ANÁLISE E SÍNTESE DE ALGORITMOS

Relatório do 2º Projeto ○ Grupo 133

João Pedrosa – 83485
João Pina – 85080

Introdução

Este relatório descreve uma possível solução para o problema proposto para o **2º Projeto** da cadeira de **Análise e Síntese de Algoritmos** do 2º Semestre de 2016/2017. O problema baseia-se na organização de uma rede de transportes entre cidades, usando estradas (que unem um par de cidades) e aeroportos (que ligam todas as cidades nas quais haja um).

Através do *input*, é-nos dado o número de cidades, bem como as estradas e aeroportos que são possíveis construir e o respetivo custo.

O programa deve decidir quantas e quais as ligações a construir de maneira a poder ligar todas as cidades, minimizando o custo total necessário. No final, o programa devolve o custo total, bem como o número de aeroportos e estradas a construir, ou uma mensagem de erro caso não seja possível obter a rede pretendida.

Descrição da Solução

Perante o problema apresentado, que descreve a necessidade de obter uma rede de cidades ligadas entre si, entendemos que a representação mais adequada a tomar seria a de um **grafo** (*graph*). Este grafo é <u>pesado</u> e <u>não dirigido</u> (uma vez que as estradas e aeroportos ligam as cidades nos dois sentidos). Os seus **vértices** representam as cidades e os **arcos** representam as estradas ou aeroportos que as ligam entre si. O **peso** de cada arco representa o custo de construção dessa ligação.

Para representar o **grafo** no nosso código criámos duas estruturas:

- A estrutura do **arco** (*struct edge*), que contém três inteiros (vértice de origem, vértice de destino e peso);
- A estrutura do **grafo** em si (*struct graph*), que contém dois inteiros (número de vértices e número de arcos) e um vetor de **arcos** (*struct edge*).

Facilmente se representa uma estrada como um arco, pois tem um custo associado (**peso**) e une duas cidades. Mas para representar um aeroporto optámos por criar um vértice extra (*graph->edge[0]*) que funciona como uma "central" e liga-se a todos os aeroportos com um **peso** igual ao custo de construção de cada aeroporto.

Após escolhida a representação a usar, verificámos que o resultado pretendido seria uma **árvore abrangente de menor custo (MST)** do grafo e para a obtermos utilizámos uma variação do **algoritmo de Kruskal**. Algoritmo esse que consiste nos seguintes passos:

1. Criar um vetor (*resultado1[]*) de dimensão nºcidades-1 para guardar a **MST** final;
2. Guardar todos os arcos existentes no **grafo** num mesmo sítio (*graph->edge[]*) e ordená-los por ordem crescente do seu **peso**;
3. Retirar do grafo o **arco** com menor **peso**;
4. Caso o grafo retirado conecte duas árvores distintas, é aprovado e colocado no vetor *resultado1[]*. Caso contrário (caso em que cria um ciclo) é descartado.
5. Continuar a executar os passos **4-5** enquanto não forem percorridos todos os arcos, ou enquanto não for preenchido na totalidade o vetor *resultado1[]* (aquele que se verificar primeiro).
6. Percorrer o vetor *resultado1[]* somando todos os pesos (*custo_total*) e contando o número de aeroportos (*aeroportos*) e estradas (*estradas1*). No final, apresentar estes valores.

No entanto, o nosso código apresenta duas alterações: A 1ª para detetar o caso "**Insuficiente**"; A 2ª para detetar um caso de desempate (custo total é igual se forem ou não usados aeroportos):

- "**Insuficiente**": É o obtido quando não é possível determinar uma única **MST**. Verifica-se o caso em que não é possível obter uma **MST** se forem percorridos todos os arcos (**Passo 6**) e o vetor *resultado1[]* não estiver totalmente preenchido. Neste caso é impresso para o output "Incoerente" e o programa termina.
- **Caso de desempate**: No início do algoritmo verificamos se o nº de estradas é ≥ nº de cidades-1 para determinar se é possível estabelecer esta rede sem aeroportos. Caso seja, executamos o algoritmo só para as estradas e guardamos o custo total mínimo que conseguimos obter dessa forma. Na segunda execução do algoritmo de **Kruskal** incluímos também os aeroportos e obtemos o custo mínimo possível com aeroportos e estradas.
  No final comparamos os custos das possíveis redes (se for sequer possível fazer só com estradas) e caso os custos sejam iguais, é escolhida a **MST** sem aeroportos. Desta forma, será sempre

construído o menor número possível de aeroportos, preferenciando a construção das estradas se o custo for igual.

Análise Teórica

Para analisar a complexidade do tempo de execução do programa, assuma-se a seguinte notação: **V = nº de vértices** (= nº de cidades); **E = nº de arcos** (= nº de aeroportos + nº de estradas).

A criação e inicialização do grafo deste problema tem complexidade **O(E)**. A ordenação dos arcos por ordem crescente de **peso**, é feita utilizando o *qsort()* da linguagem C, que tem complexidade $O(n \log n)$, ou neste caso, $\mathbf{O(E \log E)}$. O processo de criação e união de *sets* tem complexidade máxima de $\mathbf{O(\log V)}$ e este processo é executado uma vez por cada arco, logo tem complexidade $\mathbf{O(E \log V)}$.

Assim temos que a complexidade deste programa é dada pela soma $\mathbf{O(E + E \log E + E \log V)}$. Mas como **E** é, no máximo, igual a $\mathbf{V^2}$, temos que $\mathbf{O(E \log E)} = \mathbf{O(E \log V^2)} = \mathbf{O(E*2\log V)} = \mathbf{O(E \log V)}$.

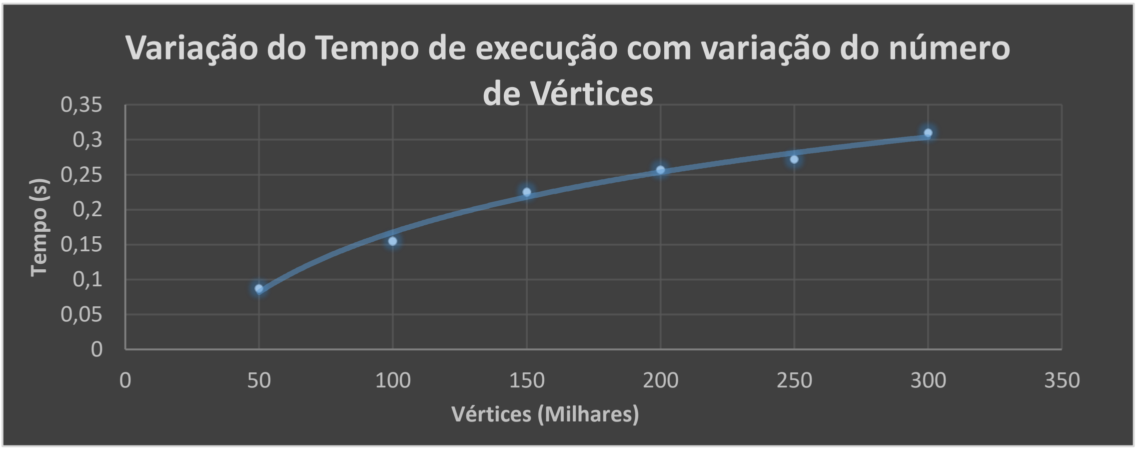
Logo, a complexidade final deste programa é $\mathbf{O(E \log V)}$.

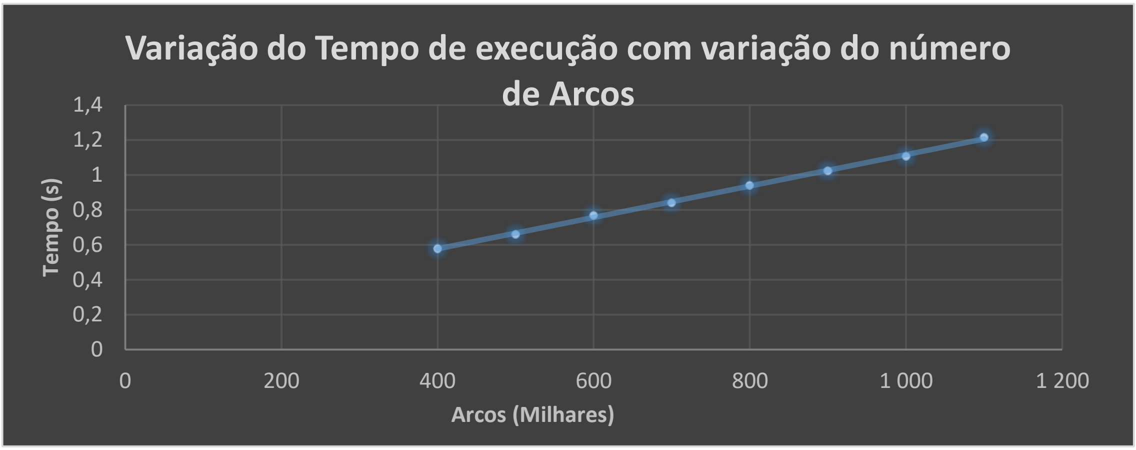
Análise Experimental dos Resultados

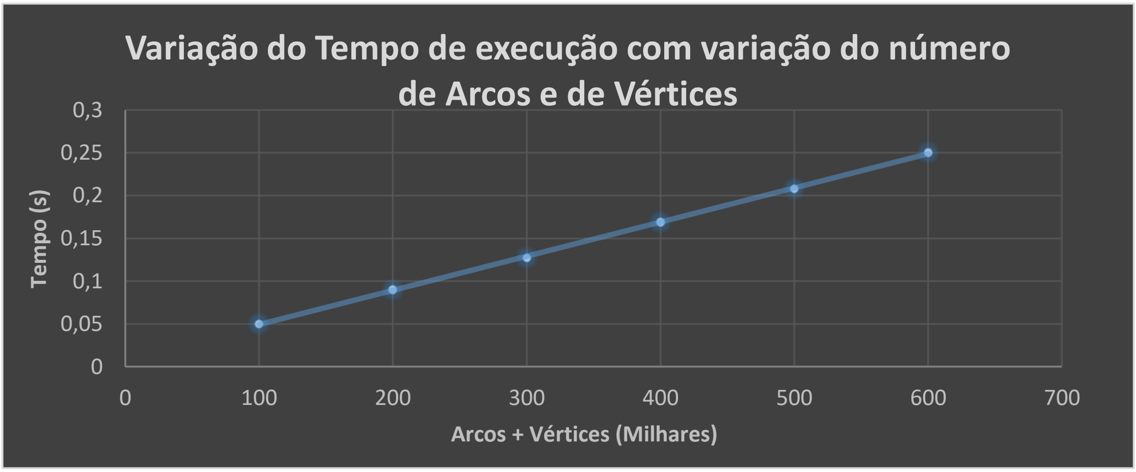
Para confirmar as conclusões da **Análise Teórica** efetuámos as seguintes medições do tempo de execução: 1º variando o número de **Vértices**, mantendo o número de arcos (500.000); 2º variando o número de **Arcos**, mantendo o número de vértices (300.000); 3º variando simultaneamente o número de **Vértices e Arcos**.

No primeiro gráfico podemos verificar o desenho de uma função logarítmica, levando-nos a concluir que o tempo varia com $\log V$. No segundo gráfico podemos verificar que o tempo de execução varia linearmente com o número de arcos (como esperado). Finalmente no terceiro gráfico podemos verificar uma função aparentemente linear, no entanto é de notar que a função $f(x) = x \log x$ (semelhante ao valor esperado) tem um crescimento inicial muito semelhante ao da função

$f(x) = x$. Se o alcance de valores testados tivesse sido maior, talvez tivesse sido possível observar um gráfico como o esperado.







Referências

- Slides das aulas teóricas de Análise e Síntese de Algoritmos;
- Slides das aulas teóricas de Introdução aos Algoritmos e Estruturas de Dados;
- *https://en.wikipedia.org/wiki/Kruskal%27s_algorithm.*