

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

João Azevedo **up201503256**@fe.up.pt

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

Conceção e Análise de Algoritmos

*À Procura de Estacionamento*

*2MIEIC02 – Grupo D*

*(19 de maio de 2017)*

Índice

|  |  |
| --- | --- |
| **Introdução** | 2 |
| Descrição do Tema | 2 |
| Identificação e Formalização do Problema | 3 |
| **Solução Implementada** | 4 |
| **Diagrama de Classes** | 5 |
| **Casos de Utilização** | 6 |
| Classes | 6 |
| Ficheiros | 6 |
| Algoritmos | 7 |
| Pesquisa Exata | 7 |
| Pesquisa Aproximada | 8 |
| Programa | 10 |
| **Dificuldades** | 14 |
| **Distribuição do Trabalho** | 15 |
| **Conclusão** | 16 |
|  |  |

***Introdução***

*Descrição do Tema – “À procura de estacionamento”*

À semelhança do primeiro projecto realizado no âmbito da unidade curricular de Conceção e Análise de Algoritmos também este se destina ao processamento de um mapa reale cálculo do caminho ideal de um local origem para um parque de estacionamento. Contudo pretende-se procurar um parque que se encontre no distrito indicado e cujo nome seja ou igual ou próximo do indicado pelo utilizador.

