

U.PORTO

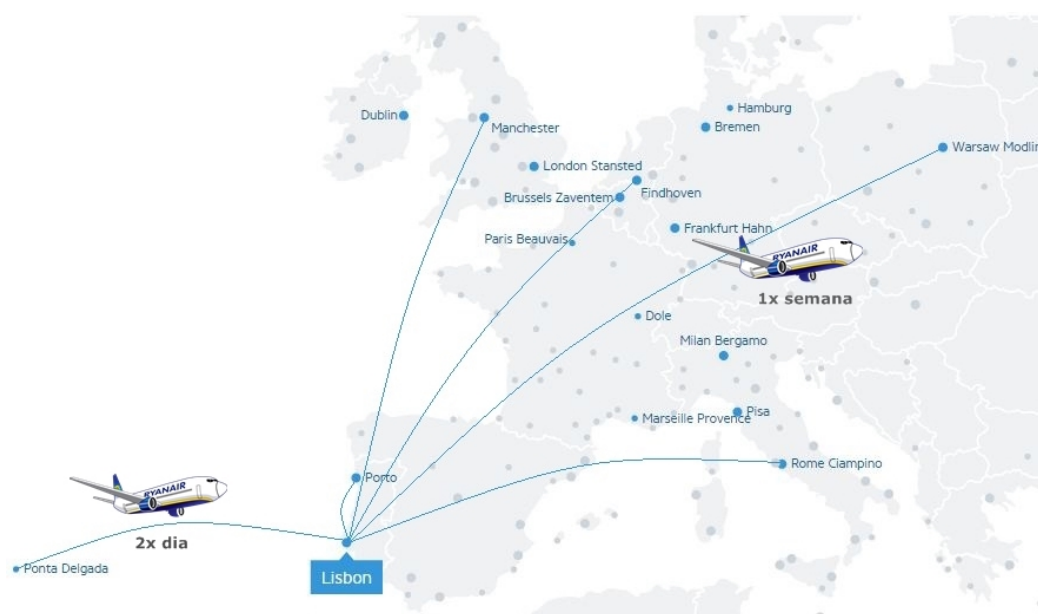
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

2ºano – MIEIC – 2016/2017

Agência de Viagens

Conceção e Análise de Algoritmos

Turma 6 - Grupo A



Beatriz de Henriques Martins

João Carlos Almeida

João Miguel Monteiro

Porto, 9 de abril de 2017



Índice

INTRODUÇÃO.....	3
ENUNCIADO	4
DESCRIÇÃO DO PROBLEMA	5
FORMALIZAÇÃO DO PROBLEMA	6
SOLUÇÃO	7
Algoritmos	7
Estrutura do Programa	8
CASOS DE UTILIZAÇÃO.....	10
MÉTRICAS DE AVALIAÇÃO	11
DIAGRAMA DE CLASSES.....	12
PRINCIPAIS DIFICULDADES	13
CONCLUSÃO.....	14



Introdução

Nos dias que correm, cada vez mais, queremos viajar da melhor forma possível, mais rápida e barata. Existem as mais diversas apps e sites para aconselhamento. Estes executam pesquisas por datas e melhor preço.

No âmbito da disciplina de Conceção de Análise de Algoritmos do curso Mestrado Integrado em Engenharia Informática e Computação foi-nos proposto a realização de uma Agência de Viagens. O nosso programa permite ao utilizador criar novos clientes e associar aos clientes uma viagem com deslocação e estadia. Fazendo uma pesquisa por melhor preço.



Enunciado

Uma agência de viagens auxilia os seus clientes na pesquisa das melhores soluções para a marcação de viagens e respetivo alojamento.

Neste trabalho, pretende-se implementar um sistema que permita à agência de viagens conseguir uma seleção mais criteriosa de uma solução de acordo com as necessidades do seu cliente, procurando sempre minimizar o seu custo. Considere que o custo das viagens (bilhetes de avião, por ex.) é fixo para o período a considerar. Mas o preço do alojamento varia de acordo com o dia da semana ou período específico (festividade local, por ex.).

