City Sightseeing

Relatório 2ª Parte

Concepção e Análise de Algoritmos

Mestrado Integrado em Engenharia Informática e Computação

Porto, 30 de Maio de 2016

Turma 6, Grupo G:

João Henrique Poceiro Vieira de Araújo up201303962@fe.up.pt
José Pedro Teles da Silva Pereira up201305101@fe.up.pt

Índice

Introdução

A cidade do Porto é cada vez mais um dos maiores destinos de turismo na Europa. Um dos meios disponíveis para o turista conhecer a cidade é o *city sightseeing*, ou seja, embarcando em autocarros de dois pisos, com o piso superior parcialmente ou totalmente aberto, que percorrem os locais onde os turistas podem facilmente avistar monumentos, prédios e outros pontos de interesse da cidade.

No âmbito da unidade curricular de *Concepção e Análise de Algoritmos*, ficámos encarregues de desenvolver uma aplicação que permite a listagem e procura de possiveis itinerários e passageiros assim como a inscrição num determinado itinerário. Os itinerários são calculados através dos pontos de interesse por onde passam, sendo o caminho escolhido aquele que for mais curto.

Problema

1) Dados

A aplicação nesta entrega permite receber como dados de entrada as informações das ruas (coordenadas, nome, comprimento e ligações), os pontos de interesse da cidade a percorrer (com as ruas onde estes se localizam), a lista de passageiros e os vários itinerários disponíveis.

Para representar a cidade do Porto, é utilizado um **grafo dirigido pesado** G = (V,E).

- Vértices (V): cada vértice corresponde a uma rua (por convenção, um ponto médio da sua extensão);
- Arestas (E): correspondem às ligações entre ruas, com o respectivo tempo para nos deslocarmos entre elas.

Para representar um itinerário é utilizado parte do grafo utilizado para representar a cidade do Porto. Os passageiros encontram-se representados num vetor de passageiros pelo nome e número do itinerário a que estão associados.

2) Restrições

O trajecto tem obrigatoriamente de se iniciar na Avenida dos Aliados, pois é este o ponto de encontro para todos os turistas. Por outro lado, sendo um grafo dirigido, podem existir ruas de sentido único. Cada passageiro tem que ter um itinerário associado e cada autocarro tem um número máximo de passageiros.

3) Limites e condições da aplicação

A solução aqui apresentada por nós não é a melhor a nível de complexidade temporal, no entanto, não foi possível melhorar o seu desempenho temporal, sem comprometer a sua eficiência e a qualidade da pesquisa efetuada.

4) Resultado Esperado

O algoritmo deverá ser capaz de descobrir um itinerário que possua um determinado ponto de interesse ou um determinado passageiro. O utilizador tem também a possibilidade de se inscrever num itinerário, sendo gerado um mapa, destacando o caminho selecionado e os pontos de interesse por onde este passa.

Solução

1) Técnicas de concepção

Para a resolução do problema utilizou-se uma adaptação dos algoritmos de pesquisa de strings aproximada e exacta abordados nas aulas práticas e teóricas da unidade curricular.

Utilizámos a estrutura vector para armazenar a informação relativa aos passageiros.

Criou-se uma classe passageiro para armazenar informações acerca do mesmo, nomeadamente o nome e o itinerário a que está associado.

A base de dados fornecida consiste em pontos de interesse reais e passageiros ficticios que estão gravados inicialmente num documento de texto.

2) Algoritmos utilizados

Para este projecto implementamos dois algoritmos um algoritmo de pesquisa exata e aproximada de *strings*, que só é utilizado caso o de pesquisa exata não encontre a string de pesquisa, utilizando as bases obtidas a nível teórico na unidade curricular e que servirá para calcular a distância de edição entre duas *strings*.

Algoritmo Knuth-Morris-Pratt

O algoritmo de Knuth–Morris–Pratt, que procura uma ocorrência de uma string dentro de uma substring empregando a simples técnica de que quando aparece uma diferença, a palavra tem em si a informação necessária para determinar onde começar a próxima comparação.

Pesquisa aproximada

Calculo de combinações

Criar um vector de C_{n-1}^{n} , em que n é o número de palavras introduzidas na pista fornecida pelo utilizador. Este algoritmo é utilizado uma vez que uma das palavras pode não pertencer ao nome de um passageiro ou de um ponto de interesse e temos que procurar a melhor combinação.

Algoritmo *EditDistance*

Este algoritmo vai receber como argumento duas *strings*:

- toSearch - as palavras dadas pelo utilizador (pistas) a procurar;
- word - que será cada palavra do nome de um passageiro ou de um ponto de interesse.

Vai assim devolver um inteiro:

- d[n] - distância entre as *strings*.

O algoritmo vai assim inicializar uma matriz que é gradualmente preenchida com a distância entre os argumentos dados. Após a conclusão do algoritmo, a célula inferior direita da matriz apresenta a distância entre as *strings*. Esta distância *d[n]* vai ser um *int* que corresponde ao número de operações necessárias para transformar a *toSearch* na *string word1*, quanto menor a distância, mais semelhantes serão as strings e quando a distância for zero, as strings serão exatamente iguais. A distância de edição entre P(padrão) e T(texto) é o menor número de alterações necessárias para transformar T em P, em as alterações podem ser: substituir um caracter por outro, inserir um caracter e eliminar um caracter.

