U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Conceção e Análise de Algoritmos

À Procura de Estacionamento

2MIEIC02 – Grupo D

(19 de maio de 2017)

Bárbara Silva **up201505628**@fe.up.pt

João Azevedo **up201503256**@fe.up.pt

Julieta Frade **up201506530**@fe.up.pt

Índice

Introdução

Descrição do Tema – "À procura de estacionamento"

À semelhança do primeiro projeto, realizado no âmbito da unidade curricular de Conceção e Análise de Algoritmos, também este se destina ao processamento de um mapa real e cálculo do caminho ideal de um local origem para um parque de estacionamento. Contudo, nesta nova versão procuramos analisar de forma exata ou aproximada todo o input fornecido, em *string*, pelo utilizador, de modo a encontrar a origem e o destino que este nos fornece. Cada local está identificado por uma freguesia e rua, assim, estes dados terão que ser disponibilizados.

O objetivo final permanece o mesmo, que é encontrar o parque de estacionamento mais perto do destino escolhido, dentro das limitações estabelecidas pelo utilizador. A maior diferença encontra-se na analise do input.

Assim, identificamos como objetivos neste segundo projeto, para além dos do primeiro, as seguintes funcionalidades:

- Possibilidade de o utilizador introduzir o nome da freguesia e da rua, de forma exata ou aproximada às que deseja, e encontrar os locais.
- Encontrar um parque de estacionamento adequado às suas preferências.

Para tal, a procura da rua desejada, dentro da freguesia especificada, será realizada de duas formas possíveis: usando pesquisa exata e aproximada.

Identificação e Formalização do Problema

À semelhança do projeto anterior, procurámos interpretar o enunciado proposto e estruturar o mesmo, partindo os problemas em subproblemas até que se tornassem elementares.

Assim, identificamos e modulámos o nosso projeto em alguns tópicos:

- A escolha de uma nova área real de teste.
- A extração de informação dos ficheiros de texto, modificação de alguns e mesmo criação de novos.
- A reestruturação de algum código proveniente do primeiro projeto.
- A determinação do nome da rua e freguesia, usando algoritmos de pesquisa em *strings*.
- A criação de um novo menu que servirá de interface com o automobilista.