O cliente especifica:

- origem e destino para a sua viagem, ou - conjunto de locais que pretende visitar na viagem
- O cliente pode ainda especificar um tempo limite para realizar a viagem.

Avalie a conectividade do grafo, a fim de evitar que locais a visitar se encontrem em zonas inacessíveis.



Descrição do Problema

De uma forma geral, pretendemos resolver o problema de transportes entre países através de uma aplicação que, apenas recebendo o local de partida e chegada associado a datas, consiga devolver o transporte mais rápido ou barato. Dependendo das preferências do utilizador.

Input

Construção de um grafo, $G=(V, E)$ de transportes disponíveis no qual:

→ V – vértices

Representam todas as cidades dos meios de transporte disponíveis;

→ E – arestas

Representam todas as distâncias, tempos de viagens e custos;

→ Nó de início de viagem e nó de destino.

Introdução de dados

Um ficheiro com as cidades e respetivas coordenadas.

Um ficheiro com os dados dos transportes disponíveis.

Output

O transporte com os melhores preços e datas pedidas.

Objetivo

Facilitar aos utilizadores a escolha dos melhores trajetos consoantes os critérios por estes preferidos.



Formalização do Problema

Formalizamos agora o problema, de acordo com a solução que achamos mais vantajosa

Input

$G \langle V, E \rangle$

V: cidades

E: ligações entre as cidades

I: cidade inicial

F: cidade final

Output

Caminho = $\{V_i\}, i=1 \dots n$

Valor

Objetivo

Min(valor):

$$Valor = \sum_{i=1}^n (E_{i,j})$$



Solução

Algoritmos

Depois de analisar muito bem o problema, chegamos à conclusão de que o melhor algoritmo a implementar seria o **Algoritmo de Dijkstra**. Este encontra o menor caminho possível entre dois vértices de um grafo, dirigido ou não, e em tempo computacional

$$O([arestas + vértices] \log (vértices))$$

O **Algoritmo de Dijkstra** é um algoritmo ganancioso, ou seja, toma decisões que parecem ótimas no momento, determinando assim os conjuntos de melhores caminhos intermediários possíveis. O valor de cada aresta está associado à distância entre duas cidades, calculada através das coordenadas (latitude e longitude) de cada cidade. O algoritmo em questão não pode ser usado em grafos com peso negativo, dado que a distância entre duas cidades nunca pode ser negativa, é perfeitamente aplicável ao problema em questão.



Estrutura do Programa

Ao iniciar o programa, começamos por construir o grafo. Introduzindo as Cidades com as respetivas coordenadas para construir os caminhos entre si (vértices) e o hotel correspondente a cada cidade. De seguida, carrega a informação dos clientes.

```
C:\Windows\system32\cmd.exe
at edu.uci.ics.jung.visualization.renderers.BasicEdgeRenderer.drawSimpleEdge(BasicEdgeRenderer.java:90)
at edu.uci.ics.jung.visualization.renderers.BasicEdgeRenderer.paintEdge(BasicEdgeRenderer.java:62)
at edu.uci.ics.jung.visualization.renderers.BasicRenderer.renderEdge(BasicRenderer.java:78)
at edu.uci.ics.jung.visualization.renderers.BasicRenderer.render(BasicRenderer.java:38)
at edu.uci.ics.jung.visualization.BasicVisualizationServer.renderGraph(BasicVisualizationServer.java:367)
at edu.uci.ics.jung.visualization.BasicVisualizationServer.paintComponent(BasicVisualizationServer.java:321)
at javax.swing.JComponent.paint(Unknown Source)
at javax.swing.JComponent.paintToOffscreen(Unknown Source)
at javax.swing.RepaintManager$PaintManager.paintDoubleBuffered(Unknown Source)
at javax.swing.RepaintManager$PaintManager.paint(Unknown Source)
at javax.swing.RepaintManager.paint(Unknown Source)
at javax.swing.JComponent._paintImmediately(Unknown Source)
at javax.swing.JComponent.paintImmediately(Unknown Source)
at javax.swing.RepaintManager$4.run(Unknown Source)
at javax.swing.RepaintManager$4.run(Unknown Source)
at java.security.AccessController.doPrivileged(Native Method)
at java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(Unknown Source)
at javax.swing.RepaintManager.paintDirtyRegions(Unknown Source)
at javax.swing.RepaintManager.paintDirtyRegions(Unknown Source)
at javax.swing.RepaintManager.prePaintDirtyRegions(Unknown Source)
at javax.swing.RepaintManager.access$1200(Unknown Source)
at javax.swing.RepaintManager$ProcessingRunnable.run(Unknown Source)
at java.awt.event.InvocationEvent.dispatch(Unknown Source)
at java.awt.EventQueue.dispatchEventImpl(Unknown Source)
at java.awt.EventQueue.access$500(Unknown Source)
at java.awt.EventQueue$3.run(Unknown Source)
at java.awt.EventQueue$3.run(Unknown Source)
at java.security.AccessController.doPrivileged(Native Method)
at java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(Unknown Source)
at java.awt.EventQueue.dispatchEvent(Unknown Source)
```

