INSTITUTO SUPERIOR TÉCNICO ○ 2017

ANÁLISE E SÍNTESE DE ALGORITMOS

Relatório do 1º Projeto ○ Grupo 133

João Pedrosa – 83485
João Pina – 85080

Introdução

Este relatório descreve uma possível solução para o problema proposto para o 1º Projeto da cadeira de Análise e Síntese de Algoritmos do 2ºSemestre de 2016/2017.

O problema trata-se da ordenação de fotografias (com a atribuição de um número inteiro a cada uma delas), usando apenas como *input* o número de fotografias que queremos ordenar, o número de pares para o qual sabemos a ordem cronológica relativa de ambas as fotografias, e cada um desses pares ordenados.

A solução devolve a ordem das fotografias se for possível descobrir a mesma com os pares que são fornecidos, ou um erro caso haja alguma incoerência nos dados fornecidos ou caso não haja informação suficiente para estabelecer a ordem pela qual foram tiradas.

Descrição da Solução

Para a execução deste projeto optámos pela escolha da linguagem de programação **C**.

Perante o problema apresentado, que descreve a necessidade de ordenar fotografias meramente com base na ordem relativa entre pares delas, entendemos que a representação mais adequada a tomar seria a de um **grafo** (*graph*). Este grafo é dirigido, os seus **vértices** representam as fotografias e os **arcos** representam a precedência existente entre cada par delas.

Para representar o grafo no nosso código escolhemos criar um vetor em que cada entrada é uma lista ligada, originando assim uma **lista de adjacências**. No vetor principal existiriam tantas entradas quanto o número de vértices existentes, e a dimensão das listas ligadas seria tanta quantas as ligações de saída de um determinado vértice. Para implementar a lista de adjacências criámos a estrutura de um nódulo

(*struct vertex*) que possui o valor de um vértice a apontar (*value*) bem como um apontador para o próximo nódulo (*next*).

Após escolhida a representação a usar, verificámos que o resultado pretendido seria uma **ordenação topológica** do grafo e para a obtermos utilizámos uma variação do **algoritmo de Eliminação de Vértices**. Algoritmo esse que consiste nos seguintes passos:

1. Para além do grafo, ter dois vetores: Um para ir colocando os vértices sem arcos de entrada (*queue[]*) e outro para colocar os vértices topologicamente ordenados (*resultado[]*);
2. Colocar todos os vértices sem arcos de entrada no *queue[]*;
3. Retirar o primeiro vértice do *queue[]* para o *resultado[]*;
4. Eliminar todos os arcos partindo desse vértice para outros;
5. Verificar se algum dos vértices de destino dos arcos removidos não possui mais nenhum arco de entrada. Caso isto se verifique, colocar esse vértice no *queue[]*.
6. Repetir os passos 3-5 até que o *queue[]* fique vazia;
7. Devolver a ordenação topológica dos vértices, presente no vetor *resultado[]*.

Para facilitar a execução deste algoritmo criámos também um vetor (*entradas[]*) que é criado em simultaneidade com o grafo, tem dimensão igual ao número de vértices e guarda, para cada um deles, o número de arcos de entrada que possuem. Sempre que é eliminado um arco entre dois vértices, decrementa-se o valor presente no vetor *entradas[]* correspondente ao segundo vértice. Desta maneira é mais fácil para o programa encontrar os vértices sem arcos de entrada (Passo 2).

No entanto, o nosso código nem sempre segue o algoritmo à letra. Apresenta, nomeadamente, duas alterações: A primeira para detetar o caso "**Incoerente**"; A segunda para detetar o caso "**Insuficiente**".

1. "**Incoerente**" é o output a apresentar quando há "relações inconsistentes", segundo o enunciado. Aplicando este conceito a grafos, um resultado dir-se-á incoerente se existir algum ciclo fechado em qualquer parte do grafo, pois tal faz com que não se possa obter uma ordenação topológica verdadeira. Para detetar este problema, criámos uma variável (*inc*), inicializada a 0, de maneira a que sempre que é adicionado um vértice à *queue[]*, essa variável é incrementada. No final do **Passo 6**, é verificado se a *inc* tem a mesma dimensão que o número de vértices existentes. Caso esse não seja o caso, indica que, em determinada altura, o grafo não teve mais vértices sem arcos de entrada, logo a informação é incoerente.