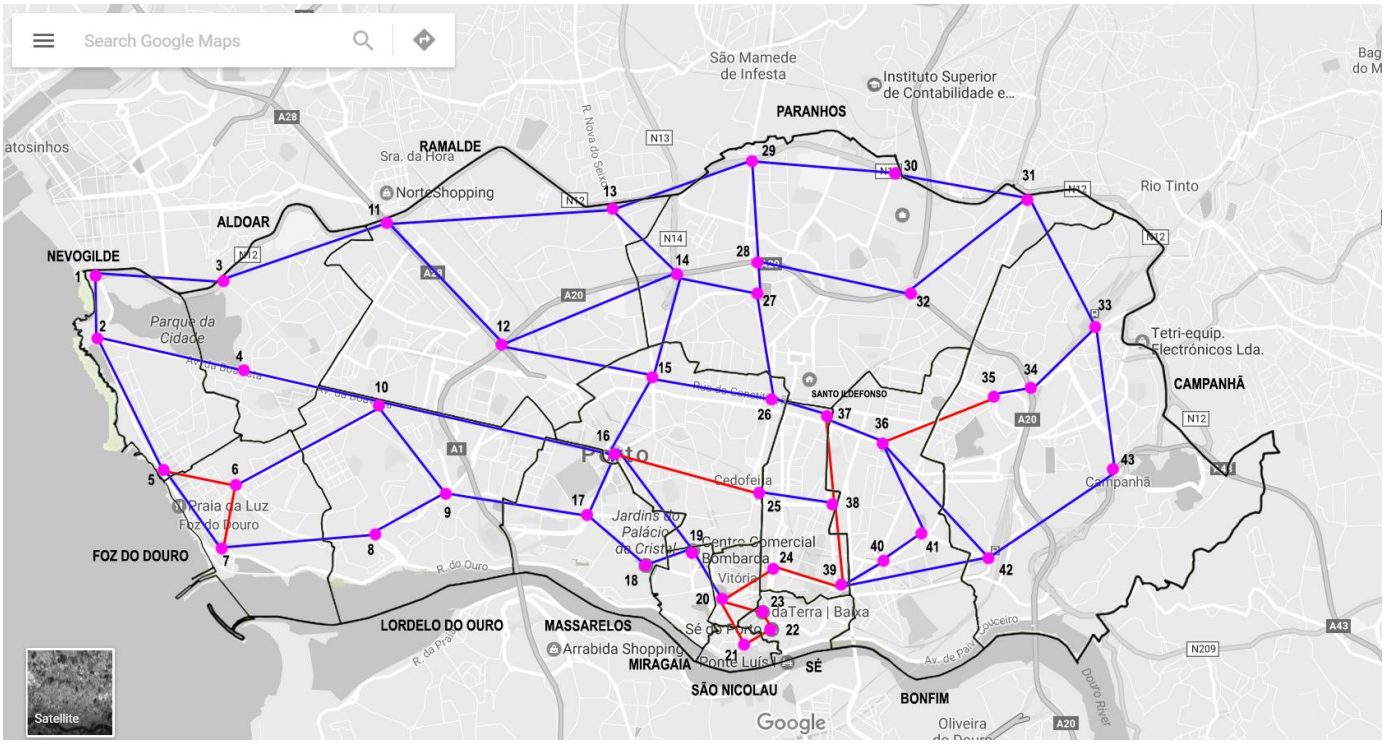
Solução Implementada

Após atualiação dos ficheiros de texto e das funções necessárias à leitura dos mesmos, estaríamos prontos a implementar os algoritmos e a nova interface. Assim sendo, a solução implementada é semelhante à já usada.

Cada nó, para além dos atributos e métodos já implementados (atributos necessários para o funcionamento do algoritmo Djisktra, ID, nome do local e coordenadas), possui agora o nome da freguesia (TOWN) a que pertence.

Assim, com o propósito de auxiliar na pesquisa por frequesia vimos como necessecidade implementar a classe Town. A mesma conteria atributos referentes à identificação da freguesia e uma coleção de ruas que pertencem à mesma.

Mapa e Legenda



Ligações a vermelho: um sentido.

Ligações a azul: dois sentidos.

Freguesia	Nós
Aldoar	3, 4
Bonfim	36, 40, 41
Campanhã	33, 34, 35, 42, 43
Cedofeita	15, 16, 25, 26
Foz do Douro	6, 7
Lordelo do Ouro	8, 9, 10
Massarelos	17, 18
Miragaia	19
Nevogilde	1, 2, 5
Paranhos	14, 27, 28, 29, 30, 31, 32
Ramalde	11, 12, 13
Santo Ildefonso	24, 37, 38, 39
São Nicolau	21
Sé	22, 23
Vitória	20

Tipo de Local	*Nós*
Parque de Estacionamento	1, 4, 7, 9, 15, 21, 27, 29, 33, 37, 41
Posto de Abastecimento	3, 12, 14, 17, 32, 38,
Universidade	8, 30, 42
Centro Comercial	11, 19, 34
Restaurante	6, 10, 23
Cinema	16, 39

