

**MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO**

CONCEPÇÃO E ANÁLISE DE ALGORITMOS

EcoRouting para veículos elétricos – Parte II

**MIEIC**

**Turma 5 / GRUPO E**

Dinis Moreira / 201503092 / up201503092@fe.up.pt

Joana Ramos / 201605017 / up201605017@fe.up.pt

Pedro Neto / 201604420 / [up201604420@fe.up.pt](mailto:up201604420@fe.up.pt) 20/05/2018

**Tema**

O projeto desenvolvido procura encontrar a melhor trajetória para um veículo elétrico chegar até ao seu destino pretendido, tendo em conta a energia disponível nesse veículo bem como o seu consumo consoante a variação de elevação do terreno: cada veículo consumirá mais bateria quanto mais acentuada for uma subida, e recarregará a bateria (gastará menos do que o consumo predefinido) quanto mais acentuada for a descida. É tida em conta a possibilidade de ser necessário carregar a bateria do veículo durante o itinerário, havendo para isso postos de recarga.

Dados dois pontos (um de origem e um de destino), o programa deverá calcular e apresentar o melhor itinerário, com base no grafo escolhido e nas limitações do veículo elétrico, sabendo a sua autonomia, consumo e energia atual da bateria.

Parte II

Esta parte do projeto pretende permitir a pesquisa de pontos de recarga em cruzamentos entre ruas. Esta pesquisa pode ser feita de 2 maneiras, através de um algoritmo de pesquisa exata, ou um algoritmo de pesquisa aproximada.

A pesquisa começa por pedir ao utilizador que introduza uma string para o nome da rua que este pretende procurar. O algoritmo é executado e é retornada uma lista de ruas que obedecem aos padrões de pesquisa do utilizador, assim como em quais destas ruas existem pontos de recarga.

**Solução implementada**

Desenvolvemos a solução para o cálculo do itinerário com base no algoritmo de *Dijkstra* aplicado com uma técnica de programação gananciosa. O algoritmo escolhe o itinerário mais “curto” sendo que o significado real de curto é o menor consumo de energia (tendo, portanto, em conta a variação da elevação do terreno). Caso esse consumo mínimo seja superior à energia atual da bateria, escolhe o caminho com menor consumo de energia e que passe num ponto de recarga.

Parte II

Para a pesquisa exata, foi desenvolvida uma solução com base no algoritmo de Knuth-Morris-Pratt. Este algoritmo recebe duas strings, um padrão P, e um texto T. Este efetua um pré-processamento do padrão, o que vai permitir uma pesquisa no texto de forma linear, evitando comparações inúteis, sendo assim a sua complexidade temporal O(|T|+|P|). Quanto á complexidade espacial, visto que é necessário guardar um array de inteiros do tamanho do padrão, esta será O(|P|).

Para a pesquisa aproximada, …

// Explicação do algoritmo utilizado para a pesquisa aproximada e complexidades

…

// Explicação do algoritmo do Pedro

**Estruturas de dados**

* *Interface*

Contém o grafo utilizado para os cálculos, métodos de inicialização do programa tais como leitura de ficheiros e inicialização das estruturas de dados e métodos para gerar e processar interação com o utilizador.

* *Graph*

Classe para armazenar a informação relativa a um grafo (vetores de nós e arestas) bem como métodos para operações básicas sobre a estrutura (adição/remoção).

* *Node*

Informação relativa a um nó (id, coordenas, arestas adjacentes …), métodos de encapsulamento e de operações básicas.

* *Edge*

Informação relativa a uma aresta (id, ids dos nós de origem/destino, peso da aresta, …) e métodos de encapsulamento.

* *Vehicle*

Dados de um veículo (incluindo as viagens que um utilizador pretende realizar com ele) e métodos de encapsulamento.

* *Trip*

Dados de uma viagem; está sempre associada a um veículo e a dois nós.

* *Graphviewer* e *Connection*

Classes de apoio à tecnologia de visualização de grafos.

**Utilização**

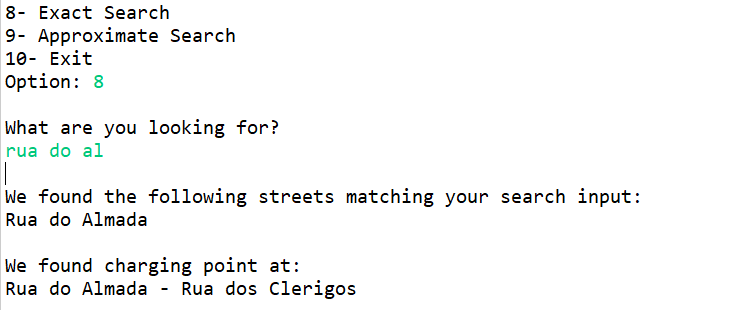
O programa começa por ler todos os ficheiros *.txt* com informações acerca de nós, arestas, veículos e viagens. É possível testar as funções de cálculo de itinerários com esta informação.

Estão implementadas funcionalidades para o utilizador poder adicionar veículos e viagens, que são adicionados às estruturas de dados do programa e aos ficheiros para posterior utilização.

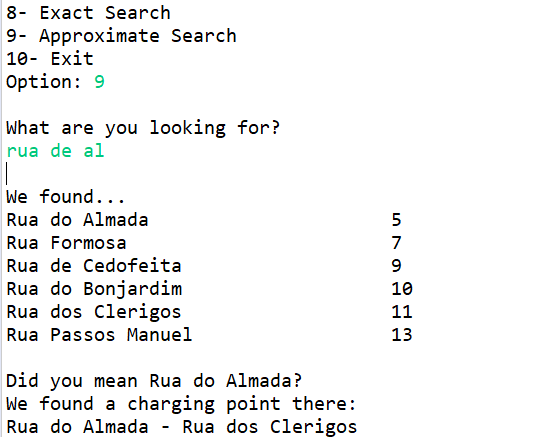
Parte II

Para realizar uma pesquisa, o utilizador deve começar por selecionar a opção de pesquisa que deseja (exata ou aproximada). Após esta seleção, o utilizador será questionado sobre qual o nome do local que este procura, e de seguida este deverá introduzir uma string. A pesquisa é realizada e de seguida são apresentados os resultados obtidos:

Pesquisa exata: São devolvidas as ruas que possuem no seu nome a sub-string introduzida pelo utilizador, seguidas dos cruzamentos entre pelo menos uma dessas ruas onde existem pontos de racarga (caso exista algum).



Pesquisa aproximada: …



**Principais Dificuldades**

**Distribuição de trabalho**

* Dinis Moreira

Contribuição para os métodos da classe Interface; algoritmo *Dijkstra*; extração de dados dos mapas; manipulação de ficheiros; elaboração do relatório.

Parte II

Reestruturação inicial do código, realização do algoritmo de pesquisa exata e respetiva documentação, elaboração do relatório.

* Joana Ramos

Funções para leitura de ficheiros e inicialização das estruturas de dados correspondentes; acesso, adição e remoção de elementos em grafos; algoritmo de *Dijkstra*; funções de utilização da API de visualização de grafos; elaboração do relatório.

Parte II

* Pedro Neto

Estruturação do projeto; construção do grafo; leitura e manipulação de ficheiros; cálculo e algoritmo de Dijkstra; gestão de veículos e viagens; documentação.

Parte II