

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*

*(7 de abril 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 |
| *Djikstra* | 7 |
| Conectividade | 9 |
| Programa | 10 |
| **Dificuldades** | 15 |
| **Distribuição do Trabalho** | 16 |
| **Conclusão** | 17 |
|  |  |

***Introdução***

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

O projeto realizado no âmbito da unidade curricular de Conceção e Análise de Algoritmos destina-se ao processamento de um mapa real, neste caso uma área por nós escolhida *à priori,* e respetivo cálculo do caminho ideal de um local, origem, para um destino, ambos indicados pelo automobilista.

No contexto do problema, o “caminho ideal” assume diferentes semânticas, uma vez que, por exemplo, o caminho pretendido pode ser o que nos leva ao destino pelo caminho mais curto ou que nos leva ao parque de estacionamento mais barato, dentro de uma distância máxima ao nosso destino.

Assim, identificamos como objetivos as seguintes funcionalidades do projeto:

* Possibilidade de o utilizador definir o local de origem e destino.
* Possibilidade de escolher o caminho:
  + Mais curto até ao parque associado ao destino.
  + Que leva ao parque mais barato associado ao destino.
* Possibilidade de escolher qualquer uma das opções acima desejando abastecer o veículo ou não.

Todos os parques por nós indicados ao utilizador estarão a uma distância máxima do local de destino, distância essa definida pelo utilizador.

*Identificação e Formalização do Problema*

Enquanto grupo de trabalho procurámos desde cedo ler e compreender o enunciado proposto e formalizar o problema, estruturando-o e repartindo problemas até que se tornassem elementares e de simples resolução.

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

* A escolha de uma área real de teste.
* O uso do *parser* providenciado.
* A extração de informação dos ficheiros de texto.
* A criação da estrutura de dados a partir dos mesmos ficheiros:
* Criação de nós que contivessem a informação pretendida.
* Criação de arestas com os valores essenciais aos cálculos futuros.
* Criação conjunta do grafo, englobando nós e arestas.
* A determinação do caminho ideal no grafo usando o algoritmo adequado.
* A criação de um menu que serviria 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***

O primeiro passo foi então definir a área a testar e se para tal usaríamos o *parser* disponibilizado ou iriamos nós próprios criar os ficheiros contendo a informação necessária. Rapidamente nos apercebemos que seria mais prático e elucidativo criarmos os nossos próprios ficheiros de teste.

Efetivamente, tendo os ficheiros prontos e uma ideia bem definida da área a utilizar, facilmente, criámos as funções necessárias à leitura de ficheiros de texto, criando também os nós e arestas do nosso grafo.

Cada nó possui atributos necessários ao funcionamento dos algoritmos à frente explicados, como se já foi visitado pelo mesmo. Em cada nó está também armazenada a lista de arestas a que o vértice (nó) está ligado. Por último, a informação útil em si, contida num objeto (**Node**) por nós definido contendo o seu ID, nome indicativo do local (cinema, shopping, universidade...), e um par de coordenadas no mapa (x, y).