Depois, é altura de o utilizador poder interagir com o programa em si. O usuário é obrigado a ter um ficha com os seus dados quando pretende agendar uma viagem.

```
*****
| Menu Agencia Viagens |
*****
1. Novo Cliente
2. Listagem de Cliente
3. Procura Cliente
4. Nova Viagem
5. Listagem de Destinos
6. Sair
Opção: 1

-----
| Menu Novo Cliente |
-----

NIF: 12345678

Nome: Nome de cliente

Morada: Morada de cliente, 123

Email: email@cliente.pt

Numero Telemovel: 912345678

Seguranca Social: 1234567890

Novo Cliente criado com sucesso!
*****
```




O Cliente escolhe uma data e um local de partida seguido de um local de chegada e a data de regresso.

(IMAGEM NOVA VIAGEM)

De seguida, é sugerido ao utilizador uma viagem com tudo o que este escolheu.

(IMAGEM Grafo com viagem e estadia)



Casos de Utilização

- Leitura e interpretação de dados de ficheiro relativos a um mapa;
- Escolha do melhor percurso em termos de preço ou tempo por viagem;
- Visualização de todo o mapa através do GraphViewer;
- Visualização do percurso escolhido através do GraphViewer.



Métricas de Avaliação

Avaliação Empírica do Desempenho

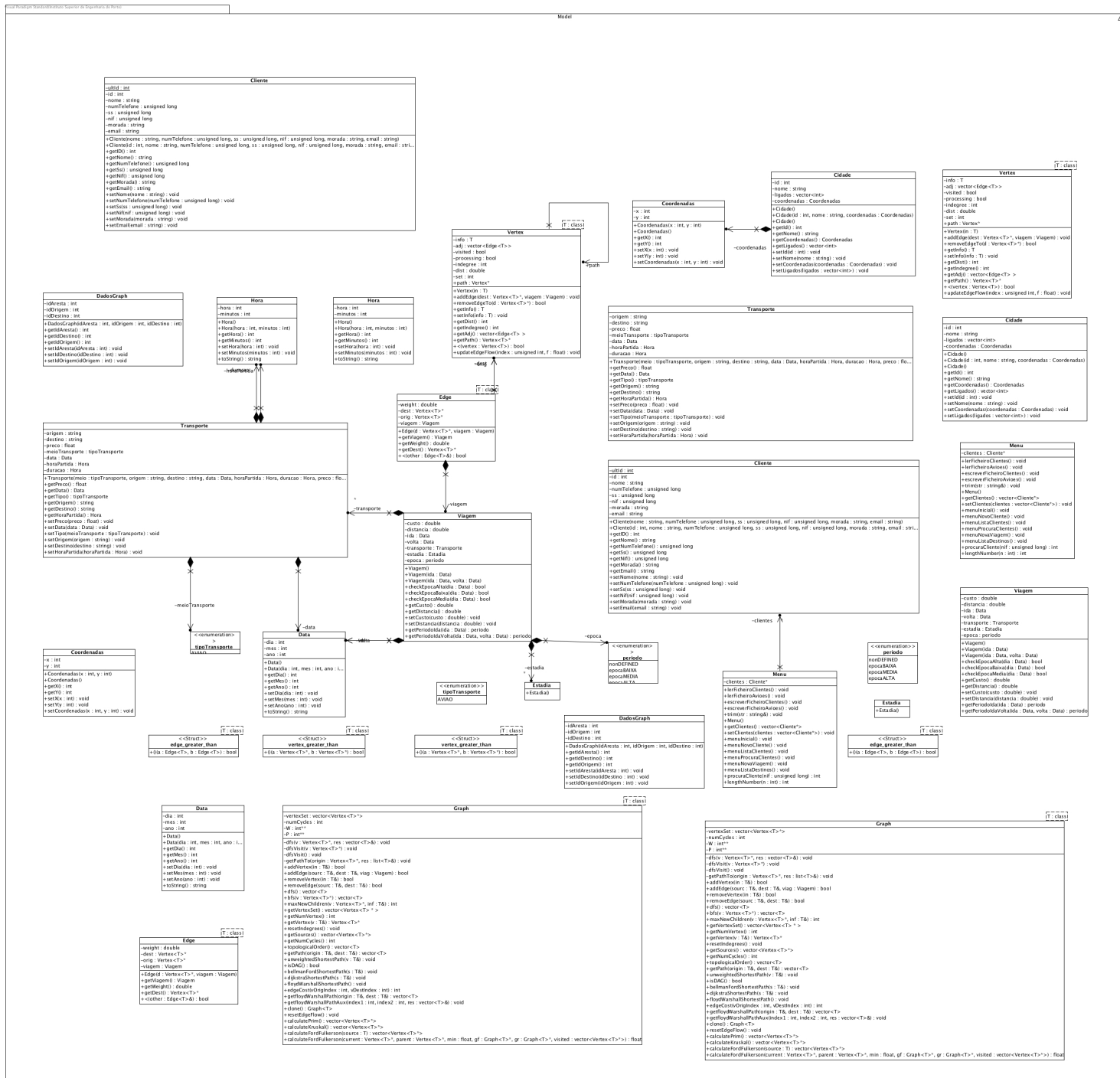
Para testar e avaliar a complexidade temporal do programa, utilizamos diferentes dados de entrada.

Complexidade Temporal