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

* Possibilidade de o utilizador procurar o parque destino com base no distrito pretendido e indicando o nome da rua onde o mesmo se deve situar.

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

(nos falamos com a apr e ele concordou em não fazermos assim. Basicamente não procuramos o parque, fazemos o mesmo que o outro projeto mas procurando o no dizendo a freguesia e a rua)

*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 e a modificação de alguns ou mesmo criação de novos
* A reestruturação de algum código proveniente do primeiro projeto.
* A determinação do noma da rua e freguesia, usando algoritmos de pesquisa em strings.
* A criação de um novo menu que servirá de interface com o automobilista.

Desde cedo que demos a devida importância à compreensão do funcionamento e estruturação dos grafos bem como os diferentes algoritmos de pesquisa nos mesmos.

***Solução Implementada***

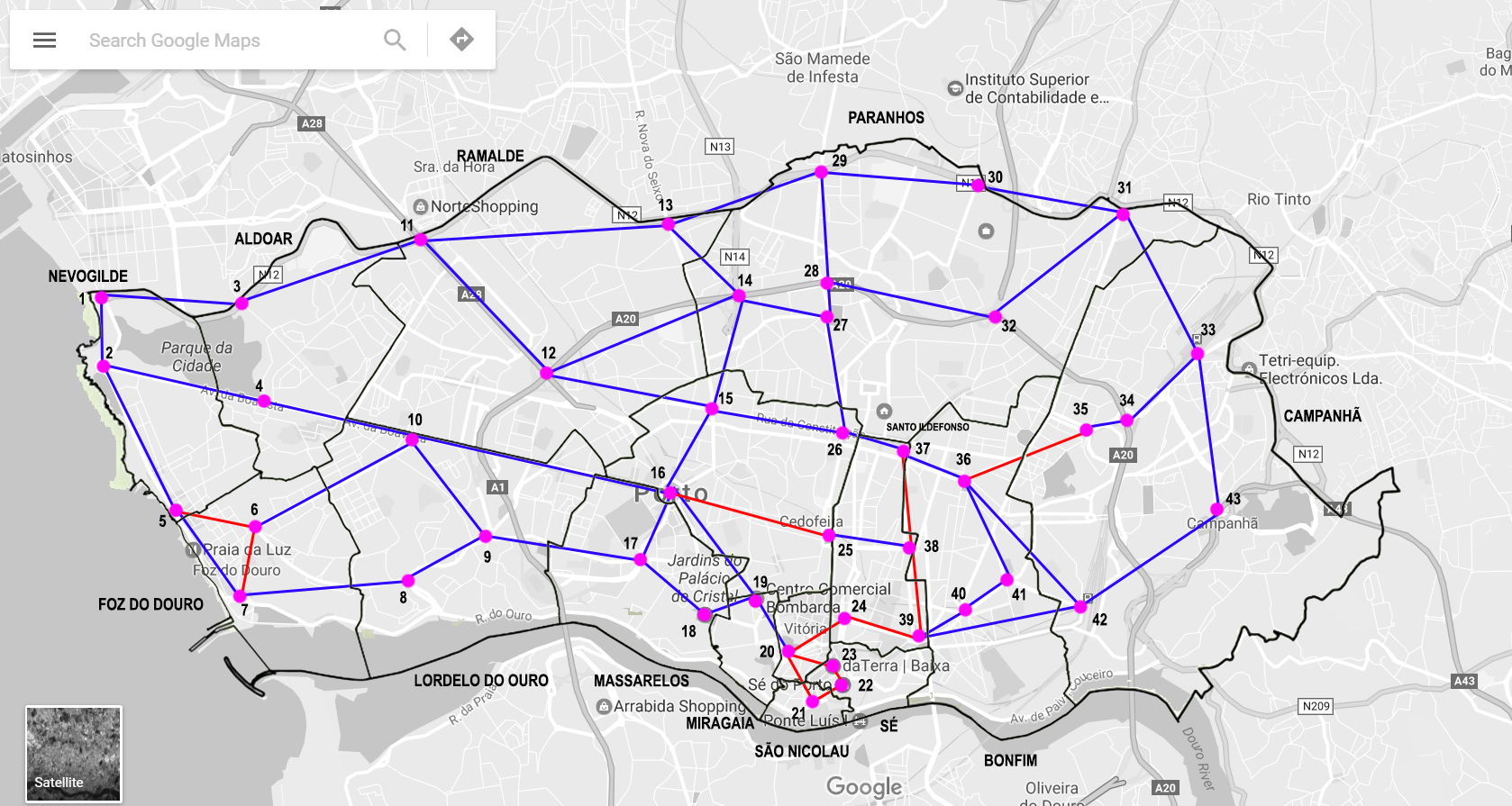
Uma vez mais e à semelhança do último trabalho, optámos por não usar o parser providenciado.