1 - 2: Via do Castelo do Queijo

2 - 5: Avenida Montevideu

5 - 6: Rua Corte Real

6 - 7: Rua da Cerca

5 - 7: Avenida do Brasil

7 - 8 - 9: Rua de Diogo Botelho

6 - 10: Avenida do Marechal Gomes da Costa

2 - 4 - 10 - 16: Avenida da Boavista

9 - 10: Rua de Serralves

9 - 17: Rua do Campo Alegre

11 - 12: A28

14 - 15 – 16 -17: Rua Antero Quental

16 - 17: Rua Gonçalo Sampaio

17 - 18: Rua Júlio Dinis

18 - 19: Rua Dom Manuel II

19 - 20: Rua Clemente Meneres

20 - 21: Rua da Vitória

21 - 22: Rua das Flores

22 - 23: Rua de Souto

20 - 23: Rua dos Caldeireiros

14 - 27: Rua de Monsanto

13 - 14: Rua Nova do Seixo

1 - 3 - 11 - 13 - 29 - 30 - 31 - 33 - 43: Estrada da Circunvalação

12 - 15 - 26: Rua da Constituição

16 - 25: Rua da Boavista

16 - 19: Rua de Cedofeita

12 - 14: A20

28 - 29: Rua do Amial

26 - 27 - 28: Rua de Vale Formoso

28 - 32: Rua Manuel Pereira da Silva

31 - 32: Avenida Fernão Magalhães

33 - 34: Rua das Linhas de Torres

34 - 35: Alameda das Antas

35 - 36: Rua Nova de São Crispim

36 – 37 - 26: Rua de Latino Coelho

37 - 38 - 39: Rua de Santa Catarina

25 - 38: Rua de Gonçalo Cristóvão

24 - 39: Rua de Passos Manuel

20 - 24: Rua dos Clérigos

39 - 42: Rua do Heroísmo

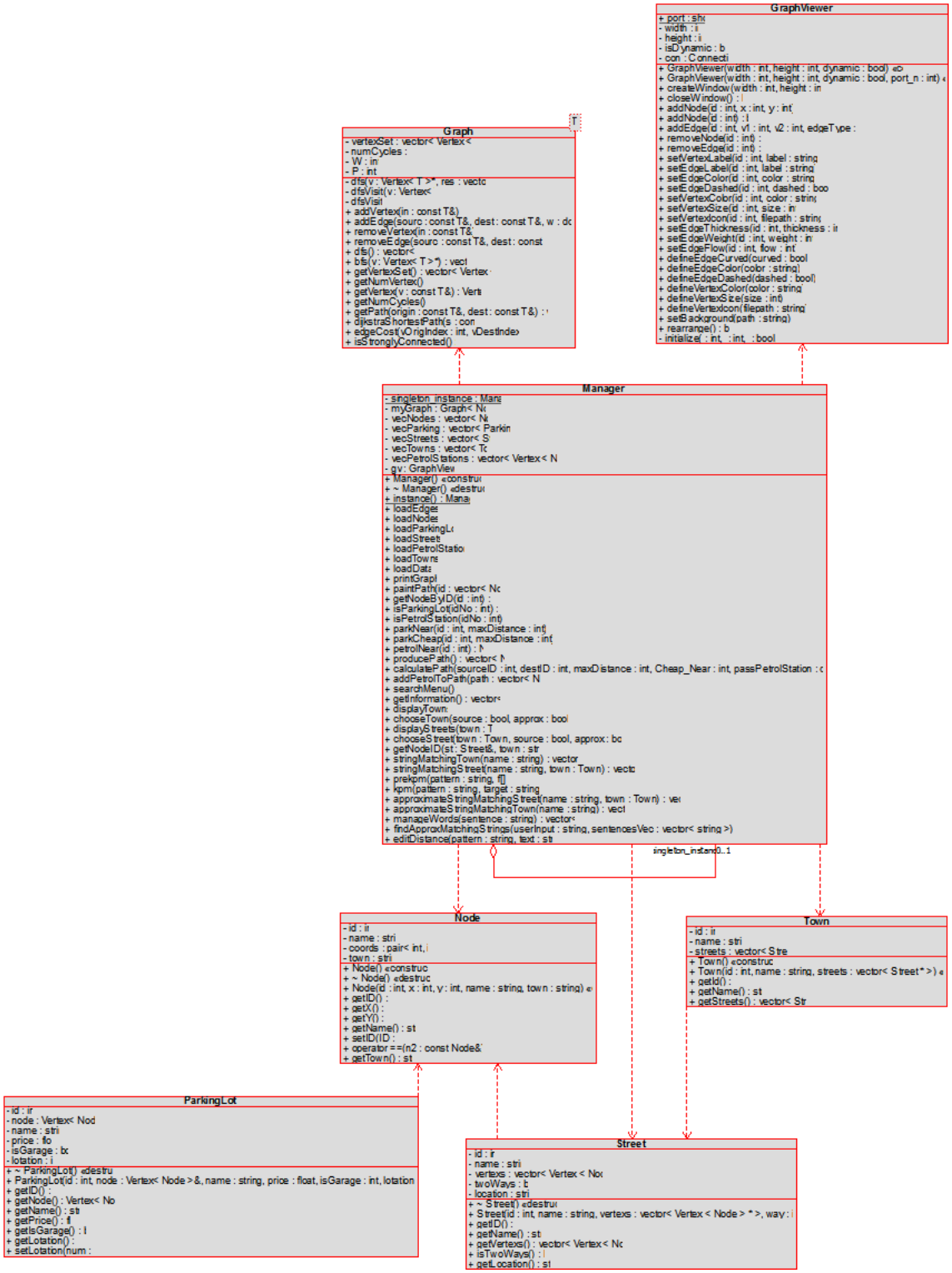
42 - 43: Rua de Bonjóia

39 - 40 - 41: Rua do Bonfim

36 - 41: Rua de Barros Lima

36 - 42: Rua de Justino Teixeira

Diagrama de Classes



Casos de Utilização

Classes

Manager → *Singleton class* (instância da mesma limitada a um objeto) que contem a informação do programa em execução assim como os métodos para o iniciar.

Node → classe representativa de um local (nó), contendo um ID, o nome desse local e um par de coordenadas no mapa.

Street → classe representativa de uma estrada (várias arestas), contendo um ID, o nome da rua, um booleano indicando se é ou não de dois sentidos e uma lista de nós que a estrada liga.

