

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*

*(9 de abril de 2017)*

Índice

|  |  |
| --- | --- |
| **Introdução** | 2 |
| Descrição do Tema | 2 |
| Identificação e Formalização do Problema | 3 |
| **Solução Implementada** | 4 |
| **Diagrama de Classes** | 6 |
| **Casos de Utilização** | 7 |
| Classes | 7 |
| Ficheiros | 7 |
| Algoritmos | 8 |
| Programa | 9 |
| **Dificuldades** | 10 |
| **Distribuição do Trabalho pelos Elementos do Grupo** | 11 |
| **Conclusão** | 12 |
| **Bibliografia** | 13 |

***Introdução***

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

Este projeto realizado no âmbito da cadeira de CAL 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 valores, uma vez que, por exemplo, o caminho pretendido pode ser o que nos leva ao destino pelo caminho mais curto.

Assim, identificamos como objetivos principais do trabalho os seguintes:

* Possibilidade do 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
* Possibilidado 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 reler o enunciado proposta e formalizar o problema, subdividindo-o em subproblemas até que se tornassem elementares e de simples resolução.

Assim, identificamos e modulámos o problema em alguns tópicos:

1. A escolha de uma área real de teste
2. O uso do parser providenciado
3. A extração de informação dos ficheiros de texto
4. 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
5. A determinação do caminho ideal no grafo usando o algoritmo adequado
6. A criação de um menu que serviria de interface com o automobilista

***Solução Implementada***

Assim, rapidamente concluímos que a melhor solução seria não usar o parser mas definirmos uma área como área de teste do programa.

O primeiro passo foi então definir a área a testar e se para tal usaríamos o parser providenciado ou nós próprios criarimos os ficheiros contendo a informação necessária. Rapidamente nos apercebemos que seria mais prático e elucidativo definirmos os nossos próprios ficheiros de teste e assim o fizemos.

Efetivamente, tendo os ficheiros prontos e a 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 template por nós definido contendo o seu ID, o seu nome, indicativo do local (cinema,hospital,...), e um par de coordenadas no mapa (x,y).

Os parques são objetos que contêm os nós a eles associaods, 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 tratar-se de um grafo pesado com apenas números inteiros positivos, foi usado o algoritmos de Dijkstra.

***Diagrama de Classes***

No fim do trabalho fazer UML disto em 5 mins e colar aqui

***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 metodos 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 boleano 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 boleano 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

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

*Algoritmos – Algoritmo de Djikstra*

Para a realização do projeto utilizámos apenas um algoritmo, sendo um tema leccionado na cadeira, para a determinação do caminho mais curto num grafo pesado com valores apenas positivos. Falamos pois do algoritmo de D jikstra.

Em termos de analisa 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 é responsável pelo cálculo do caminho mais curto do local de partida ao destino. É usado tanto para a determinação do caminho mais curto como do caminho que leva ao parque menos custoso em redor do destino. Faz parte do cálculo ainda dos trajetos:

1. Partida 🡪 Destino
2. Partida 🡪 Abastecimento 🡪 Destino

|  |  |  |
| --- | --- | --- |
| V | Arestas | T(ms) |
| 26 | 43 | < 1 |
| 200 | 100 | 7 |
| 600 | 300 | 18 |
| 1000 | 500 | 28 |

*Programa*

Mais texto, mas não sei se este ponto é necessário.

Tambem acho este parvo. Mas se for preciso cago aqui alguma coisa.

***Dificuldades***

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

Efetivamente, provavelmente o maior obstáculo terá sido a compreensão da utilidade do software apresentado (GraphViewer e Parser) e de como utilizá-lo.

Tal como referido, optámos por descartar o parser por considerarmos desnecessário e não benéfico mas usámos o GraphViewer para visualização do nosso grafo.

Ainda relativamente a este ponto, o setup do projeto em diferentes IDE’s (Visual Studio e Eclipse) demorou talvez um pouco mais do que seria esperado.

***Distribuição do Trabalho pelos Elementos do Grupo***

A Julieta fez 90% rip.

Yap :D mais um 0 pa mim lets go :D

Conde: relatorio 10/10

Ju: trabalho 10/10

Sofia: pics 10/10

***Conclusão***

Concluio que cal é giro

***Bibliografia***

Mais texto.

Bibliografia in 2k17 lul