Efetivamente, após atualiação dos ficheiros de texto e das funções necessárias à leitura dos mesmos, assim como do código de algumas estaríamos prontos a implementar os algoritmos.

Assim sendo, a solução implementada é extremamente semelhante à já usada, acrescentando porém que cada parque possui agora uma lotação.

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

(falar da classe town maybe)

*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 - 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

**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

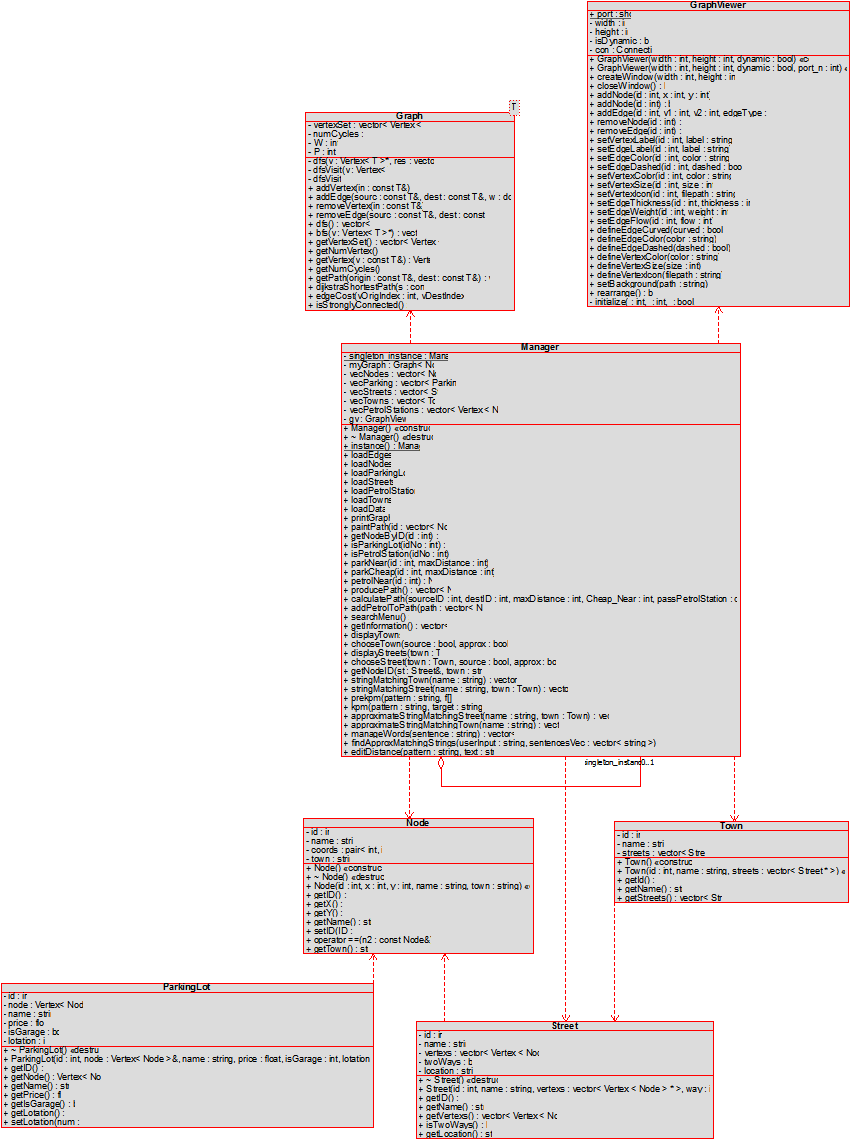
**22 - 23**: Rua de Souto

**20 - 23**: Rua dos Caldeireiros

**14 - 27**: Rua de Monsanto

**13 - 14**: Rua Nova do Seixo

***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.

*Algoritmos*

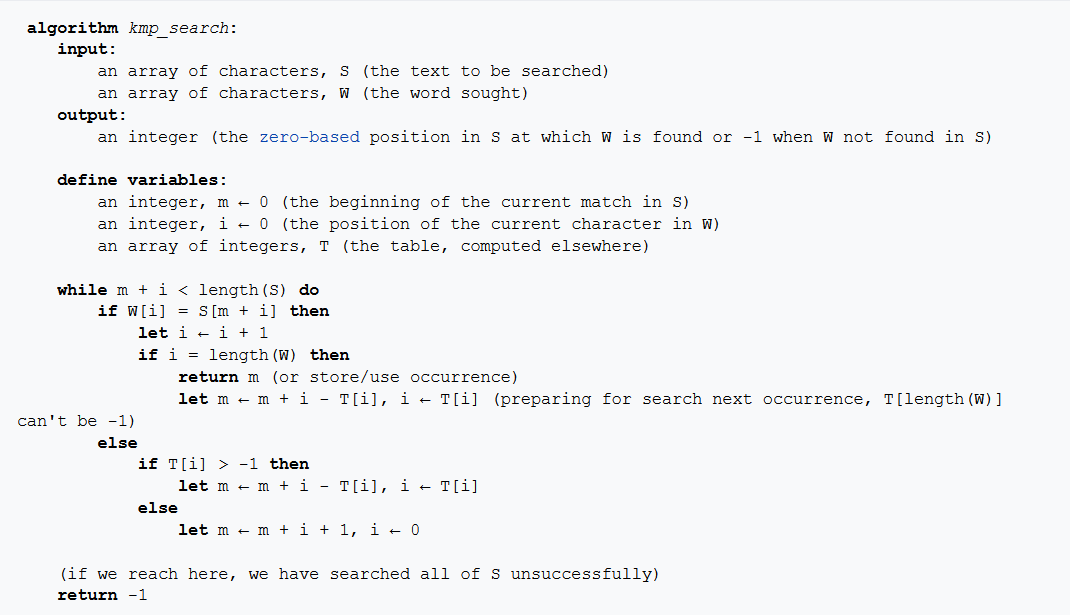
***Pesquisa Exata***

Para a 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 algoritmo 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.

