Instituto Superior Técnico

Análise e Síntese de Algoritmos – 2º Semestre – 2016/2017

Relatório do primeiro projeto – Grupo18T – Francisco Barros 85069 – Rafael Ribeiro 84758

Introdução

O projeto teve como objetivo organizar elementos, fotografias, de um conjunto tendo em conta as relações, cronológicas, de cada par. O desafio encontrado foi verificar que a ordenação obtida era única, se ela fosse possível. A solução proposta é funcional, tendo passado a todos os testes efetuados pelo sistema de avaliação *Mooshak*, refletindo-se na avaliação 16/16 na parte prática.

Descrição da solução

A solução proposta foi escrita em C++, devido às suas classes otimizadas e de utilização prática. Para manter a ordenação e todas as relações, escolheu-se um grafo que se traduz (para código) em:

- Dois inteiros que representam o número de vértices (fotografias) e o número de arestas (relações);
- Dois *arrays* de inteiros. Um deles mantém o estado de cada vértice representado pelas cores branco (não visitado), cinzento (em visita) e preto (visitado). O outro mantém a ordenação topológica do conjunto;
- Dois valores booleanos que representam se o *input* é incoerente ou insuficiente;
- Um *array* de *list<int>* que forma a relação de todos os vértices. Cada *list* representa as ligações de saída de um vértice. A *array* possui um *list* para cada vértice;

Para fornecer um valor de *output* (“Incoerente”, “Insuficiente” ou a ordenação dos elementos) é realizada uma DFS (*Depth First Search*) para se encontrar uma ordenação topológica dos elementos. Durante esta pesquisa é também realizado um teste para verificação de ciclos. Um ciclo é encontrado na situação em que um elemento tem um descendente que também é um dos seus ascendentes. Se durante a pesquisa em

profundidade for encontrado um ciclo a pesquisa é interrompida e é impresso no *stdout* "Incoerente". Se a DFS for realizada com sucesso é garantido que o grafo gerado pelos inputs já não é incoerente. Seguidamente, verifica-se se a ordenação topológica gerada pela DFS é uma solução única, o que poderá fazer com o grafo seja classificado como suficiente ou insuficiente. Se a solução não é única é impresso no *stdout* "Insuficiente", significando que não são conhecidas uma ou mais relações entre os elementos fornecidos. Se a solução é única então os elementos do *array* que mantém a ordenação topológica são impressos no *stdout*. Esta verificação é feita verificando se a ordenação topológica é um caminho Hamiltoniano[4].

Análise Teórica

Para uma ordenação preliminar dos elementos foi utlizada um DFS a partir do vértice inicial (no problema em questão - a fotografia mais recente), todos os vértices até ao último (a fotografia mais velha).

Este algoritmo funciona porque é feito um ciclo (*DFS*)[1.2] sobre todos os vértices p_k guardados na estrutura, no qual se invoca a visita (*DFSVisit*) [1.2] ao vértice se e só se este ainda não tiver sido visitado, isto é, se estiver marcado a branco. Neste momento o vértice p, que estava marcada a branco, sobre o qual se invocou a DFSVisit é marcado como descoberto (cinzento). De seguida para cada vértice i, adjacente ao vértice atualmente em visita p, que podem ser acedidos através da lista ligada na posição p-1 do *array* que guarda as arestas de partida de cada vértice p, é chamada também a função de visita sobre esse vértice i. Quando todos os vértices i, que são adjacentes a p tiverem sido visitados, então p é marcado como finalizado (preto) e colocado no *array* de ordenação topológica, na primeira posição livre, a partir do fim do *array* para o princípio, isto porque sendo o algoritmo feito recursivamente, as primeiras visitas a entrar na *stack* são as últimas a sair[1.5].

Se um vértice i ao ser visitado a partir de um vértice qualquer p, já estiver marcado como cinzento, no momento em que é visitado, significa que estamos perante um ciclo pelo que sabemos que existem arcos-para-trás no grafo e como consequência o grafo é incoerente.[1.3]

Desta forma, numa situação em que não existem ciclos, recursivamente, todos os vértices do grafo são visitados uma e uma só vez.

A conjunção das funções DFS e DFSVisit, corre com limite assintótico superior $O(|V|+|E|)^{1.6}$, visto que ao fazermos um ciclo sobre os V vértices, na função DFS, e ao visitar todos as suas arestas, nunca iremos percorrer mais do que as E arestas do grafo.

Na posse de uma ordenação topológica gerada pelo algoritmo DFS, a verificação da solução ser única passa por verificar cada vértices na posição n, dessa ordenação, existe uma ligação direta para o vértice $n+1$[5]. Esta verificação é feita com limite assintótico superior de $O(|V-1|)$.

Avaliação experimental dos resultados

Para conferir que a análise teórica por nós escrita, acima, baseada nas várias referências bibliográficas consultadas, testamos o nosso programa, com inputs válidos, isto é, coerentes e suficientes, com um número de vértices com cardinal pertencente ao conjunto {10, 100, 1000, 100000, 1000000} e colocamos os tempos de execução no gráfico abaixo. Que tal como se pode observar, corre em tempo linear.

Para calcular o tempo de execução utilizou-se o método *high_resolution_clock* da biblioteca *chrono* de C++.

Referências Bibliográficas

1. Introduction to Algorithms – Thomas Cormen et al. – 3rd ed;

   1.1 – Depth-First search – Color Marking – 3rd paragraph, page 603.

   1.2 – DFS Algorithm Pseudo-code – Figure 22.2 – page 604.

   1.3 – Classification of Edges, 2nd point in list, page 609.

   1.4 – Topological Sort – 1st paragraph, page 612.

   1.5 – Topological Sort – 10th Line, page 613.

   1.6 – Topological Sort – Time Complexity – 1st paragraph page 606 & Lines 16 to 18 page 613.

2. Introdução aos Algoritmos e Estruturas de Dados (*Course Slides, 25th lesson, page 86 – As distributed by course principal professor).* At:

   https://fenix.tecnico.ulisboa.pt/downloadFile/1689468335557962/iaed2015_16-2s-aula25-Grafos.pdf

3. Topological Sorting – Wikipedia – Last verified to be online on 20/03/2017 – At: https://en.wikipedia.org/wiki/Topological_sorting#Complexity

4. Hamiltonian Path – Wikipedia – Last verified to be online on 20/03/2017 – At: https://en.wikipedia.org/wiki/Hamiltonian_path

5. Algorithms – Chapter 4.2 – "*Directed Graphs – Creative Problems – Exercise* 33" - Robert Sedgewick et al. – 4th ed. – Online version at: http://algs4.cs.princeton.edu/42digraph/

6. CPlusPlus – *high_resolution_clock method reference*: http://www.cplusplus.com/reference/chrono/high_resolution_clock/now/