Routing Algorithm for Ocean Shipping and Urban Deliveries

2º Projeto D.A.

LEIC | 2ºANO | Turma 3 | Grupo 5

José Lopes - up202208288
Mário Araújo - up202208374
Mariana Pereira - up202207545

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Diagrama de classes e ficheiros

Diagrama de classes e ficheiros

Nome	Nome dos ficheiros	Tipo
App	App.h e App.cpp	Classe
Coordinates	Coordinates.h e Coordinates.cpp	Classe
Edge	Edge.h e Edge.cpp	Classe
FileParse	FileParse.h e FileParse.cpp	Namespace
Graph	Graph.h e Graph.cpp	Classe
MutablePriorityQueue	MutablePriorityQueue.h	Classe
UI	UI.h e UI.cpp	Funções apenas
Vertex	Vertex.h e Vertex.cpp	Classe

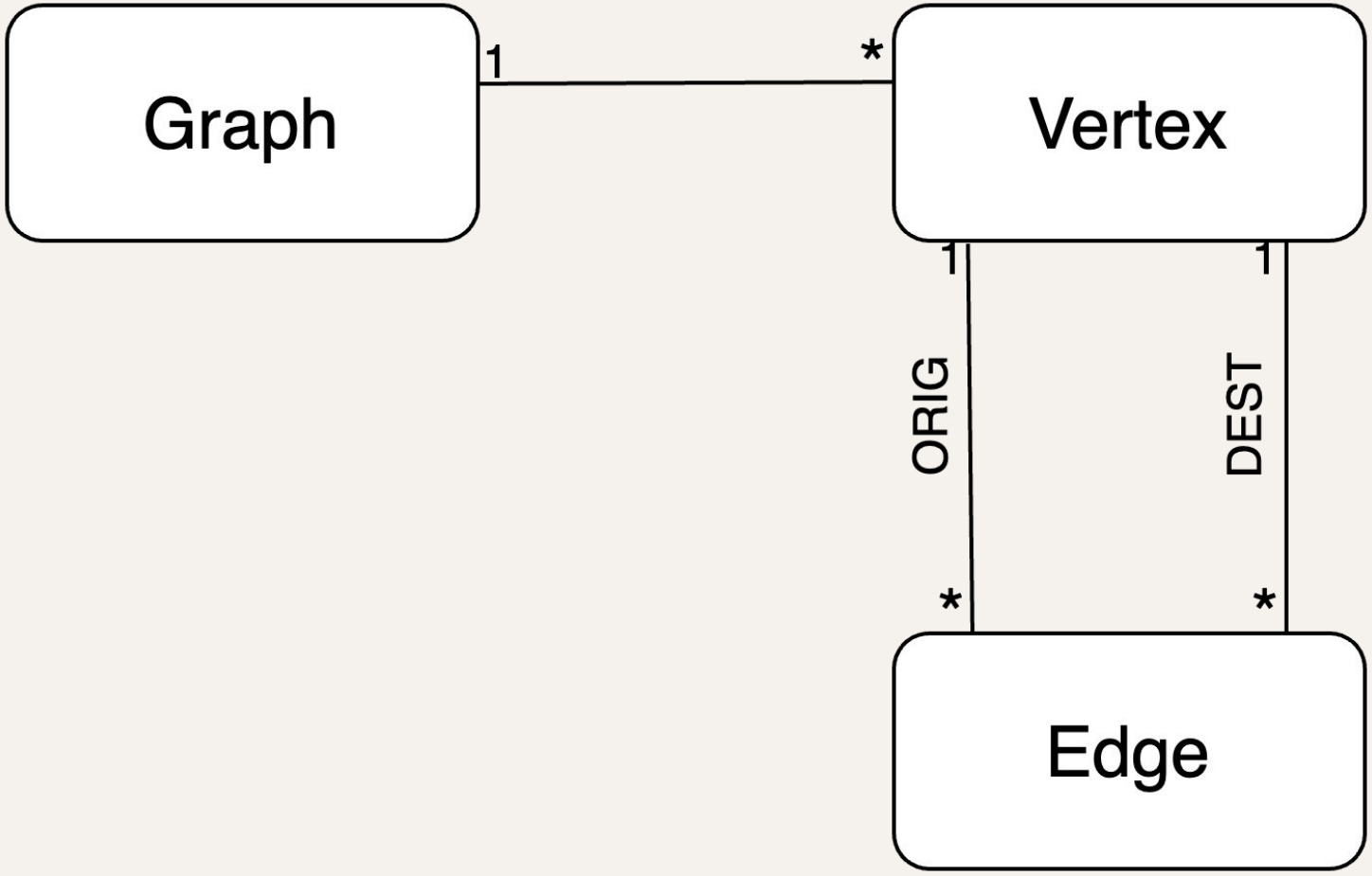
Como é efetuada a leitura dos ficheiros de dados?