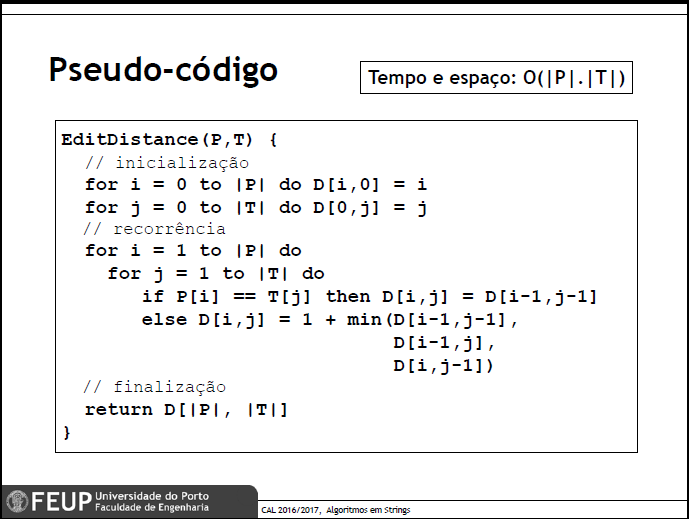


***Pesquisa Aproximada***

Para a pesquisa aproximada, o algoritmo baseia-se no cálculo de uma “distância” 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. (TODO JU falar da divisão das strings)

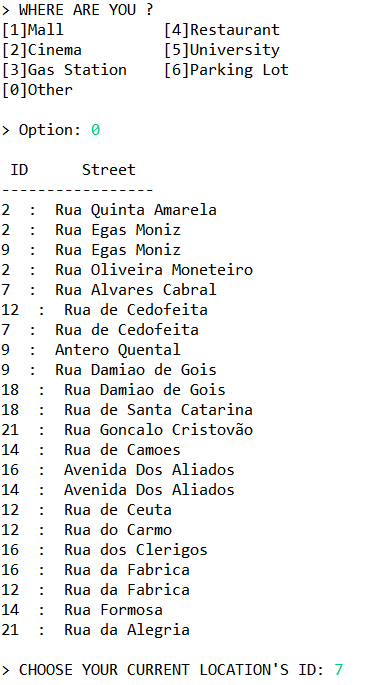


*Programa*

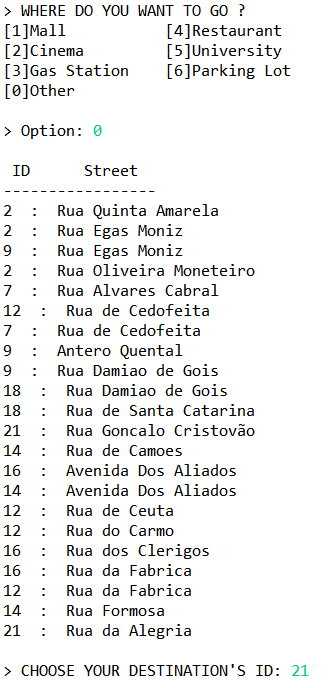
Primeiramente, o programa executa o processamento da informação dos ficheiros de texto, respetiva criação do grafo e visualização do mesmo com toda a informação identificadora de cada aresta e nó.

Numa segunda fase, é requerida informação ao automobilista através da consola.

Para começar, é requerida a posição atual do utilizador, como abaixo indicado:



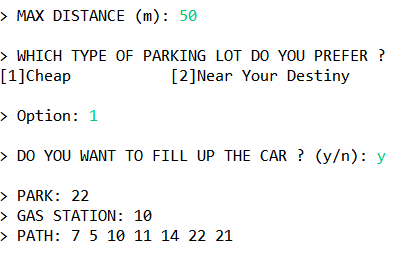
De seguida é necessário, tal como abaixo demonstrado, indicar o destino pretendido:



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.



