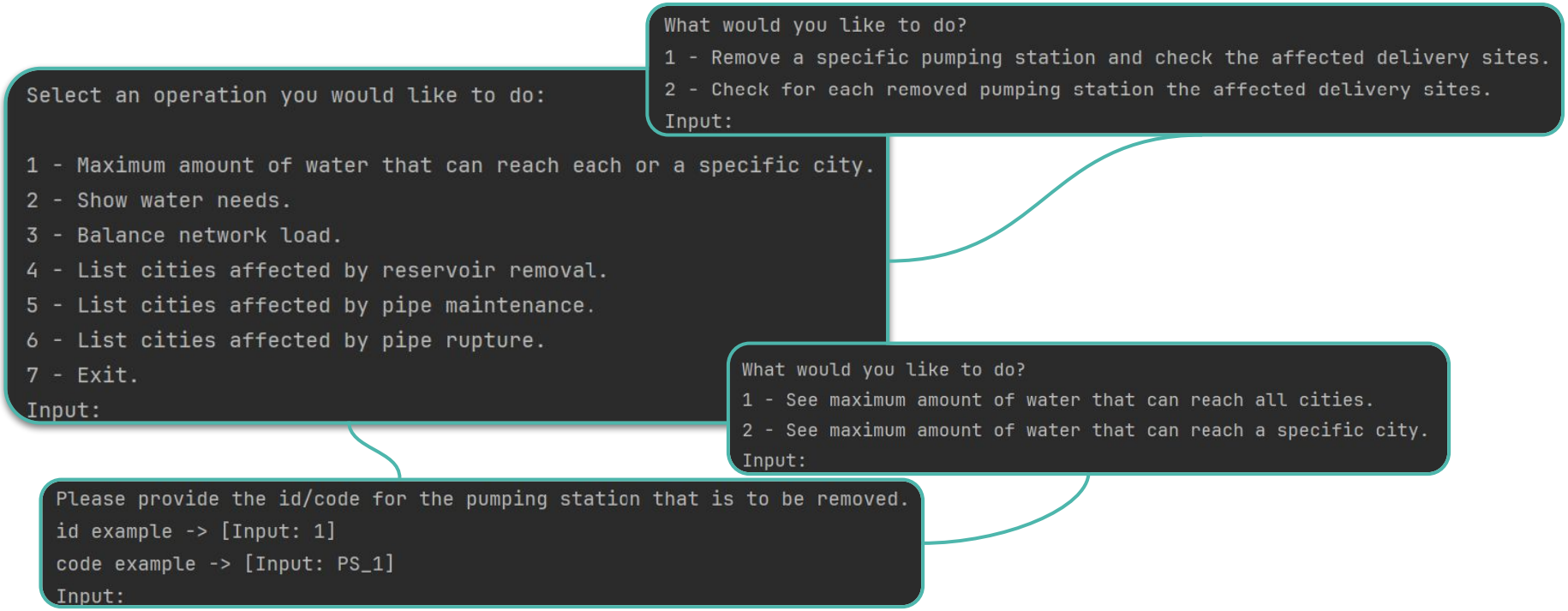


Grafo Utilizado

- `dynamic_cast` para aceder aos vários tipos de vértices
- Utilizado para aceder a atributos que são específicos a uma subclasse de WaterElement

```
auto* isDS = dynamic_cast<DS*>(vertex->getInfo());
if (isDS){
    isDS->setCurrentFlow( flow: 0.0);
}
```

Menu Interativo



Funcionalidades Implementadas

[T2.1] - Maximum flow

[string maximumFlowAllCities() and string maximumFlowSpecificCity(string &city)]

- Determina o maior flow capaz de atingir todas as cidades do grafo.
- *Edmonds-Karp* com adição de uma Super Source e uma Super Sink,
- Uso de uma variável currentFlow.

Time Complexity - > $O(VE^2)$.

Funcionalidades Implementadas

[T2.2] - List water needs

- Lista as cidades que estão a necessitar de mais água, ao checar o flow que está a chegar nestas.

Time Complexity: $O(VE^2)$

Faz uso da função maximumFlowAllCities().

```
void WaterManager::listWaterNeeds() {
    maximumFlowAllCities();  // Run Edmonds-Karp algorithm

    for (auto ds : pair<const string, DS *> : waterCityMap) { // Only check flow for delivery sites
        int receivedFlow = 0;
        WaterElement *ds_to_we = ds.second;
        Vertex<WaterElement *> *v = waterNetwork.findVertex( in: ds_to_we);
        if (v == nullptr) {
            std::cout << "An error has occurred...\n";
            return;
        }
        for (auto e : Edge<WaterElement *> * : v->getIncoming()) receivedFlow += e->getFlow();
        int demand = ds.second->getDemand();
        if (receivedFlow < demand) {
            std::cout << ds.second->getCity() << " (" << ds.first << ") needs more water:"
                      << "\n- Demand: " << demand
                      << "\n- Actual flow: " << receivedFlow
                      << "\n- Deficit: " << demand - receivedFlow << "\n\n";
        }
    }
}
```

Funcionalidades Implementadas

Métricas Usadas:
1. Média das diferenças
2. Máxima diferença
3. Variância das diferenças

[T2.3] - Balancing algorithms

[void balancingAlgorithmNeighborDistribution]

- Balanceamento do sistema ao redistribuir capacidade de pipes entre a sua vizinhança.

[void balancingAlgorithmSortingDistribution]

- Balanceamento do sistema ao redistribuir a capacidade das pipes que tem maior diferença de capacidade e flow para as que têm as menores.

[void balancingAlgorithmAverageDistribution]

- Balanceamento de capacidade das pipelines ao definir todas as capacidades como a média inicial das capacidades da pipes.

Time Complexity: $O(VE^2)$, dado que usa o maximumFlowAllCities().

Funcionalidades Implementadas

[T3.1] - Cities affected by reservoir removal

[void listCitiesAffectedByReservoirRemoval()]

Time Complexity: $O(VE^2)$, dado que usa o maximumFlowAllCities().

[T3.2] - Cities affected by pump station removal

[string citiesAffectedByMaintenance_SpecificPump(string idCode) e string citiesAffectedByMaintenace_AllPumps]

Time Complexity: $O(VE^2)$

Funcionalidades Implementadas

[T3.3] - Cities affected by pipeline removal

[citiesAffectedByPipeRupture(string &cityCode) e citiesAffectedByPipeRupture()]

Determina as cidades afetadas pela remoção de uma pipeline do grafo. Implementada de duas formas:

1 - Recebe o código da cidade como input e procura para quais rupturas de pipelines aquela cidade seria afetada.

2 - Calcula para cada pipeline do sistema as consequências de sua remoção (cidades que seriam afetada)

Time Complexity: $O(VE^3)$

Funcionalidades a Destacar

- Utilização de unordered_map para acesso com menor complexidade aos reservoirs, pumping stations e cities.

```
std::unordered_map<std::string, WR*> waterReservoirMap;

std::unordered_map<std::string, PS*> waterPumpMap;

std::unordered_map<std::string, DS*> waterCityMap;
```

- Output dos resultados obtidos no exercício T2.1 para um ficheiro com data e hora registada.

```
void Application::outputToFile(std::string header, std::string text) {
    std::ofstream out( s: "../outputFiles/functions.txt",  mode: std::ios::app);

    // Get current time
    std::time_t currentTime = std::time(nullptr);

    // Convert the current time to a struct tm
    std::tm *localTime = std::localtime(&currentTime);

    out << std::put_time( tm: localTime,  fmt: "%Y-%m-%d %H:%M:%S") << " -> "
        << header << "\n\n" << text << "\n\n";

    out.close();
}
```

Dificuldades Encontradas

- Exercício 2.1 e 2.3
- Ambiguidades na descrição do projeto
- Outputs diferentes em sistemas operativos diferentes

Esforço de cada elemento:

António Santos: 100%

Vanessa Queirós: 100%

Leonardo Garcia: 100%