- Para efetuar a leitura de dados, é requisitado ao utilizador que indique o nome dos ficheiros que serão utilizados, assim como o caminho para o diretório onde estão localizados esses ficheiros e mais algumas informações importantes para permitir uma leitura sem erros.

02

Estrutura do Grafo

Estrutura do Grafo



Algoritmos implementados

TSP - Backtracking

- O algoritmo que resolve o TSP com backtracking implementado utiliza uma função de bounding para avançar ramos que nunca iriam conduzir à solução ideal. Para evitar estar constantemente a recalcular o tamanho do caminho, guarda o valor do tamanho.
- A função de bounding verifica se o tamanho do caminho atual é superior ao tamanho do melhor caminho encontrado até à aquele momento, nesse caso deixa de explorar o ramo, e tenta outros caminhos.

- Considerando o pior cenário possível, isto é, quando o grafo está totalmente conectado, a complexidade temporal será O(n!), onde ‘n’ é o número de vértices. Uma vez que a complexidade é muito elevada, este algoritmo só é uma abordagem válida para grafos muitos pequenos.

TSP - Triangular Approximation

- O algoritmo de aproximação triangular divide-se nos seguintes passos:
  1. Encontrar uma MST para o grafo;
  2. Ler o grafo usando travessia preorder, para tal, só são utilizadas as arestas que o prim usou.
  3. Verificar se existe uma arestas que liga todos os grupos de vertices adjacentes.
- Para obter a MST, utilizamos o Prim com uma MutablePriorityQueue.

- O motivo pelo qual é importante obter a MST e, posteriormente, a preorder, é porque assim garantimos que o algoritmos utiliza o maior número possível de arestas que foram utilizadas na MST. Consequentemente, essas arestas vão ser mais leves do que outras que ligam o vértice em questão a qualquer outro, caso contrário não estariam na MST.

- A complexidade temporal é O(E log V), onde ‘E’ é o número de arestas e ‘V’ o número de vértices, já que o Prim utiliza uma MutablePriorityQueue. Este algoritmo permite, no pior dos casos uma aproximação de 2 vezes a solução ideal.

TSP - Christofides

- O algoritmo de Christofides divide-se nos seguintes passos:
  1. Construir uma MST;
  2. Calcular o grau de todos os vértices (usando apenas as arestas utilizadas na MST);
  3. Encontrar uma “Perfect matching” mínima entre os vértice com grau ímpar;
  4. Encontrar um ciclo Euleriano, usando as arestas do “Perfect matching” e da MST.

- A lógica por trás deste algoritmo é, tal como na aproximação triangular, usar o maior número de arestas leves possíveis entre os vértices

- A complexidade temporal será O(n^3), onde ‘n’ é o número de vértices. Este algoritmo garante, no pior caso, uma aproximação 1,5 vezes pior que a solução ideal.

TSP - Nearest Neighbour (NN)

- O algoritmo de NN baseia-se nos seguintes passos:
  1. Escolher qualquer vértice para ser o inicial;
  2. Encontrar o vértice ainda não visitado mais próximo;
  3. Mover para o vértice mais próximo;
  4. Repetir o passo 2. e 3. até completar o ciclo hamiltoniano.

- Este algoritmo é muito bom, tal como foi possível observar pelos resultados quando comparados com o Christofides e aproximação Triangular, para grafos muito conectado. Contudo para grafos que tenham vértices pouco conectados, ficava quase sempre preso e não encontra caminhos. Este algoritmo é muito rápido.

- Assumindo que o o graph já está totalmente conectado, a complexidade temporal será O(V x n), onde ‘V’ é o número de vértices e ‘n’ o número médio de vértices ligados a cada vértice, porque é necessário percorrer todos os vértices e encontrar o vértice mais próximo.