ParkingLot → classe representativa do parque de estacionamento, contendo um ID, o nome do mesmo, o preço a pagar, um booleano indicando se é ou não garagem (caso não o seja é um parquímetro), um nó ao qual o parque está associado e a lotação do mesmo.

Town → possui como atributos um ID, o nome da freguesia e uma coleção de ruas que pertencem à mesma. Agrupa assim todas as ruas na freguesia a procurar.

Ficheiros

Edges.txt – ficheiro contendo a informação relativa a arestas como o seu ID e os nós que liga.

Nodes.txt – ficheiro contendo o ID do nó, as coordenadas x e y no mapa e o nome identificativo do local.

Parking.txt – ficheiro contendo o ID de cada parque, o ID do nó a que está associado, o nome do mesmo, o preço a pagar por lá estacionar, um valor que indica se é garagem ou não e a lotação do mesmo.

Streets.txt – ficheiro contendo a informação das ruas (arestas), como o seu ID, o nome da mesma.

Towns.txt – ficheiro contendo o ID de cada freguesia assim como o seu nome e os ID's das ruas que lhe pertencem.

Pesquisa Exata

Relativamente à procura das ruas e das freguesias através dos seus nomes foi necessária a implementação de algoritmos de pesquisa em *strings*.

Assim, para a pesquisa exata, em que o utilizador escreve exatamente a mesma *string* ou parte desta, caracter a caracter, foi implementado o algoritmo de *Knuth-Morris-Pratt*. Este baseia-se no pré-processamento do padrão (*string* a procurar) para a geração de um autómato finito, porém não o chega a gerar explicitamente. De seguida, efetua um processamento do texto em que será realizada a procura da *string* desejada.

Em termos de complexidade temporal este algoritmo aprente O(|T|+|P|), ou seja, O(|P|) na realização do pré-processamento do autómato e depois O(|T|) a processar o texto.

