

**Mestrado Integrado em Engenharia Informática e Computação**

**Conceção e Análise de Algoritmos**

**Central de atendimento de urgências**

Turma 4 / Grupo 9

Carlos Miguel da Silva de Freitas (201504749) - [up201504749@fe.up.pt](mailto:up201504749@fe.up.pt)

Luís Noites Martins (201503344) – [up201503344@fe.up.pt](mailto:up201503344@fe.up.pt)

Rui Emanuel Cabral de Almeida Quaresma (201503005) - [up201503005@fe.up.pt](mailto:up201503005@fe.up.pt)

9 abril 2017

Índice

[Introdução 3](#_Toc478910297)

[Identificação do Problema 4](#_Toc478910298)

[Descrição do Problema 5](#_Toc478910299)

[Input 5](#_Toc478910300)

[Dados de entrada 5](#_Toc478910301)

Introdução

Este trabalho foi realizado no âmbito da Unidade Curricular de Conceção e Análise de Algoritmos do 2º ano do MIEIC.

O objetivo é recorrer aos algoritmos de cálculo do caminho mais curto, abordados nas aulas, para encontrar uma solução para um tema que nos foi proposto.

Neste relatório é descrito o problema, feita a sua formalização, bem como apresentada a solução implementada.

Identificação do Problema

Uma central de atendimento de urgências é responsável por receber chamadas telefónicas realizadas ao número de urgência (e.g. 112) e desencadear de forma coordenada os processos e os meios para o atendimento da chamada.

Para uma maior eficiência de resposta aos diversos tipos de emergência é necessário que seja enviado ao local da chamada quer o veículo que se encontre mais próximo quer a(s) entidade(s) consoante a gravidade da situação.

Dado este problema, procuramos desenvolver uma interface que permitia ao utilizador localizar a sua chamada num mapa, bem como o tipo de emergência e que lhe mostre o itinerário mais curto desde o(s) veículo(s) de emergência até ao local da chamada.

O caminho mais curto consiste na distância mais reduzida desde o local de um veículo até ao local da emergência.

Descrição do Problema

Pretende-se identificar qual o(s) veículo(s), de entre os disponíveis, que se encontra(m) mais perto do local da chamada (dado pelo utilizador) e de acordo com o tipo de emergência (dado pelo utilizador), mostrando de seguida o itinerário mais curto do(s) veículo(s).

Input

Criação de um grafo, G = (V, A), de cruzamentos e ruas em que:

* V (vértices): representam pontos de interceção de duas ruas, ou seja, cruzamentos;
* A (arestas): ligações entre dois cruzamentos (distâncias entre dois vértices);
* Nó que representa o local da chamada;
* Tipo que representa o tipo de emergência.

Output

Nós (cruzamentos) pelos quais o(s) veículo(s) passou(ram) desde o local onde se encontrava(m) até ao local da emergência, pertencentes ao caminho mais curto/otimizado.

Dados de entrada

* Quatro ficheiros que contêm os nós (cruzamentos) do mapa, com os ids, coordenadas x e y dos mesmos:
* Um ficheiro que contém os nós onde não está localizado nenhum veículo ou hospital, tem os ids, o x e o y dos nós;
* Um ficheiro que contém os nós onde estão localizadas as ambulâncias;
* Um ficheiro que contém os nós onde estão localizados os carros dos bombeiros;
* Um ficheiro que contém os nós onde estão localizados os carros da PSP;
* Um ficheiro que contém os nós onde estão localizados os hospitais.
* Um ficheiro que contém as arestas, ou seja, as ligações entre os diferentes vértices, com os ids das mesmas e os ids dos nós de início e de fim de cada uma.
* Um ficheiro que contém as ruas, ou seja, conjuntos de arestas que podem ser unidirecionais ou bidirecionais, com os ids e o nome das mesmas, a indicação sobre a se são ou não bidirecionais e ainda os ids dos nós que as compõem.

Restrição

No mesmo percurso não é permitido passar pelo mesmo cruzamento duas vezes.

Objetivo

Providenciar ao utilizador o trajeto mais eficiente para os veículos de emergência solicitados a partir do local à sua escolha.

Formalização do Problema

Input

Output

Objetivo

Restrição

Solução

Algoritmos

Optamos por implementar dois algoritmos, o de Dijkstra e o de Floyd-Warshall.

O algoritmo de Dijkstra, que encontra o caminho mais curto num grafo dirigido ou não dirigido com tempo de execução . Tem a restrição de só poder ser usado em grafos com pesos positivos, que por sua vez se verifica no tema do trabalho.

Este algoritmo é um algoritmo ganancioso, ou seja, toma as decisões que parecem melhores no momento, o que lhe permite determinar o conjunto de melhores caminhos intermediários. O peso de cada aresta está associado à distância, calculada através das coordenadas (x e y) de cada cruzamento.

O algoritmo de Floyd-Warshall, que encontra o caminho mais curto num grafo dirigido com tempo de execução , e é mais aconselhado para grafos densos. Pode ser usado em grafos com pesos positivos, bem como com pesos negativos. O peso de cada aresta está associado à distância, calculada através das coordenadas (x e y) de cada cruzamento.

No nosso trabalho o algoritmo de Dijkstra é mais eficiente dado que temos um mapa com poucos nós (pouco denso). Se usássemos o algoritmo de Floyd-Marshall perderíamos eficiência, dado que ele criaria uma matriz com o peso da ligação de cada dois nós, o que para um grafo pouco denso não seria o melhor.

Funcionamento do programa

Meter o k acontece durante a execução do programa.

Meter imagens do programa a correr.

Análise da complexidade

Avaliação analítica da complexidade temporal

Tal como enunciado acima, a complexidade temporal do algoritmo de Dijkstra é .

Avaliação analítica da complexidade espacial

NÃO SEI

Avaliação empírica da complexidade temporal

Para analisar a complexidade temporal medimos os tempos de execução do programa usando diferentes dados de entrada (diferentes números de vértices, arestas e mudando os locais onde se encontravam os veículos, hospitais e da chamada).

Lista de casos de utilização

* Leitura dos dados de ficheiros representativos das componentes de um mapa;
* Identificação dos veículos a deslocar;
* Identificação do caminho mais curto desde esses veículos até ao local da chamada;
* Possível retorno de um dos veículos ao hospital mais próximo através do caminho mais curto;
* Visualização do mapa usando o GraphViewer;
* Apresentação do caminho mais eficiente usando o GraphViewer.

Diagrama de classes

METER AKI O UML

Principais dificuldades

No início do projeto deparamo-nos com alguns entraves à sua execução, enunciados de seguida:

* Como obter um mapa do site openstreetmaps e como convertê-lo para ficheiros de texto, e fazer a respetiva leitura;
* Perceber qual o melhor algoritmo a utilizar.

Soluções encontradas

Decidimos criar o nosso próprio mapa (criação de ficheiros de nós, arestas e ruas).

Concluímos que o algoritmo mais eficiente para o nosso tema é o de Dijkstra.

Contribuição dos membros do grupo

Todos os membros contribuíram de forma equitativa para a realização do projeto.

Conclusão