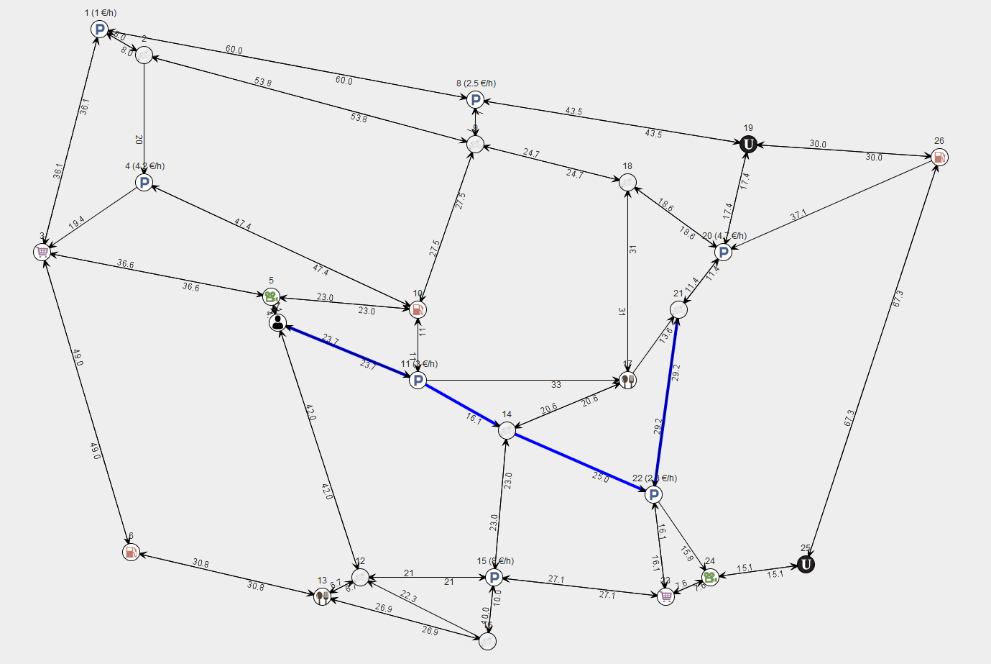
No final da execução do programa é sempre indicado o tempo de execução e se o grafo usado era fortemente conexo ou não:



De seguida, apresentámos alguns dos resultados visualizados mediante cada um dos quatro casos possíveis.

*Partida 🡪 Destino mais barato*

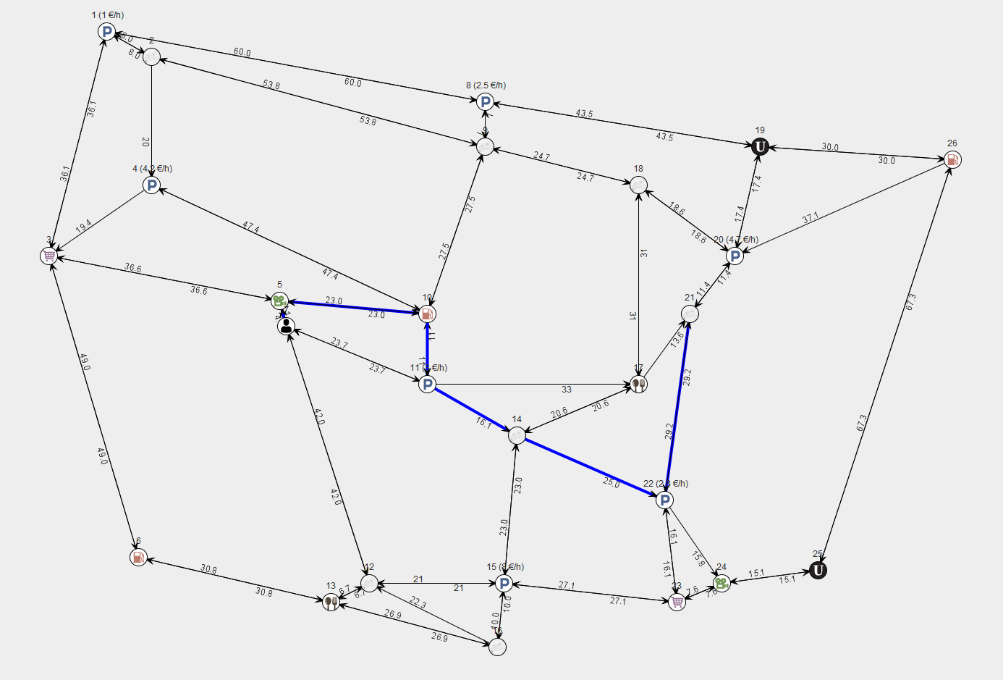
Caso se pretenda estacionar no parque mais barato perto do destino e não se faça questão de abastecer, como por exemplo, fazendo este mesmo percurso entre os nós 7 e 21, o resultado visualizado será o seguinte:



É de notar que o caminho ideal calculado se encontra delineado a azul.

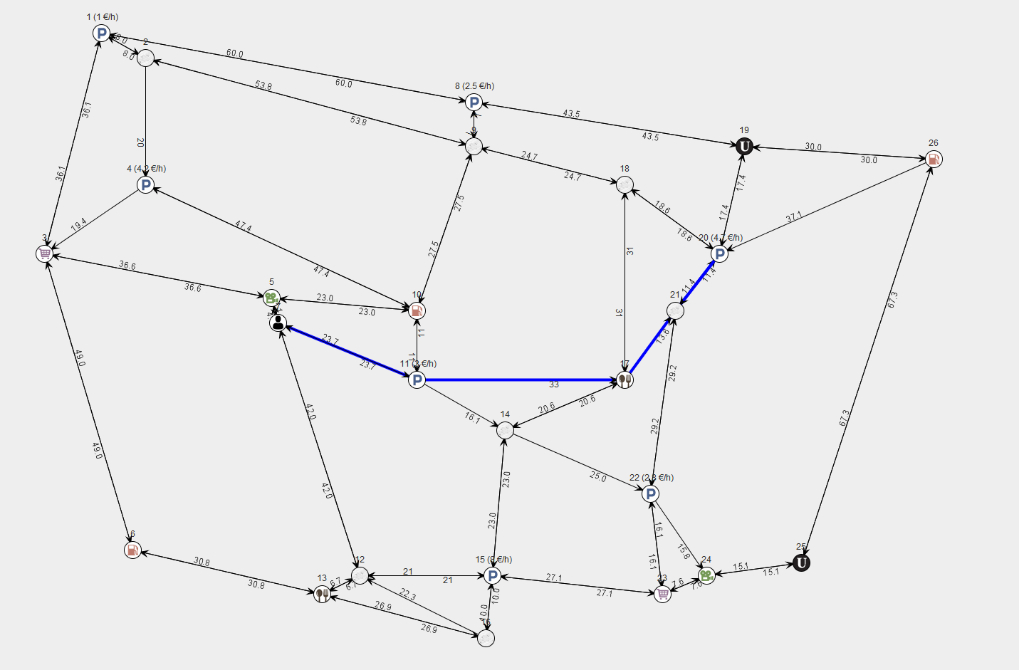
*Partida 🡪 Posto Abastecimento 🡪 Destino mais barato*

Ainda trabalhando sobre os nós 7 e 21, podemos visualizar o seguinte resultado quando pretendido o caminho que leve ao parque mais barato perto do destino, passando pelo posto de abastecimento.



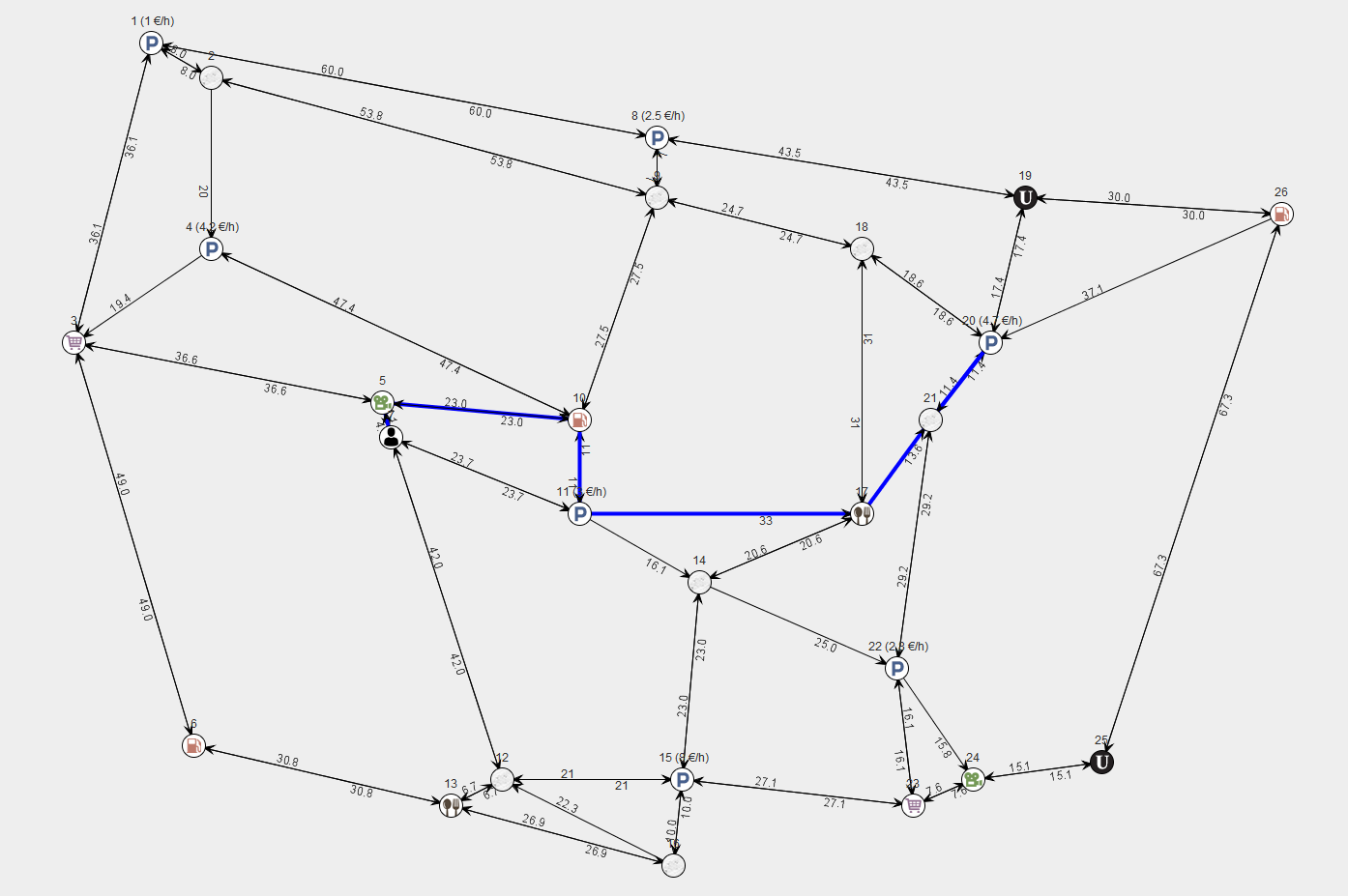
*Partida 🡪 Destino pelo caminho mais curto*

Abaixo é apresentado o resultado de optar por não querer abastecer o veículo e pretender colocar o parque o mais perto possível do local destino.



*Partida 🡪 Posto Abastecimento 🡪 Destino pelo caminho mais curto*

Por fim, caso o automobilista pretenda o caminho mais curto do local em que se encontra ao destino, passando por um posto automóvel, referindo-nos ainda como exemplo aos nós 7 e 21, obtemos o seguinte grafo:



***Dificuldades***

Na realização do presente projeto não surgiram muitas dificuldades uma vez que se tratava de uma simples modificação ao anterior. Contudo, o principal foco de estudo foi perceber e aplicar os algoritmos de pesquisa exata e aproximada em strings para a pesquisa das ruas.

(TODO JU: falar da dificuldade em fazer a função)

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 que fosse necessário.

***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.
* Extração de informação de ficheiros.
* Implementação do menu e verificação do input.
* Criação do novo mapa.
* Estruturação e revisão do relatório.
* Implementação de parte dos algoritmos.
* **Percentagem:** 30%

**João Azevedo**

* Reestruturação dos ficheiros .txt
* Reestruturação da extração de informação dos ficheiros .txt
* Redação do relatório.
* Pesquisa sobre os algoritmos a utilizar.
* **Percentagem:** 30%

**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.