2. "**Insuficiente**" é o output esperado quando não é "possível determinar uma única organização *(…)* porque a informação é insuficiente", segundo o enunciado. Em termos de grafos, isto significa que em determinado instante da ordenação topológica, existiriam simultaneamente dois vértices sem arcos de entrada, não sendo possível saber qual deles o primeiro. Para registar esta insuficiência de informação reservámos a última posição do vetor final (*resultado[-1]*), para assinalar caso tal se verifique. Após cada execução do **Passo 5**, verifica-se a dimensão da *queue[]* e, se esta for superior a 1, é colocado o valor `-1` em *resultado[-1]*. Desta maneira é possível concluir que existiu mais do que um vértice sem arcos de entrada, logo a informação é insuficiente.
   É de notar que esta verificação só é efetuada após a do caso "Incoerente", de maneira a respeitar os requisitos do enunciado.

Análise Teórica

A criação do grafo tem complexidade **O(V)** e a inicialização do mesmo (guardando o número de entradas de cada vértice num vetor *entradas[]*) tem complexidade **O(E)** pelo que a complexidade total da inicialização (realizada na função *createGraph()*) das estruturas de dados necessárias é **O(V+E)** , sendo linear com o crescimento do número de vértices (fotografias) e também com o de arcos (nº de pares ordenados) inseridos no *input*.

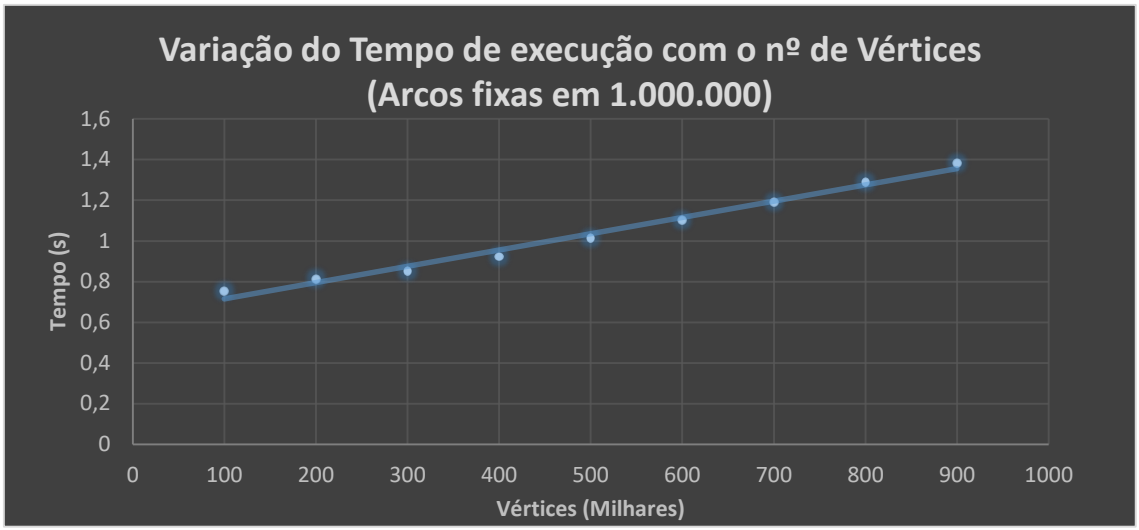
A inserção de um novo vértice no grafo é **O(E)** já que no pior caso é a última adjacência de um vértice que já esteja ligado a todos os outros vértices.