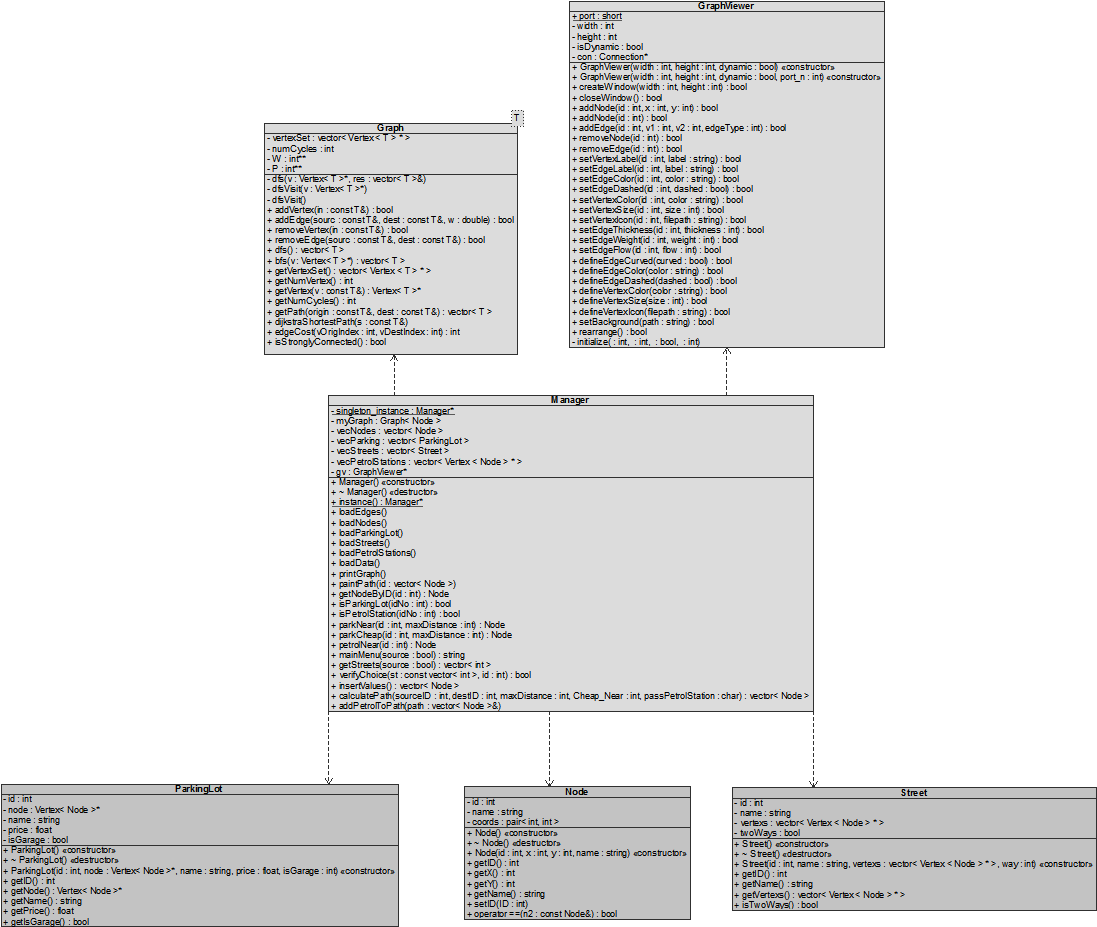
Os parques são objetos que contêm os nós a eles associados, isto é, um parque está associado a diferentes destinos (próximos do mesmo). Para além disso, guardam o seu nome e preço e ainda a informação de se é um parquímetro ou garagem.

As arestas são as ruas em si e contêm não só um ID e o nome da mesma, mas também a informação sobre a existência de dois sentidos ou apenas um e uma lista de nós que liga.

Para o cálculo do caminho mais curto, uma vez que se trata de um grafo pesado com apenas números inteiros positivos, foi usado o algoritmo de ***Dijkstra*** sempre que necessário.

INSERIR MAPA

***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 (aresta), 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) e uma lista de nós ao qual o parque se associa, ou seja, nós do qual o parque é considerado perto o suficiente para ser uma boa opção.

*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 e um valor que indica se é garagem ou não.

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

*Algoritmos*

***Djikstra***

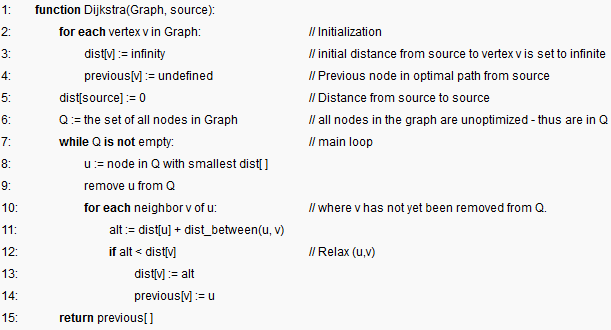
Para o cálculo do trajeto mais curto, sempre que necessário, tratando-se de um grafo pesado com valores apenas positivos, usamos o algoritmo de ***Djikstra***.

