ANÁLISE E SÍNTESE DE ALGORITMOS

2º PROJECTO 2016/2017
GRUPO 046
ANDREIA ROGÉRIO Nº78557

DESCRIÇÃO DO PROJETO

O trabalho realizado consistiu na construção de ligações entre cidades da forma mais barata possível, garantindo que todas as cidades estão acessíveis a partir de qualquer uma das outras. Para realizar as ligações podem ser construídas estradas entre pares de cidades ou aeroportos, individual de cada cidade com o respectivo custo associado. A diferença entre as ligações reside no facto de uma estrada permitir a ligação entre as duas cidades origem e destino, enquanto a construção de um aeroporto, nas cidades que têm essa possibilidade, torna a cidade automaticamente ligada às outras cidades possuidoras de aeroporto nesse instante, com um custo de ligação nulo, para além dos respectivos custos de construção de aeroportos (que variam de cidade para cidade).

O programa devolve não só o custo total das construções a realizar mas também o número de aeroportos e estradas construídas.

IMPLEMENTAÇÃO DO PROGRAMA

A implementação do programa foi feita na linguagem C, com recurso às bibliotecas <stdio.h>, <stdlib.h> e <limits.h>.

Para modelar o problema recorre-se a uma estrutura de grafo representado por lista de adjacências. Neste grafo os vértices são as cidades ligadas por arcos não direccionados que representam as estradas e inclui ainda um vector que guarda as identificações das cidades nas quais é possível construir um aeroporto, e ao qual apenas estas têm acesso, vector **central[]**. Existe ainda um vector auxiliar, **info[]**, que guarda as informações das cidades: custo de criar um aeroporto, **custoa**; parâmetro que descreve a cidade como visitada ou não visitada, **q**; parâmetro que descreve a cidade de acordo com a presença ou ausência de aeroporto, **a**; e identificador do vértice predecessor na arvore, **pai**.

A estratégia usada passa pela implementação do algoritmo de Prim para árvores abrangentes de menor custo, que faz uso de uma lista de prioridades com estrutura MinHeap para ordenar as cidades encontradas de acordo com o custo de construção que acarretam (**key**). O valor da key varia conforme o tipo de ligação que está a ser construída, e conforme os nós que estão a ser ligados, dependendo do custo da estrada, ou dos aeroportos caso seja necessário construí-los. As cidades serão

visitadas, e o seu parâmero **q** passará de 1 para 0, quando são extraídas da lista de prioridades.

O algoritmo foi modificado de forma a obter a solução com o menor número de aeroportos, quando varias soluções com o mesmo custo mínimo são possíveis. Para que existam várias soluções óptimas numa árvore abrangente é necessário que exista mais do que um arco com o mesmo peso. Neste caso, nem todos os caminhos são definidos por arcos, pelo que a condição aplica-se à existência de nós com o mesmo valor de key na MinHeap, ou à existência de caminhos que permitem chegar ao mesmo nó com o mesmo custo associado (mesma key). Assim sendo recorreu-se a várias estratégias:

- Criação de duas funções de relaxamento de arcos distintas, conforme se pretenda construir uma estrada ou uma ligação aérea. No caso de ser construída uma estrada a função **relaxEstrada** substitui o valor da key associada ao nó na MinHeap sempre que o valor seja igual ou inferior ao existente, substituindo desta forma possíveis ligações aéreas previamente construídas, que teriam o mesmo custo e número acrescido de aeroportos. Em oposição, a função **relaxAeroporto** só substitui caso a key seja inferior à já existente.
- Uma prioritarização extra associada à função **extractMin** que verifica se o nó a ser extraído comtempla uma ligação aérea. Em caso afirmativo verifica se os nós abaixo do nó que vais ser extraído têm o mesmo valor de key, e caso tenham e contemplem ligações por estrada então troca os nós e extrai o nó associado à estrada preferencialmente.

Foi ainda construída uma função, **reavaluate**, que têm em conta a necessária reavaliação dos custos das ligações quando uma cidade adquire um aeroporto ao ligar-se a outra. Nesses casos, as ligações aéreas que já tinham sido inseridas na lista de prioridades partindo desse nó continham custos associados à criação de um aeroporto que já foi construído, e não terá de ser contabilizado novamente.

FUNÇÃO MAIN

Inicialmente contabiliza-se nos custos o preço de aeroportos que obrigatoriamente têm de ser construídos, e contam-se os mesmos para o número total de aeroportos. Estas cidades são aquelas que têm aeroporto mas não têm estradas (estão presentes na primeira lista do inut, mas não na segunda).

A raiz para o algoritmo Prim é definida consoante os tipos de cidades do input. Caso exista alguma cidade que obrigatoriamente terá de ter um aeroporto, pois não possui estradas com nenhuma outra cidade, então essa deverá ser a raiz (caso existam mais será a primeira do int nessa condição). Desta forma, as ligações entre cidades que obrigatoriamente têm de ter aeroporto são construídas de imediato, por terem custo

zero (os custos dos aeroportos já foram anteriormente contabilizados), e o primeiro custo será respectivo ao aeroporto mais barato disponível entre as cidade que opcionalmente têm aeroporto.

No caso de não existirem ligações possíveis entre todas as cidades então existirá mais do que uma cidade sem pai, e a função **sem_pai** verifica essa ituação e devolve o output I**nsuficiente**.

De notar que os valores enviados para a estrutura de dados para os valores das arestas são u-1 e v-1. Isto deve-se ao facto de o programa esperar como input N vértices de valores de 1 a N, e a estrutura de dados interna funcionar com N vértices de valores 0 a N-1. Da mesma forma, existe um desfasamento nos índices da MinHeap pois a primeira posição desta não é utilizada. No entanto o parâmetro **v** de cada nó na MinHeap corresponde à sua posição no vector de adjacências do Grafo.

COMPLEXIDADE DO PROGRAMA

- Inicialização do grafo: O(|V|)
- Inserção das arestas: O(|E|) , neste caso referente apenas às estradas
  Subtotal: O(|V|+|E|)

- Função Prim modificada: O(|V|) -> Uma extração por vértice
  +O(|E|) -> análise das estradas é feita uma vez apenas por estrada
  *O(lgV) -> Por cada extração actualiza-se a Queue

Total: O(|V|lgV+|E|lgV)