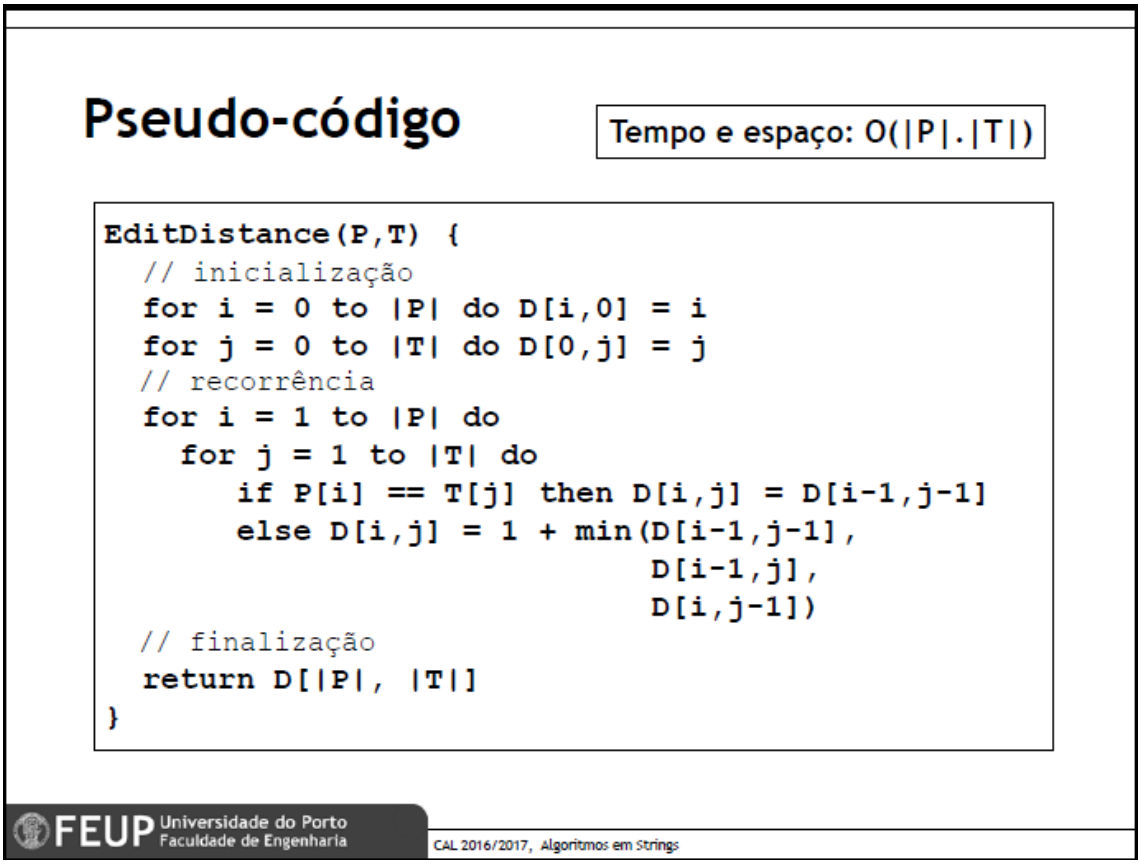
```
algorithm kmp_search:
    input:
        an array of characters, S (the text to be searched)
        an array of characters, W (the word sought)
    output:
        an integer (the zero-based position in S at which W is found or -1 when W not found in S)

    define variables:
        an integer, m ← 0 (the beginning of the current match in S)
        an integer, i ← 0 (the position of the current character in W)
        an array of integers, T (the table, computed elsewhere)

    while m + i < length(S) do
        if W[i] = S[m + i] then
            let i ← i + 1
            if i = length(W) then
                return m (or store/use occurrence)
                let m ← m + i - T[i], i ← T[i] (preparing for search next occurrence, T[length(W)]
can't be -1)
        else
            if T[i] > -1 then
                let m ← m + i - T[i], i ← T[i]
            else
                let m ← m + i + 1, i ← 0

    (if we reach here, we have searched all of S unsuccessfully)
    return -1
```

Pesquisa Aproximada

Relativamente à pesquisa aproximada, o algoritmo baseia-se no cálculo de uma "distância" de edição da *string* a procurar às *strings* comparadas. No fundo, calcula-se o número de operações de modificação (inserção, remoção e substituição) para transformar a *string* encontrada com a procurada. Usualmente é então feito usando programação dinâmica.

Inicializando-se uma matrix e preenchendo-a do topo esquerdo para o fundo direito. Cada "salto" horizontal na matrix ou vertical corresponde a uma inserção ou remoção, respetivamente. Por norma, o custo é de normalmente uma unidade por operação. Um salto diagonal custa uma unidade caso os 2 caracteres na coluna e linha não sejam iguais ou zero, caso sejam.

Cada célula da matrix minimiza assim localmente o número de modificações, ou seja, a distância entre as *strings*. Assim, o número no fundo direito da matrix é a distância entre *strings*.



