Análise e Síntese de Algoritmos

2016/2017

Relatório do Primeiro Projeto

Grupo 127

81186 - Stéphane Duarte

INTRODUÇÃO

No âmbito da cadeira de Análise e Síntese de Algoritmos, foi-nos proposto um projeto cujo objetivo era desenvolver um sistema que ajudasse a organizar fotografias pela sua ordem cronológica. O utilizador numera as fotografias e introduz no sistema números aos pares, em que o primeiro representa a foto mais antiga do par e o segundo a foto mais recente.

Deste modo, o problema vai ser encarado como um grafo dirigido no qual terá de ser aplicado um algoritmo de procura, recorrendo a uma adaptação do algoritmo DFS (procura em profundidade).

DESCRIÇÃO DA SOLUÇÃO

Este problema é encarado como um grafo dirigido, em que cada vértice representa uma fotografia e cada aresta representa uma ligação entre fotografias (ligação u->v, em que u procede v na ordem cronológica).

A solução foi implementada em Java para facilitar a implementação de ADTs, usando as bibliotecas List, ArrayList, LinkedList e Exception.

A representação do grafo foi feita recorrendo a um grafo com listas de adjacências.

Como input, é recebido:

1. Uma linha com dois valores:
    a. O primeiro valor corresponde ao número de fotografias (N), ou seja, o número de vértices do grafo. Este valor é utilizado para a criação do grafo e para a criação das listas necessárias.
    b. O segundo valor corresponde ao número de pares (L) existentes entre as fotografias. Este valor vai ser utilizado no ponto 2.
2. L linhas com dois elementos numéricos u e v entre 1 e N, representando assim a ligação de u a v, em que u procede v na ordem cronológica.

Posteriormente, é criada a lista onde vão ser guardados os vértices por ordem inversa à cronológica e outra lista onde os vértices vão ser marcados como não visitados, abertos ou visitados. Um vértice não visitado é um vértice que ainda não foi chamado pelo algoritmo. Um vértice aberto é um vértice que já foi chamado pelo algoritmo e este continua a chamar os seus vértices adjacentes. Um vértice visitado é um vértice que já concluiu o processo todo.

O programa começa então a correr o algoritmo de ordenação topológica, percorrendo, por ordem, cada vértice não visitado. Tal como na DFS, o algoritmo vai visitar os filhos do vértice até encontrar um que não tenha filhos. Isto, no contexto do problema, significa que a foto é a mais recente, logo, não tem fotografias posteriores, não tendo vértices adjacentes. Assim que é encontrado um vértice deste género, é adicionado à lista que guarda a ordenação. Quando já todos os vértices adjacentes foram visitados, o vértice é também adicionado à lista e marcado como visitado. O algoritmo corre até todos os vértices estarem visitados.

Há ainda dois problemas a ter em consideração: quando existe um ciclo (ou seja, o utilizador enganou-se a colocar um ou mais pares) ou quando existem várias hipóteses de ordenação. Para lançar a exceção de ciclo, basta verificar se o vértice que está a ser tratado não encontra um filho aberto. Se encontrar, significa que estamos perante um ciclo. Para verificar se existe apenas uma ordem, é chamada, antes da inserção de um vértice na lista ordenada, uma função que verifica se o vértice a inserir tem uma ligação com o último vértice inserido. Isto assegura que existe apenas uma ordem possível.

Como output, obtém-se uma linha com o número das fotografias ordenadas por ordem cronológica. No caso de existir um ciclo, o output gerado será “incoerente”, e no caso de haver mais do que uma alternativa, o output gerado será “insuficiente”.

ANÁLISE TEÓRICA

Após cuidada análise ao código, é possível determinar o tempo de execução de cada ciclo e, consequentemente, do programa. Considere-se N o número de vértices do grafo e L o número de arestas.

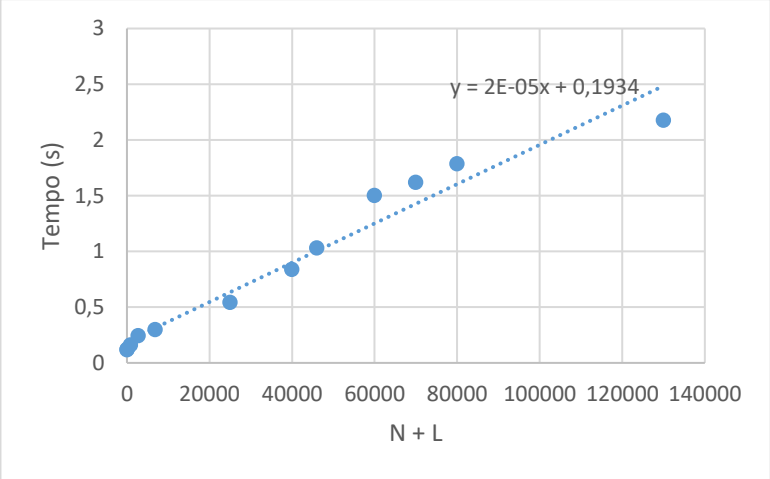
1. Inicialização do grafo: O(N);
2. Ciclo para ler as relações: O(L);
3. Ciclo para inicializar a lista “visited”: O(N);
4. Algoritmo de Ordenação Topológica: O(N+L);
5. Ciclo para imprimir a ordenação: O(N).

Tendo em conta as complexidades apresentadas anteriormente, concluímos que a complexidade total do programa será: N + L + N + (N+L) + N = O(N+L).

AVALIAÇÃO EXPERIMENTAL DOS RESULTADOS

Casos de teste:

N	L	Tempo (s)
10	15	0.119
20	46	0.126
200	689	0.162
400	2363	0.244
500	6325	0.300
5000	19986	0.545
10000	29987	0.839
13000	32992	1.033
20000	39992	1.503
25000	44994	1.622
30000	49998	1.784
40000	89995	2.178



Tal como visto na análise teórica, a complexidade do programa é O(N+L). Foram então realizados testes experimentais para comprovar este valor.

Após 12 testes, foi verificado que com o aumento da soma vértices+arestas, o tempo de execução aumentava também linearmente, como se pode ver no gráfico acima.

Deste modo, verifica-se que a complexidade do programa é, de facto, O(N+L).

REFERÊNCIAS

https://en.wikipedia.org/wiki/Topological_sorting

http://www.geeksforgeeks.org/topological-sorting/

https://courses.cs.washington.edu/courses/cse373/02au/lectures/lecture19l.pdf

Introduction to Algorithms, Third Edition: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein September 2009