Análise e Síntese de Algoritmos

1º Projeto - 24 de março de 2017

77906 António Sarmento 84711 Diogo Redin

Introdução

Neste projeto foi-nos pedido que criássemos um programa que ordenasse cronologicamente um dado número de fotografias, sendo dadas as suficientes relações cronológicas entre as mesmas. O objetivo do projeto consistiu em escolher uma estrutura de dados adequada e um algoritmo que nos permitisse ordenar as fotografias, sempre que possível.

Para representar as relações entre as fotografias, decidimos utilizar um grafo dirigido, em que cada vértice representa uma fotografia e cada arco consiste numa relação cronológica entre as mesmas (i.e., se a fotografia A acontece antes que a fotografia B, no grafo representamos a relação como A -> B). Com esta implementação, encontrar uma ordenação cronológica das fotografias, traduz-se em encontrar uma ordenação topológica do grafo.

Os principais desafios deste problema consistiram em saber quando as relações dadas entre as fotografias eram insuficientes ou incoerentes.

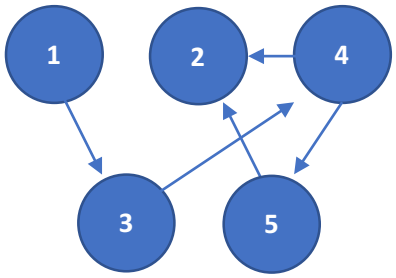
Análise Teórica

Como estrutura de dados para representar o grafo dirigido, escolhemos um vetor de arcos, recorrendo a três vetores:

1. Tamanho: No. de vértices (V); Índice: Número do vértice; Conteúdo: Índice do arco;
2. Tamanho: No. de arcos (E); Índice: Número do arco; Conteúdo: Vértice Final;
3. Tamanho: No. de arcos (E); Índice: Número do arco; Conteúdo: Arco Irmão;

Recorremos ainda a dois vetores adicionais de tamanho igual ao número de vértices para respetivamente guardar o grau de cada vértice e o resultado da ordenação topológica.

Exemplo da representação de um Grafo Dirigido:



#	Vértice
1	1
2	0
3	2
4	3
5	5

#	Arco	Irmão
1	3	-
2	4	-
3	2	4
4	5	-
5	2	-

#	Grau
1	0
2	2
3	1
4	1
5	1

A nossa estrutura de dados tem, portanto, a seguinte eficiência para cada operação:

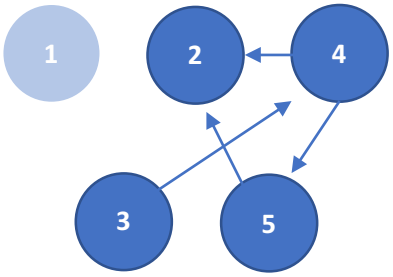
- Inicialização: O(1)
- Inserir Arco: O(1)
- Encontrar Arco: O(E)

Para fazer a ordenação topológica do grafo, implementámos o algoritmo de Khan. Este algoritmo consiste em:

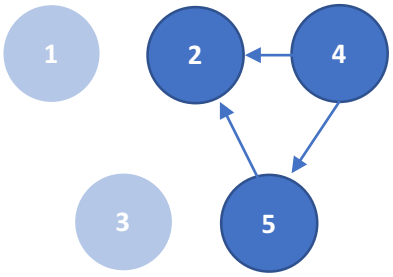
1. Colocar todos os vértices sem arcos de chegada (grau 0) numa pilha;
2. Criar um contador de vértices visitados que começa a 0;
3. Remover um dos vértices da pilha e:
    a. Guardar o vértice no vetor de resultados;
    b. Incrementar em 1 o contador dos vértices visitados;
    c. Decrementar em 1 o grau de todos os vértices filhos;
    d. Se o grau de algum dos vértices filhos passar a ser 0 adicioná-lo à pilha;
4. Repetir o passo 3 até não haver vértices com grau 0 na pilha;

Exemplo da aplicação do algoritmo de Khan num Grafo Dirigido:

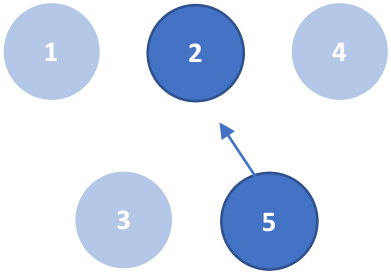
1º



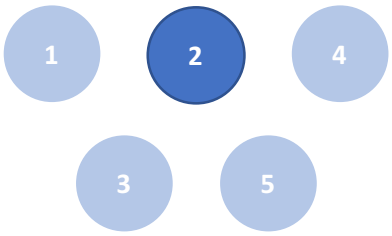
2º



3º



4º



Resultado: <1,3,4,5,2>

Output: “Insuficiente”

Sabemos que não há relações suficientes para haver uma única ordenação topológica quando na execução do passo 3.c. um vértice tem mais do que um vértice filho com grau 0.

Output: “Incoerente”

Sabemos que há relações inconsistentes quando o número de vértices no resultado é diferente do número de vértices dados, o que aconteceria se algum vértice aparecesse mais do que uma vez na ordenação, o que não é possível.

O algoritmo tem eficiência O(V + E) uma vez que percorremos todos os vértices uma vez para descobrir quais têm grau 0 (1.), e posteriormente percorremos todos os arcos existentes (3.d.).