3) Complexidade dos algoritmos

Algoritmo Knuth-Morris-Pratt

O algoritmo de Knuth–Morris–Pratt tem uma complexidade temporal:

$$O(|T| + |P|)$$

Pesquisa aproximada

Calculo de combinações

Este algoritmo calcula todas as combinações utilizados todas as palavras da Pista sendo que no máximo não usa uma. Sendo então a sua complexidade de (N = nr. de pistas):

$$O({}^{N}C_{N-1} + 1)$$

Algoritmo *EditDistance*

Sendo que, neste algoritmo, todos os caracteres de ambas as *strings* são analisados, a complexidade temporal será:

$$O(|toSearch| . |word|)$$

A matriz de que este algoritmo faz uso será uma matriz bidimensional com |*toSearch*| linhas e |*word1*| colunas para guardar progressivamente a distância de edição entre as *strings*. Podemos então concluir que a complexidade espacial é igual à complexidade temporal.

Diagrama de Classes

O diagrama de classes é exactamente igual ao da primeira parte do projecto, com excepção da nova classe *Passengers*.

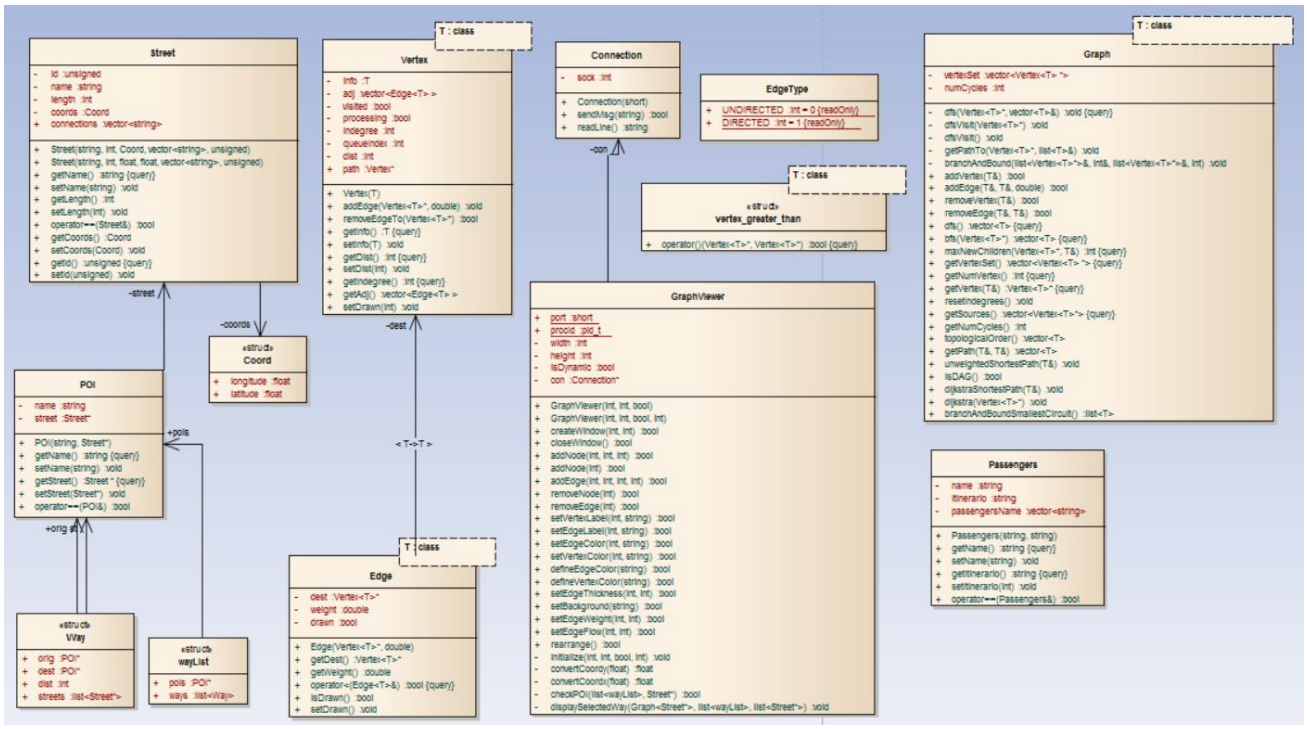

Fig 1. Diagrama de Classes

Dificuldades encontradas

Não foram encontradas grandes dificuldades ao longo do desenvolvimento desta segunda parte do projecto.

Esforço dedicado

A colaboração de ambos os membros do grupo foi equitativa:

- João Araújo - 50%
- José Teles - 50%

Referências

ROSSETTI, R.; ROCHA, A.P.; - *Slides* de CONCEPÇÃO E ANÁLISE DE ALGORITMOS, 2015/2016

“Knuth–Morris–Pratt algorithm” Wikipedia. Wikipedia Foundation, n.d. Web Maio.2016
https://en.wikipedia.org/wiki/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm