Relatório do 1º projeto de ASA

1. Introdução

O problema proposto pode ser expresso através de um grafo, em que os vértices representam as fotos, e as arestas relações temporais entre elas. Uma aresta do vértice U para o vértice V, indica que a foto n.º U foi tirada antes da n.º V. Os objetivos são:

1. Averiguar se o grafo tem ciclos (caso tenha, o input é incoerente);
2. Averiguar se o grafo permite mais de uma ordenação topológica (caso permita, o input é insuficiente);
3. Apresentar os valores dos vértices ordenados topologicamente.

2. Descrição da solução

2.1 Averiguar se o grafo tem ciclos

No problema proposto, o input deve ser considerado incoerente se forem detetadas relações inconsistentes, i. e., se for afirmado, direta [fig.1] ou indiretamente [fig. 2], que uma dada foto A foi tirada antes de B, mas também que B foi tirada antes de A. Qualquer um dos casos implica a existência de um ciclo, na representação em grafo.

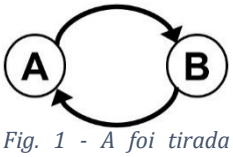


Fig. 1 - A foi tirada antes de B e B foi tirada antes de A

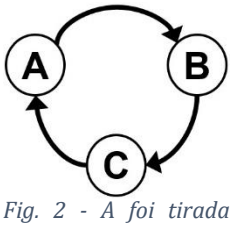


Fig. 2 - A foi tirada antes de B, que foi tirada antes de C, que foi tirada antes de A

Se, ao executar o algoritmo de procura em profundidade primeiro (DFS), for encontrado um arco de entrada para um vértice que já foi descoberto mas ainda não foi fechado, existe um ciclo, pelo que a procura para e o input é declarado incoerente.

2.2 Averiguar se o grafo permite mais que uma ordenação topológica

Pretende-se ordenar as fotos cronologicamente. Ou seja, o objetivo é obter uma ordenação dos vértices do grafo onde, existindo uma aresta de U para V, U vem obrigatoriamente antes de V. Isto corresponde a uma ordenação topológica. O input deve ser considerado insuficiente se houver mais do que uma ordenação topológica.

Durante a DFS, os vértices "fechados" vão sendo colocados numa lista, ordenada pelo tempo de fecho decrescente.

Se a DFS terminar (o input não é incoerente), obtém-se uma ordenação topológica dos vértices. Depois, verifica-se que de cada vértice sai um arco para o que se lhe segue na ordenação topológica. Se este teste falhar para algum vértice, o input é declarado insuficiente.

Esta verificação não é feita durante a DFS porque foi estipulado que se o input for simultaneamente incoerente e insuficiente deve ser apresentado o output do caso incoerente. Isto é, não se poderia terminar a DFS mais cedo, pelo que não traz vantagem fazer a verificação durante o ciclo.

2.3 Apresentar os valores dos vértices ordenados topologicamente

Percorrer a lista de vértices ordenados topologicamente obtida anteriormente, e imprimi-los no output, de acordo com a formatação indicada.

3. Análise teórica

3.1 Algoritmo de deteção de ciclos

A deteção de ciclos num grafo dirigido corresponde à deteção de arcos para trás na árvore DFS, ou seja, arcos que entram num predecessor do vértice do qual saiem. Quando é encontrado um arco para um vértice que já foi visitado previamente na DFS, está-se na presença de uma de duas situações: ou foi encontrado um arco para trás (o vértice onde entra é predecessor do atual no mesmo ramo da árvore DFS), ou um arco de cruzamento (o vértice onde entra pertence a outra árvore ou subárvore DFS). Na procura DFS, isto deduz-se a partir estado do vértice em que o arco entra: se este for preto (i. e., já foi fechado), trata-se de um arco de cruzamento; se, por outro lado, for cinzento (ainda não foi fechado), é um arco para trás. Este passo tem a complexidade de uma DFS – O(V+E).

3.2 Algoritmo de verificação da unicidade da ordenação topológica

A verificação da unicidade da ordenação topológica faz-se testando, para cada vértice, se dele sai um arco para o vértice que o sucede na ordenação obtida. Se numa dada ordenação não existisse um arco entre dois vértices consecutivos, estes poderiam ser trocados, gerando-se outra ordenação topológica possível [fig. 3]. Do mesmo modo, se nesta ordenação topológica o vértice U precede V, e noutra V precede U, pode concluir-se que não há nenhum arco de U para V (nem de V para U). Como, no pior caso, são percorridas todas as arestas, este passo tem complexidade O(E).

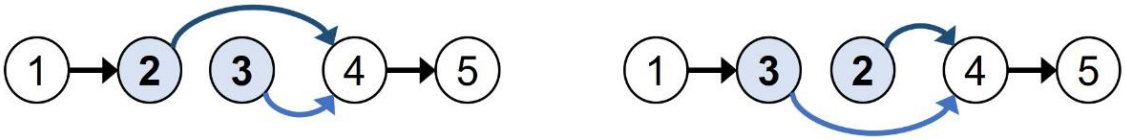


Fig. 3

3.3 Complexidade do algoritmo

Quando o input é incoerente ou insuficiente, o algoritmo é interrompido. O pior caso ocorre quando o input é válido.

1. Inicialização da lista de adjacências O(V)
2. Leitura e inserção das arestas na lista O(E)
3. Ordenação topológica (utilizando a DFS) O(V+E)
4. Verificação da unicidade da ordenação topológica O(E)

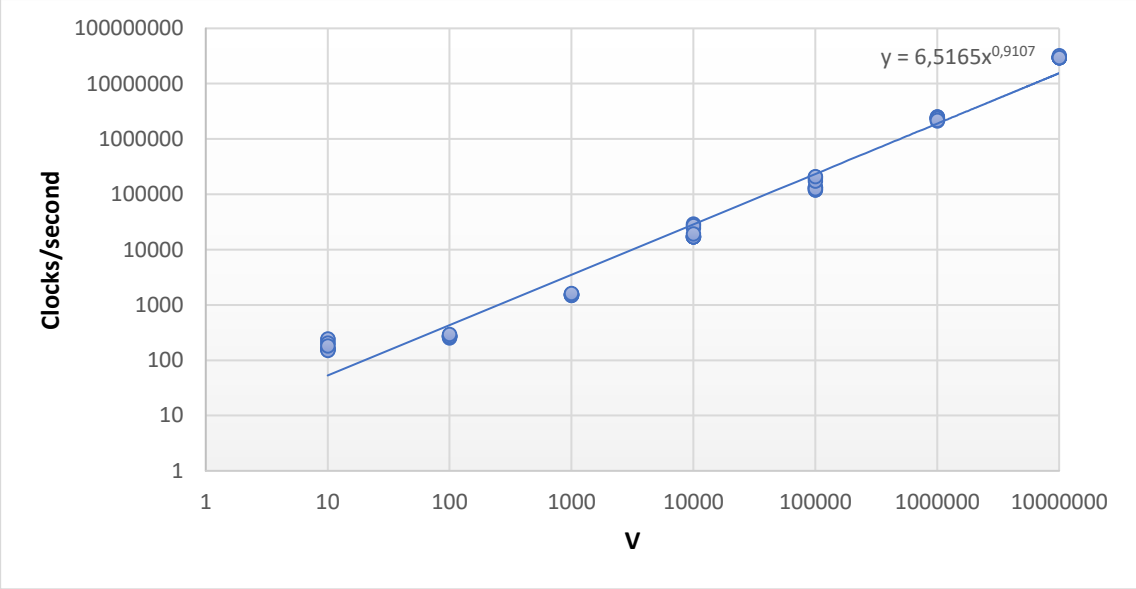
Conclusão: o algoritmo tem complexidade O(V+E).

4. Avaliação experimental dos dados

Foram medidos (em ciclos de relógio do processador) os tempos de execução do programa para inputs de tamanhos diferentes:

- V = {10, 100, 1000, 10 000, 100 000, 1000 000, 10 000 000}
- $V-1 \leq E < 2 \cdot V$

Os resultados são apresentados no gráfico seguinte:



Pode observar-se, a partir da regressão apresentada no gráfico, que o tempo de execução cresce linearmente com o tamanho do input, conforme previsto na secção 3.3.