Como referido anteriormente, o **Algoritmo de Dijkstra** tem um tempo computacional de

$$O([arestas + vértices] \log (vértices))$$

Diagrama de Classes





Principais Dificuldades

Durante a realização deste projeto, deparamo-nos com alguns problemas. Um dos maiores problemas foi a introdução de dados fidedignos sobre viagens entre países. Inicialmente, íamos usar uma **API** gratuita e muito completa. Até que percebemos que fazer a leitura do ficheiro seria uma muito difícil e atrasaria em muito o projeto. Por esse motivo, decidimos usar um ficheiro de texto com toda a informação que necessitávamos para a construção do grafo.

Também tivemos alguns problemas de leitura de dados de ficheiros, devíamos às diferentes gamas implementadas nos diferentes sistemas operativos usados pelos membros do grupo.

Relativamente à parte essencial do projeto, sentimos algumas dificuldades na implementação da pesquisa em grafos.



Conclusão

A realização deste projeto desempenhou um papel crucial no auxílio à interpretação de grafos e dos algoritmos que lhes são relacionados. Nomeadamente, o ***Algoritmo de Dijkstra***.

A ponto de minimizar os erros, decidimos criar uma “Agência de Viagens” apenas para a Europa. Assim, conseguimos que os nossos dados fossem o mais fidedignos possíveis.