Para pesquisar *strings* de forma aproximada, tanto nas freguesias como nas ruas, decidimos não comparar diretamente a *string* de input do utilizador com as *strings* do programa. Portanto dividimos cada uma nas diferentes palavras que a compunham e comparamos assim palavra a palavra. A função que faz isto chama-se **findApproxMatchingString**, que recebe a *string* de input e um vetor de *strings* onde pesquisar. Segue-se o *pseudocódigo*:

```
findApproxMatchingString(string userInput, vector<string> sentencesVec){

    vector<string> userInputVec=palavras de userInput
    map <string, int> mapWord //map com as ruas e a distancia minima das palavras da rua a userInputVec[i]
    vector<map<string,int>> mapVecs //vetor de mapWords correspondentes a cada palavra de userInput

    for i = 0 to userInputVec:
        for j=0 to sentencesVec:
            vector <string>sentencesInWordsVec = palavras de sentencesVec[j]
            difference= -1 //distancia minima de userInputVec[i] a cada palavra dessa rua
            for k=0 to sentencesInWordsVec:
                differenceTemp=editDistance(userInputVec[i], sentencesInWordsVec[k]) //algoritmo de pesquisa aproximada
                if(difference == -1 || differenceTemp<difference)
                    difference = differenceTemp
            mapWord.insert(sentencesVec[j],difference)
    mapVecs.push_back(mapWord)

    multimap<int,string> finalMultiMap;

    for i=0 to sentencesVec:
        difference =0 //soma das distancias minimas daquela rua a todas as palavras de input
        for j=0 to mapVecs:
            difference += mapVecs[j][sentencesVec[i]]
        finalMultiMap.insert(difference, sentencesVec[i])

    vector<string> finalVec;

    for i=0 to finalMultiMap
        if(i->first <= 4* userInputVec.size()) finalVec.push_back(i->second) //se a distancia total for menor que 4 vezes o numero de palavras de input

return finalVec;
}
```

Programa

Primeiramente, o programa executa o processamento da informação dos ficheiros de texto, respetiva criação do grafo e visualização do mesmo com toda a informação identificadora de cada aresta e nó. Numa segunda fase, é requerida informação ao utilizador através da consola.

Para começar, é perguntado ao automobilista que tipo de pesquisa pretende que seja utilizada para analise do seu input.

```
> TYPES OF SEARCH:
[1] Exact
[2] Approximate

> Type your choice:
```

De seguida, são disponibilizadas todas as freguesias conhecidas ao programa, tendo o utilizador que escrever uma delas. Vamos exemplificar usando pesquisa exata.

```
> TOWNS

            Aldoar              Bonfim           Campanha
          Cedofeita       Foz do Douro    Lordelo do Ouro
         Massarelos           Miragaia          Nevogilde
           Paranhos            Ramalde    Santo Ildefonso
        Sao Nicolau                 Se           Vitoria

> Where are you? Type the town's name: Ald

> TOWN(S) FOUND.

> WE FOUND THESE TOWNS:

ID - TOWN
-------------
1 : Aldoar

> Type the town's ID: 1
```

Como podemos confirmar, após o input da freguesia ter sido analisado, foram retornadas as escolhas possíveis. O mesmo se irá verificar agora para a escolha da rua.

```
 > STREETS OF: Aldoar

Estrada da Circunvalacao
Avenida da Boavista

> Where are you? Type the street's name: Cir

> STREET(S) FOUND.

> WE FOUND THESE STREETS:

ID - STREET
-------------
21 : Estrada da Circunvalacao

> Type the streets's ID: 21

> Current location valid.
```

Após a validação do local origem, seguimos para a escolha do destino, usando exatamente o mesmo método.

```
> TOWNS

            Aldoar              Bonfim           Campanha
          Cedofeita       Foz do Douro    Lordelo do Ouro
         Massarelos           Miragaia          Nevogilde
           Paranhos            Ramalde    Santo Ildefonso
        Sao Nicolau                 Se           Vitoria

> Where do you want to go? Type the town's name: ampan

> TOWN(S) FOUND.

> WE FOUND THESE TOWNS:

ID - TOWN
-------------
3 : Campanha

> Type the town's ID: 3
```

```
 > STREETS OF: Campanha

Estrada da Circunvalacao
Rua Nova de Sao Crispim
Alameda das Antas
Rua das Linhas de Torres
Rua de Justino Teixeira

> Where do you want to go? Type the street's name: Alame

> STREET(S) FOUND.

> WE FOUND THESE STREETS:

ID - STREET
-------------
30 : Alameda das Antas

> Type the streets's ID: 30

> This street has more than one node in this town.
> Choose more precisely.
 ID      Place
-----------------
34  :  mall
35  :  null

> Insert the ID:
```

Caso a mesma rua, dentro da mesma freguesia, tenha mais que um nó conhecido, é pedido ao utilizador que escolha o tipo de local onde se encontra.

Por fim, a distância máxima do local destino ao parque de estacionamento deve ser indicada, assim como a especificação do caminho ideal, podendo este ser o mais curto ou o que conduz ao parque mais barato dentro da distância máxima especificada ao destino.

Como parâmetro adicional, o automobilista pode, para ambas as opções anteriormente especificadas, optar por um trajeto que inclua um posto de abastecimento ou não. A imagem seguinte ilustra então o resultado final, indicando-se o ID do parque ideal e o caminho a percorrer, nó a nó, e, se pretendido, o posto de abastecimento automóvel onde parar.

```
> Destination location valid.

> MAX DISTANCE (m): 300

> WHICH TYPE OF PARKING LOT DO YOU PREFER ?
[1]Cheap           [2]Near Your Destiny

> Option: 1

> DO YOU WANT TO FILL UP THE CAR ? (y/n): y

> PARK: 1
> GAS STATION: 3
> PATH: 3 1 3 11 12 15 26 37 36 35 34

> EXECUTION TIME (ms): 1
> IS GRAPH STRONGLY CONNECTED: yes
```

No final da execução do programa é sempre indicado o tempo de execução e se o grafo usado era fortemente conexo ou não.

De forma análoga, o fluxo do programa optando por pesquisa aproximada é semelhante diferindo apenas em alguns passos, a seguir demonstrados.

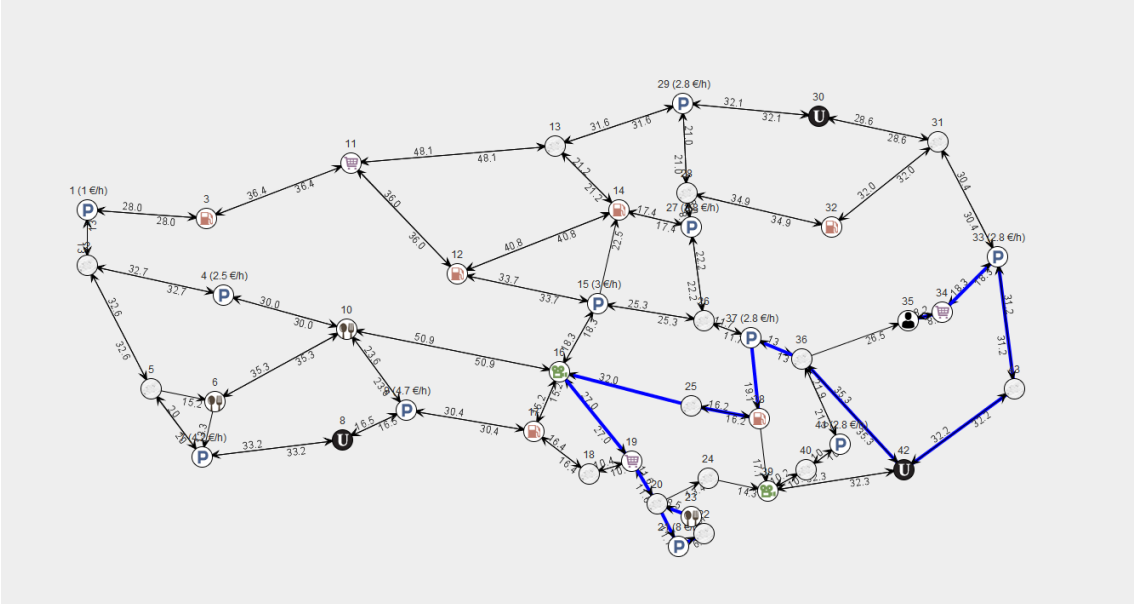
```
> TYPES OF SEARCH:
[1] Exact
[2] Approximate

> Type your choice: 2

 > TOWNS

            Aldoar              Bonfim           Campanha
          Cedofeita        Foz do Douro     Lordelo do Ouro
         Massarelos            Miragaia          Nevogilde
           Paranhos             Ramalde     Santo Ildefonso
        Sao Nicolau                  Se            Vitoria

> Where are you? Type the town's name: Capmahna

> APPROXIMATE TOWN(S) FOUND.

> WE FOUND THESE TOWNS:

ID - TOWN
-------------
3 : Campanha

> Type the town's ID: 3
```

Evidentemente, o programa inicia-se de igual forma e deve-se optar pela pesquisa aproximada.

Porém, usando agora aproximada, não necessitamos de escrever nem a freguesia nem o nome da rua exatamente igual (caracter a caracter) ao nome das mesmas.

```
 > STREETS OF: Campanha

Estrada da Circunvalacao
Rua Nova de Sao Crispim
Alameda das Antas
Rua das Linhas de Torres
Rua de Justino Teixeira

> Where are you? Type the street's name: Cispim

> APPROXIMATE STREET(S) FOUND.

> WE FOUND THESE STREETS:

ID - STREET
-------------
31 : Rua Nova de Sao Crispim

> Type the streets's ID: 31

> Current location valid.
```

A introdução do local de destino é feita de igual forma. No fim, é novamente mostrado o tempo de execução e se o grafo usado é fortemente conexo ou não, visualizando-se após um gráfico semelhante ao da figura:



Dificuldades

Na realização do presente projeto não surgiram muitas dificuldades uma vez que se tratava de uma modificação ao anterior. Contudo, a elaboração de um novo mapa e conceção de todos os dados fundamentais ao programa, terá sido um pouco trabalhoso. Após ultrapassar este obstáculo, o principal foco de estudo foi perceber e aplicar os algoritmos de pesquisa exata e aproximada em *strings* para a pesquisa das ruas. A parte mais trabalhosa foi chegar a uma conclusão em relação à análise de *strings* em pesquisa aproximada para não aplicarmos diretamente o algoritmo.

Em suma, com a realização deste segundo projeto fomos capazes de mais uma vez trabalhar de forma eficaz enquanto grupo de trabalho, distribuindo as tarefas de forma rápida para realizar o necessário à implementação da solução para o problema pedido.

Distribuição do Trabalho

Todos os membros do grupo de esforçaram igualmente na estratificação do problema e compreensão do enunciado. Quanto à implementação da solução, houve algumas discrepâncias, estando abaixo explicadas com devidas percentagens.

Bárbara Silva

- Estruturação do código.
- Criação de um novo mapa e respetivos ficheiros de texto.
- Extração de informação de ficheiros.
- Implementação do menu e verificação do input.
- Estruturação e revisão do relatório.
- Implementação de parte dos algoritmos.
- **Percentagem:** 40%

João Azevedo

- Reestruturação do ficheiro dos parques e respetiva extração.
- Redação do relatório.
- Pesquisa sobre os algoritmos a utilizar.
- **Percentagem:** 20%

Julieta Frade

- Implementação do *Graph Viewer*.
- Implementação das funções que calculam as diferentes opções de caminho.
- Implementação do método de cálculo da conectividade do grafo.
- Implementação de parte dos algoritmos.
- **Percentagem:** 40%

Conclusão

A proposta de trabalho continha um intuito educativo, sendo requerido da nossa parte que compreendêssemos e usássemos não só novas estruturas como grafos, mas também algoritmos de pesquisa nos mesmos. Não só, mas também o desenvolvimento da componente de trabalho em grupo.

Concluímos, portanto, que os objetivos pretendidos com este projeto de grupo foram atingidos, quer a nível individual quer a nível coletivo, uma vez que cada elemento domina agora os temas lecionados na unidade curricular e é capaz de os aplicar numa componente prática.