Em termos de espaço, o nosso vetor de arcos recorre aos seguintes valores:

- V para os vértices e 2E para os arcos = V + 2E
- V para os graus
- V para a lista ordenada de vértices

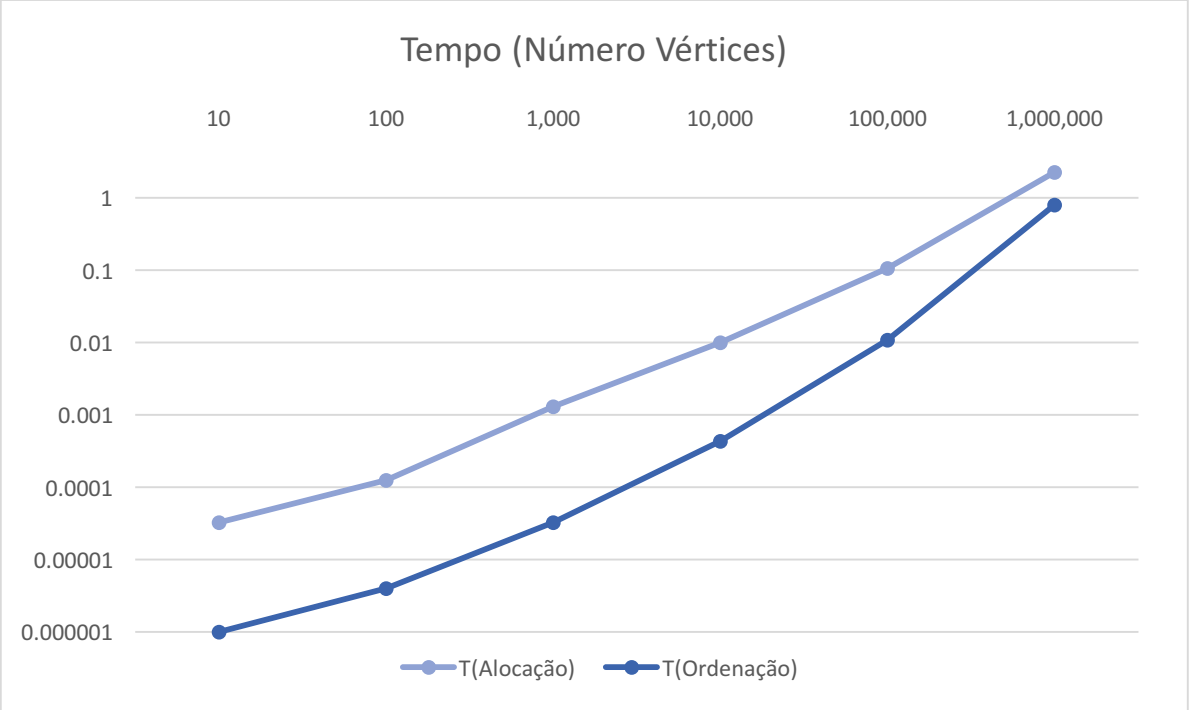
Em suma, temos uma complexidade de O(3V + 2E) = O(V + E) de espaço.

Implementação e Resultados

Implementámos o nosso programa em C por maior familiaridade com a linguagem e porque era suficiente para a estrutura de dados que queríamos implementar. No sistema Mooshak passámos a todos testes, tendo obtido 16 valores.

Apresentamos de seguida cinco testes automaticamente gerados pelo programa fornecido na página da cadeira e aplicado ao nosso programa:

V	E	V + E	T(Alocação)	T(Ordenação)	T(Total)
10	16	26	0.000033	0.000001	0.000034
100	196	296	0.000126	0.000004	0.000130
1 000	1995	2995	0.001317	0.000033	0.001350
10 000	19 995	29 995	0.010111	0.000437	0.010548
100 000	199 996	299 996	0.106840	0.010936	0.117776
1 000 000	2 999 993	3 999 993	2.268193	0.809921	3.078114



Nota: Gráfico apresentado com escala algorítmica.

Como podemos observar, o tempo demorado para criar o grafo é bastante superior ao tempo gasto para encontrar uma ordenação topológica. Podemos também observar que os tempos obtidos experimentalmente formam uma reta que corresponde à complexidade teórica esperada de O(V+E).
Se formos mais específicos, a complexidade temporal de acordo com o nosso código é baseado no seguinte:

- Alocação de memória das estruturas de dados (juntamente com a inicialização a 0 a alguns): O(2V + E + 2)
- Ligação de vértices através dos arcos: O(E)
- Colocação de todos os vértices sem arcos de chegada (grau 0) na pilha: O(V)
- Ordenação topológica do Grafo a partir do algoritmo de Kahn: O(V + E)
- Escrita do resultado caso haja apenas uma ordenação possível: O (V)

Total: O(5V + 3E + 2) = O(V + E)

Referências

- Introduction to Algorithms (3rd ed.), MIT Press and McGraw-Hill, ISBN 0-262-03293-7.

- Algoritmos em Grafos: Pesquisa e Ordenação Topológica, R. Rossetti, A.P. Rocha, A. Pereira, P.B. Silva, T. Fernandes CAL, MIEIC, FEUP http://paginas.fe.up.pt/~rossetti/rrwiki/lib/exe/fetch.php?media=teaching:1011:cal:05_1.grafos1_b.pdf

- Grafos: Slides Introdução Algoritmos e Estruturas de Dados, Profº Francisco Santos IST https://fenix.tecnico.ulisboa.pt/downloadFile/1689468335557962/iaed2015_16-2s-aula25-Grafos.pdf