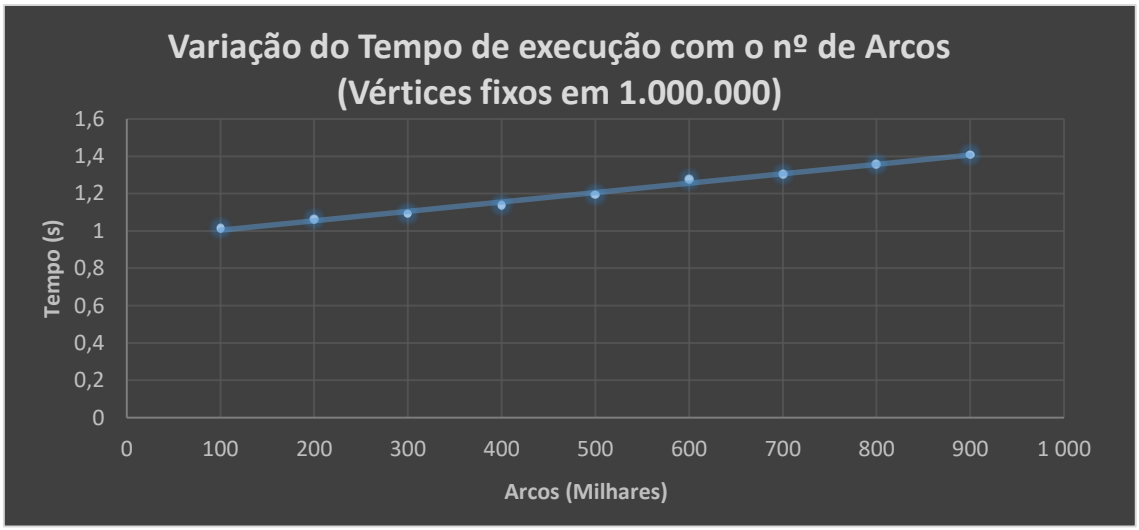
O algoritmo de ordenação de vértices é baseado na ordenação topológica que nos foi apresentada nas aulas teóricas da cadeira, que no pior caso percorre todos os vértices e todos os arcos pelo que a sua complexidade teórica é **O(V+E)**.

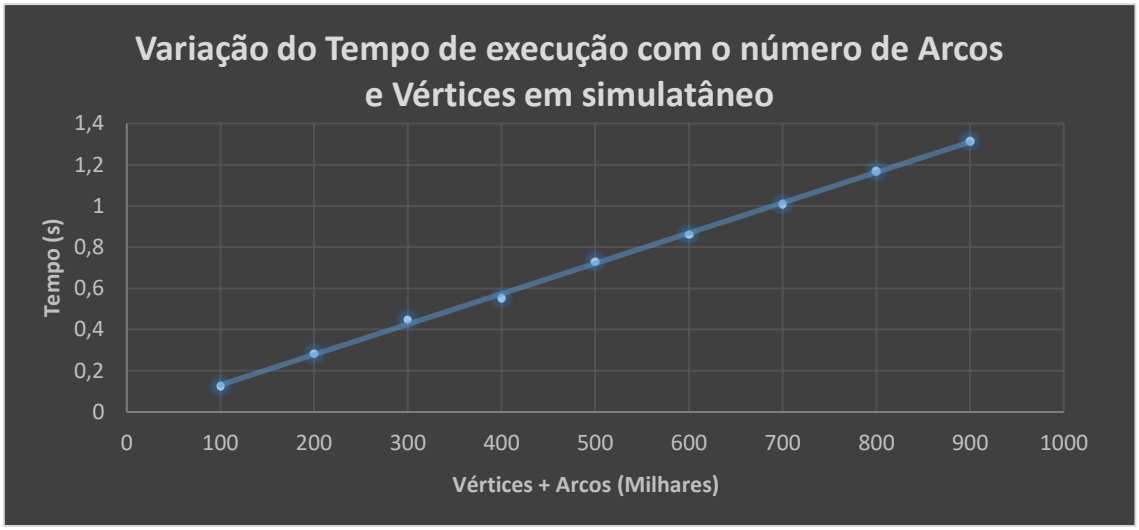
Análise Experimental dos Resultados

Para confirmar as conclusões da **Análise Teórica** efetuámos várias medições do tempo de execução do programa, variando o número de **Vértices**, de **Arcos** e por fim simultaneamente **Vértices e Arcos**.

Dos primeiros dois gráficos podemos concluir que o tempo varia linearmente, tanto com o número de vértices como com o número de arcos. Finalmente, o terceiro gráfico permite-nos verificar a o que tínhamos concluído previamente: A complexidade é **O(V+E)**.







Referências

- Slides das aulas teóricas de Análise e Síntese de Algoritmos