Relatório do projeto de Conceção e Análise de Algoritmos

**Trabalho 1**

# Tema 5: Planeamento de férias

****

**Turma 2 Grupo F:**

Joel Márcio Torres Carneiro 201100775 ei110536@fe.up.pt

Daniel Arménio Silva Mendonça 200906506

Francisco Teixeira Lopes 201106912

**Breve descrição da aplicação:**

Uma pessoa pretende planear um circuito turístico em automóvel para as férias em família em regime de *fly and drive*, visitando um conjunto de locais de interesse turístico, começando e terminando numa determinada cidade (com aeroporto próximo).

Para cada local de interesse a visitar, sabe-se a localização e a duração recomendada de cada visita (horas).  Existe também uma lista de locais possíveis de alojamento (que podem ser ou não locais de interesse). Está também definido um número máximo de horas por dia que se pretendem gastar em viagem e/ou em visita aos locais de interesse (por exemplo, 10 horas).

Pretende-se gerar um plano de viagem de duração mínima, com a sequência de locais (locais de interesse ou locais de alojamento) e dias (1º dia, 2º dia, etc.).

Numa primeira versão, determinou-se um percurso de duração total mínima, que comece e acabe na cidade especificada e passe pelo menos uma vez em cada local de interesse. Melhorou-se depois o programa para encontrar uma solução ótima de acordo com os objetivos propostos. No caso de não ser viável encontrar a solução ótima num tempo razoável, pode obter uma solução aproximada.

Formalização:

G = (V,A), V é o Set de N locais (todos os locais) a serem visitados. A é o set de caminhos possíveis de percorrer.

D = (Cij) é o custo em horas associado a cada percurso i para j. (duração)

Cv = (Cj) é o custo em horas da visita à cidade j, se esta ainda não foi visitada.

M é o número máximo de horas diárias disponíveis.

Pé o local atual.

Ud é o número de horas gastas no dia.

N é o número de dias acumulados.

R é uma cidade com aeroporto.

Xj é 1 se a cidade ainda não foi visitada e é de visita obrigatória, 0 para outros casos.

Dados de entrada:

Ca.

Cv.

G

Dados de saída:

Lista dos locais visitados

Objetivo:

Minimizar ∑(d)[ ∑(i, j) (Dij + Dj) i🡨j]

Restrições:

Assegurar que a primeira viagem é feita a partir de uma cidade com aeroporto.

(Ud=0, D=0) → Pj = R.

Assegurar que a última posição é um local com aeroporto.

∑(i, j) Xj = 0 → Pj = R.

Garantir que não são ultrapassadas o tempo máximo diário disponível.

M – Ud≥ 0.

**Descrição da solução encontrada:**

Para a resolução deste problema o grupo decidiu implementar 4 classes: City, Vertex, EdgeType, GraphViewer e Graph.

**City:**

Nesta classe a cidade é “criada”. Cada cidade tem um nome, um tempo de visita, se tem aeroporto e/ou alojamento ou não e o tempo de viagem.

É possível aceder a cada uma das especificações através de funções criadas na aplicação.

**Vertex:**

Nesta função através das propriedades do grafo, e das funções então criadas, é possível saber-se se uma cidade já foi visitada, saber qual é a cidade mais próxima e por sua vez o tempo de viagem para essa cidade.

**EdgeType:**

Nesta classe são enumerados os tipos de arestas. Arestas do tipo com e sem direção.

**GraphViewer**:

Esta classe guarda o grafo e por sua vez também o representa. Todas as suas funções retornam um booleano a indicar se a sua execução decorreu ou não com sucesso.

**Graph:**

**Diagrama UML do modelo de dados concebido**

**Lista de Casos de Utilização Identificados para a Aplicação**

1. Criar
2. Criar
3. Criar
4. Criar
5. Criar
6. Atribuir
7. Atribuir
8. Atribuir
9. Atribuir
10. Cancelar
11. Cancelar
12. Eliminar
13. Eliminar
14. Eliminar
15. Eliminar
16. Eliminar

**Principais dificuldades encontradas na resolução do trabalho**

O desenvolvimento desta aplicação trouxe algumas adversidades na escrita do código. Contudo, com alguma reflecção foram ultrapassadas, como por exemplo fazer a divisão de classes e a utilização do graph. Após a execução deste projeto ficaram mais clarificadas as noções graph, edges weight, o que nos trás vantagens para podermos trabalhar de melhor forma em qualquer projeto futuro.