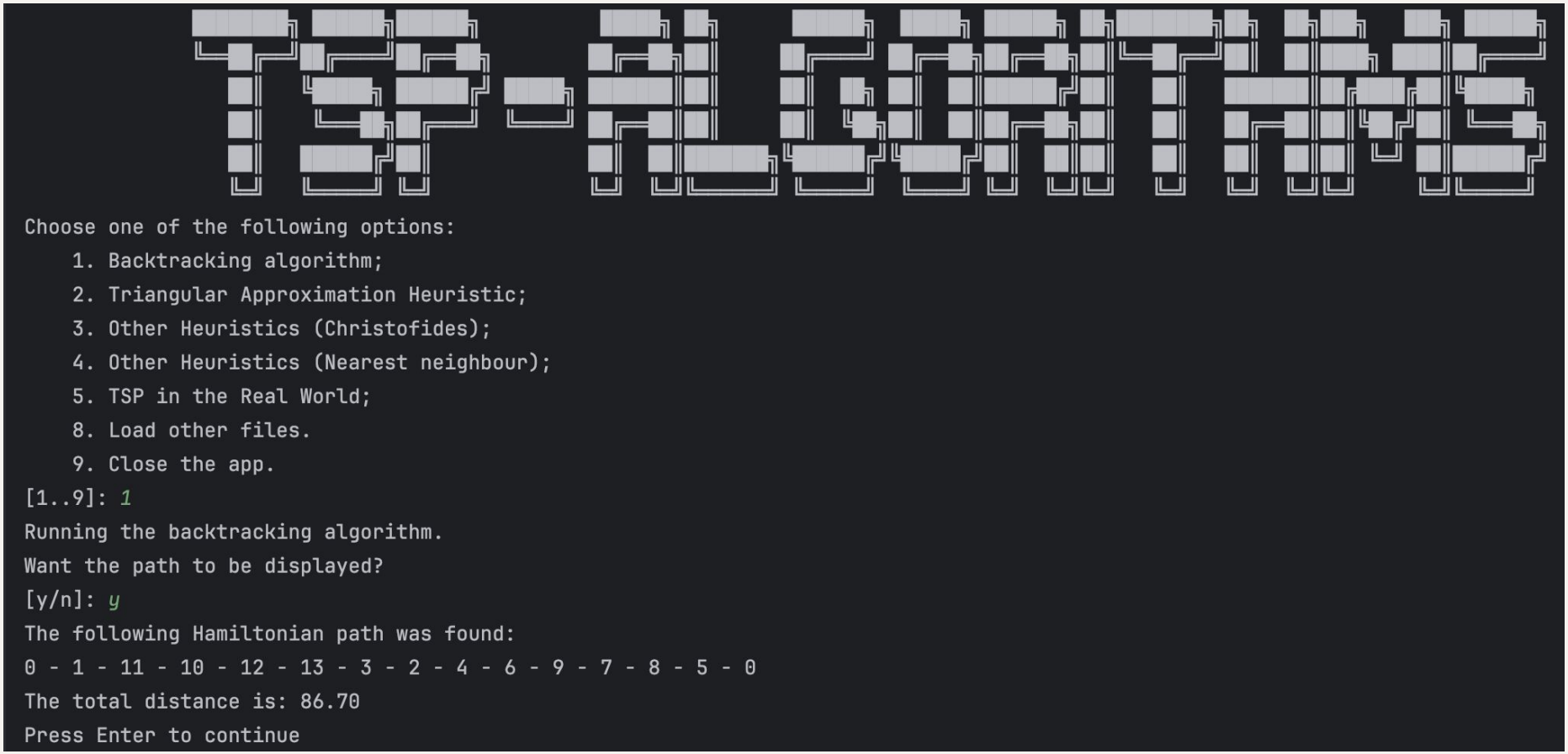
TSP - Real-World

- O algoritmo para os grafos reais baseia-se nos seguintes passos:
  1. Ordenar as arestas de um vértice, pela aquela que conduz a um vértice com menor grau. Em caso de empate, por peso da aresta;
  2. Utilizando um algoritmo de backtracking, é procurado um hamiltoniano;
  3. Depois de ser encontrado um hamiltoniano, é dada a opção de melhorar o caminho com um algoritmo 2-opt.

- Como os grafos reais são muito grandes é necessário um algoritmo que seja muito rápido, logo optámos pelo algoritmo descrito em cima. O motivo pelo qual funciona muito bem para encontrar hamiltonianos, é porque deixa os vértices que têm maior conectividade para o fim, diminuindo assim a probabilidade de ficar preso, tendo em conta que esses vértices estão ligados a muitos outros.
- Porém este algoritmo tem um problema, não garante que vai encontrar um caminho ideal ou próximo do ideal, por esse motivo optamos por dar a possibilidade de melhor o caminho com um algoritmo 2-opt.

- Em teoria, por usar backtracking a complexidade será no pior caso O(n!), onde ‘n’ é o número de vértices, mas na prática o algoritmo é muito rápido quando comparado com o Christofides e a aproximação Triangular. A complexidade do 2-opt é O(k x V x V), onde ‘k’ é o número de iterações e ‘V’ o número de vértices.

Design da interface



Funcionalidades a destacar

- As funcionalidades que achámos mais desafiantes e que implicaram mais tempo de trabalho foram as funções de:
    - Algoritmo para os grafos reais.
    - O algoritmo de Christofides.

Dificuldades e Participação

- Nós consideramos que o enunciado era bastante claro, o que facilitou desenvolver os algoritmos. O facto de ser o 2º projeto também foi um fator a ter em conta, pois já sabíamos o que é que era preciso melhorar em relação ao 1º projeto.

- José Lopes - 33.3 %

- Mariana Pereira - 33.3 %

- Mário Araújo - 33.3 %