Em termos de analise de complexidade espacial e temporal este algoritmo apresenta:

* Espacialmente: O (N^2 ) onde N representa o número de nós.
* Temporalmente: O (A + N log(N) ) onde N representa o número de nós e A o de arestas.

Este algoritmo revela-se como o mais eficiente dos lecionados, impondo apenas a restrição dos valores das arestas serem positivos.

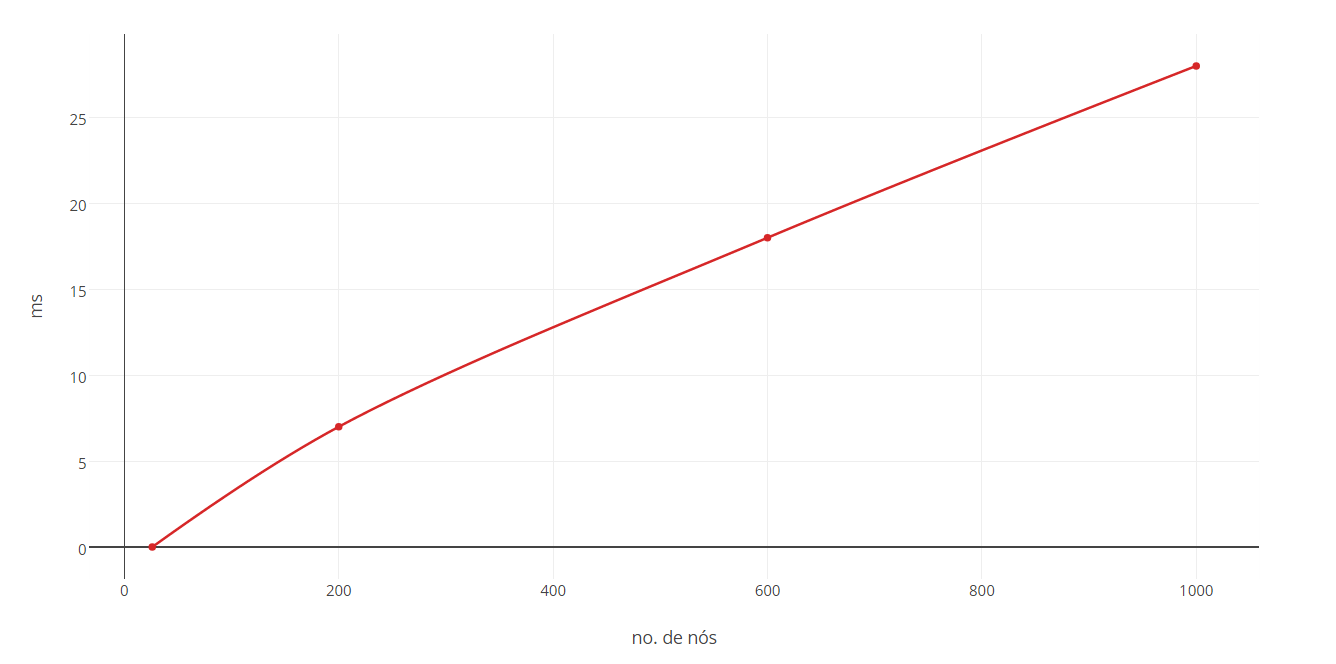
Em baixo apresenta-se o pseudocódigo do algoritmo, sendo que elucida melhor o funcionamento do mesmo:



Como já referido, o algoritmo de ***Dijkstra*** apresenta complexidade temporal de O(n\*log(n) + e), sendo “n” o numero de nós e “e” o número de arestas. Assim, o gráfico que mostra o tempo de execução do algoritmo em função do número de vértices e arestas seria dado por uma função aproximadamente como:

Tal gráfico é corroborado pela análise e medida dos tempos de execução do nosso programa usando diferentes dados de entrada, como os apresentados na tabela a seguir:

|  |  |  |
| --- | --- | --- |
| Número de vértices (N) | Número de arestas (E) | Tempo (ms) |
| 26 | 43 | < 1 |
| 200 | 100 | 7 |
| 600 | 300 | 18 |
| 1000 | 500 | 28 |



***Conectividade***

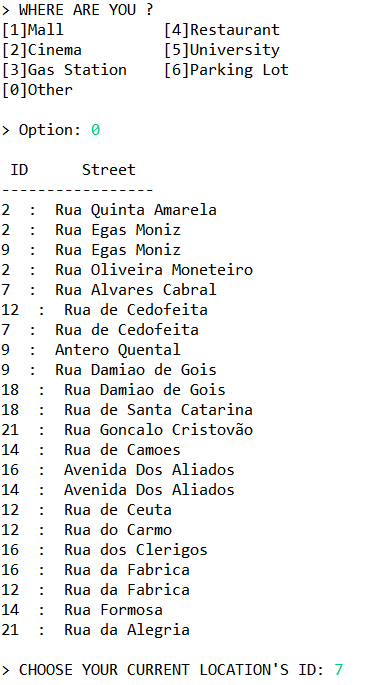
Algoritmo da conetividade

*Programa*

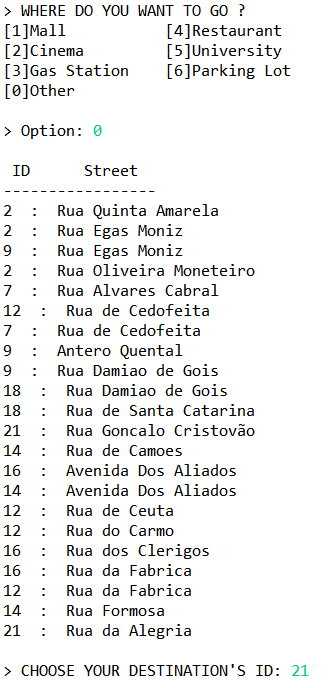
Primeiramente, o programa executa o processamento da informação dos ficheiros de texto, respetiva criação do grafo e visualização do mesmo com toda a informação identificadora de cada aresta e nó.

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

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



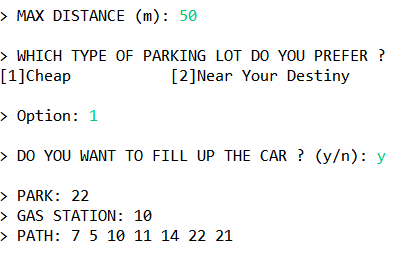
De seguida é necessário, tal como abaixo demonstrado, indicar o destino pretendido:



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.

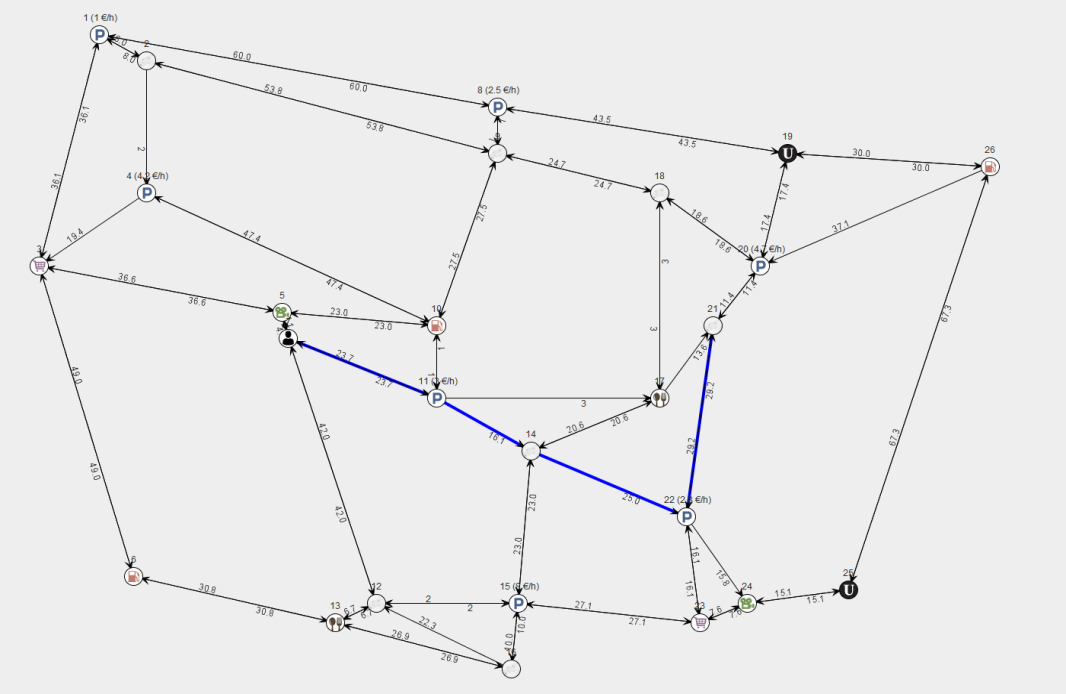
A imagem seguinte ilustra então o resultado final, indicando-se o ID do parque ideal e o caminho a percorrer, nó a nó, e, se pretendido, o posto de abastecimento automóvel onde parar.



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

*Partida 🡪 Destino mais barato*

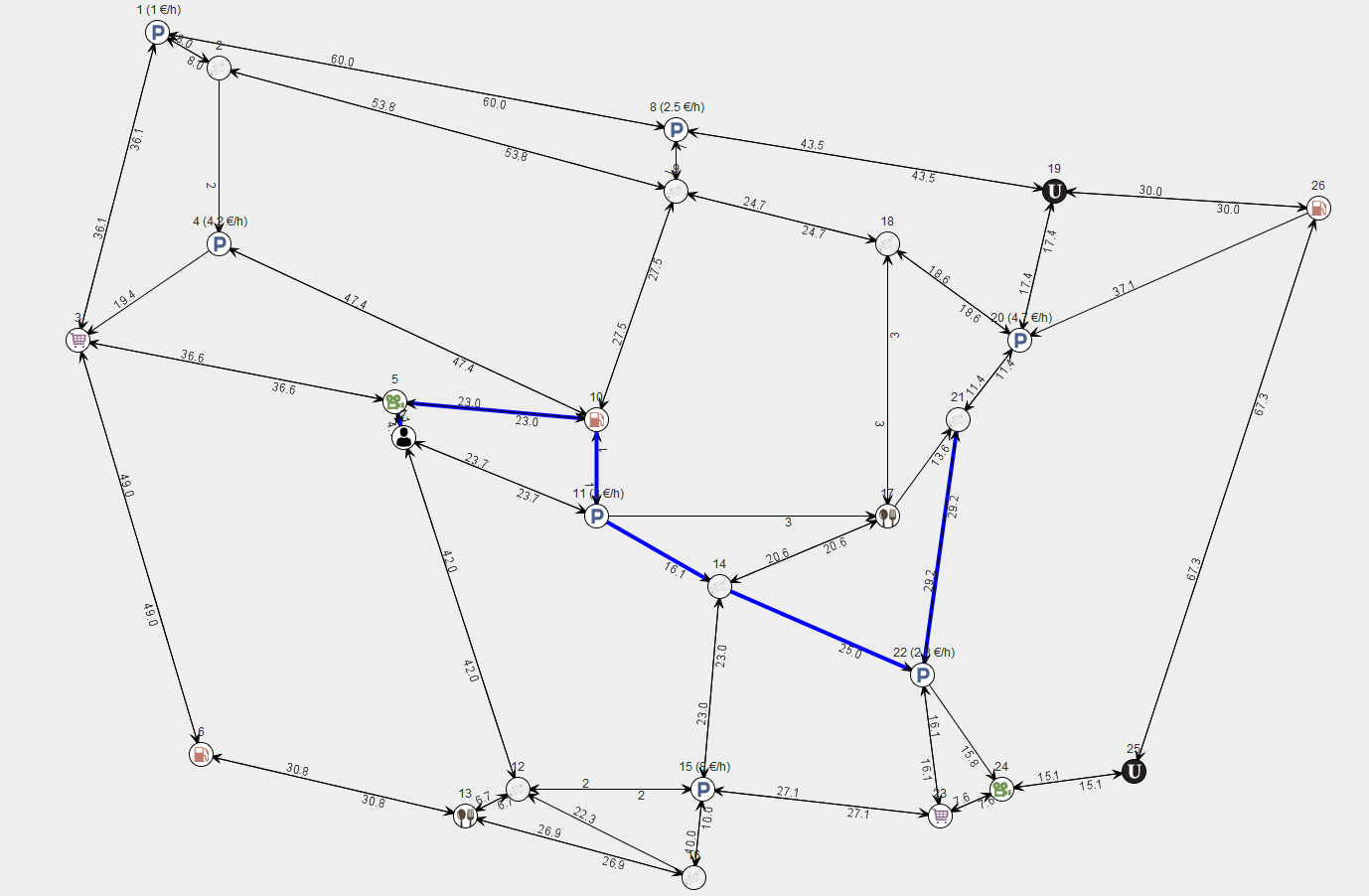
Caso se pretenda estacionar no parque mais barato perto do destino e não se faça questão de abastecer, como por exemplo, fazendo este mesmo percurso entre os nós 7 e 21, o resultado visualizado será o seguinte:



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

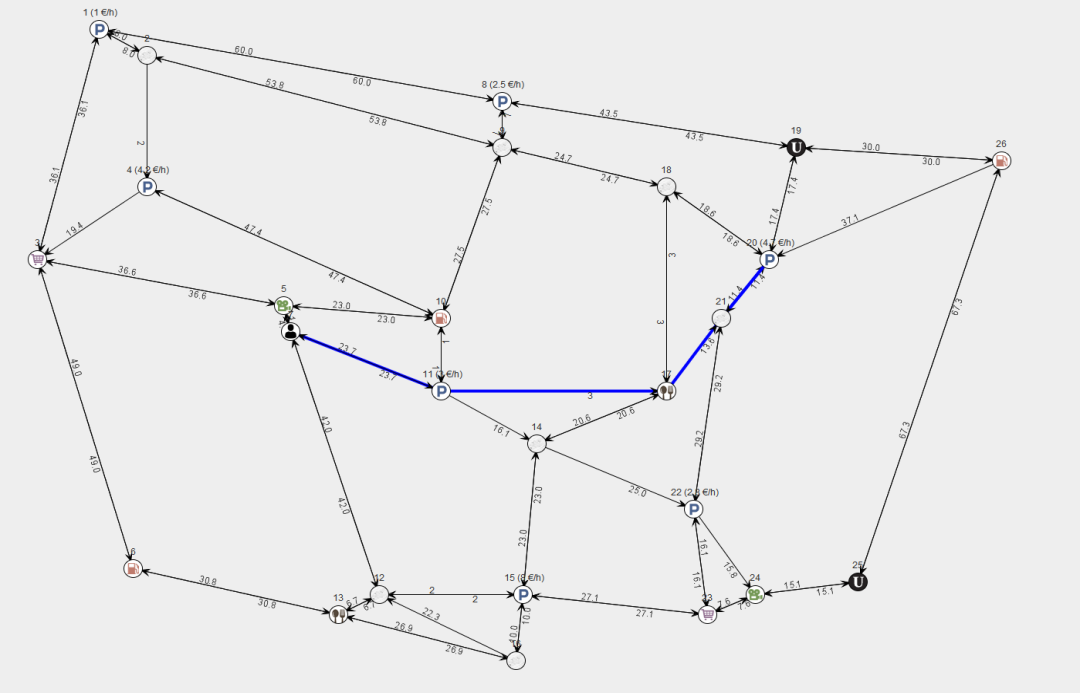
*Partida 🡪 Posto Abastecimento 🡪 Destino mais barato*

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.



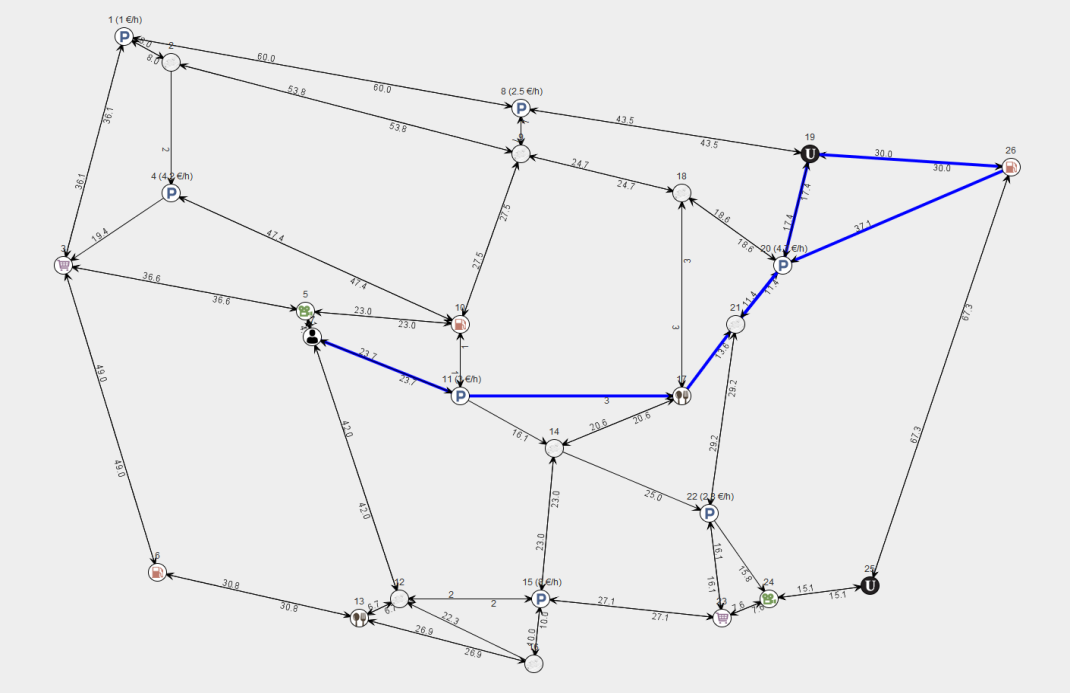
*Partida 🡪 Destino pelo caminho mais curto*

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***

Durante a realização deste projeto não se apresentaram grandes dificuldades a nível de programação mas sim de edificação e planeamento do mesmo.

Efetivamente, o maior obstáculo terá sido a compreensão da utilidade do software apresentado (*GraphViewer* e *Parser*) e de como utilizá-lo. Após expormos as nossas duvidas à professora optámos por descartar o *parser*, visto que este não ia ao encontro do que realmente precisávamos e ainda tinha alguns problemas na sua implementação. Assim, consideramos desnecessário e não benéfico.

No entanto, foi na mesma usado o *GraphViewer* para visualização do nosso grafo, por este motivo, o *setup* do projeto em diferentes IDE’s (*Visual* *Studio* e *Eclipse*) demorou um pouco mais do que seria esperado, devido a problemas de compilação.

***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.
* Estruturação e revisão do relatório.
* Percentagem: 30%

**João Azevedo**

* Redação do relatório.
* Pesquisa sobre os algoritmos a utilizar.
* Percentagem: 30%

**Julieta Frade**

* Implementação dos algoritmos usados.
* Percentagem: 40%

ESCREVAM MAIS SOBRE VOCES

***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.