**Análise e Síntese de Algoritmos**

**Projeto 1 - Grupo 45**

## **Introdução**

O presente relatório tem como principal objetivo demonstrar a fundamentação da solução concebida, para o problema proposto.

Este, consiste em ordenar cronologicamente um número finito de fotografias, o mais eficientemente possível, comparando-as duas a duas. Dependendo do *input* recebido, há três *outputs* possíveis:

* No caso de existir apenas uma ordenação possível, apresenta-se uma lista de números, correspondendo à ordenação cronológica das fotografias;
* Caso existam diversas ordenações possíveis, é devolvida a mensagem “**Insuficiente**”;
* Se houver relações inconsistentes que impossibilitam a formação de uma ordenação, é devolvida a mensagem “**Incoerente**” (caso os últimos dois pontos ocorram simultaneamente, a mensagem a apresentar é “**Incoerente**”).

Para resolver o enunciado problema, foi desenvolvido um programa na **linguagem C**.

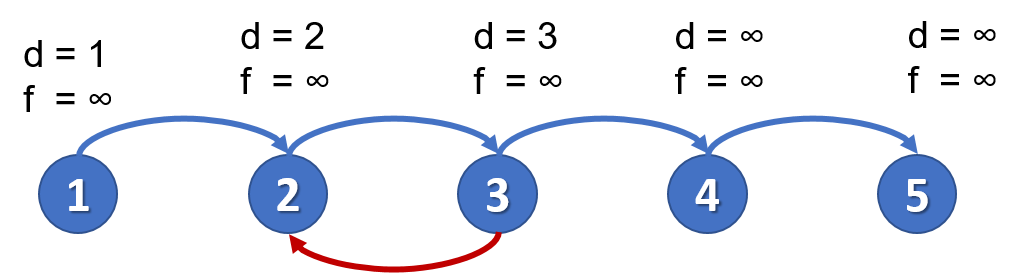
## **Descrição da solução**

A representação do problema foi feita através de um grafo dirigido, onde os vértices representam as fotografias e os arcos, a ordem pela qual estas foram fotografadas, da mais antiga para a mais recente.

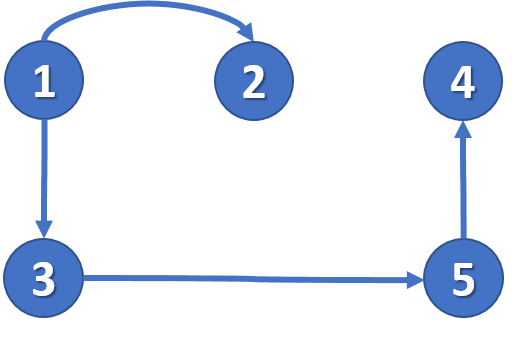
O grafo foi implementado através de uma lista de adjacências, onde cada novo arco é adicionado no início da lista, para evitar percorrê-la por inteiro.

Para a determinação da ordenação, foi realizada uma DFS, ligeiramente modificada, de forma a obter uma ordenação topológica dos vértices. Essa será única, se e só se existir um arco entre cada par de vértices consecutivos em ordem topológica. De forma a verificar essa condição, comparam-se todos os arcos de saída do -ésimo vértice da ordenação topológica com o vértice da ordenação - se o valor dos correspondentes vértices for igual, passa-se para o próximo par de vértices. Estas comparações são sucessivas e feitas para todos os vértices, parando no caso de não se verificar a igualdade. Se chegar ao último vértice, então verificou-se a condição e é possível apresentar a ordenação das fotografias.

No caso de não se verificar, significa que existem diversas organizações topológicas e, consequentemente, várias ordenações cronológicas das fotografias, apresentando-se então a mensagem “**Insuficiente**” (Grafo 2).

Quando na DFS é detetado um ciclo, o que ocorre quando um vértice visitado tem o seu tempo de descoberta diferente de infinito e o seu tempo de fecho igual a infinito, significa que existem relações inconsistentes e que é impossível arranjar uma ordenação coerente, logo é apresentada a mensagem “**Incoerente**” (Grafo 1).

**Grafo 1:** Caso incoerente na DFS.



**Grafo 2**: Caso insuficiente.

## **Análise Teórica**

O algoritmo começa inicializando todos os tempos de descoberta e de fecho a infinito. Logo, é necessário percorrer um vetor que contém tantas entradas quantas o número de vértices. Assim, a complexidade é .

Sempre que se insere um arco na lista ligada, e uma vez que a inserção é feita no início da lista, a sua complexidade é constante, .

De seguida, é realizada a DFS, cuja complexidade é . No caso de não ser incoerente, a complexidade é para verificar se existem arcos entre os vértices da ordenação topológica.

Finalmente, caso seja possível determinar uma ordenação válida, é escrito o resultado no *stdout*, cuja complexidade é .

Conclui-se que **a complexidade do algoritmo é**:

## **Avaliação Experimental dos Resultados**

Os resultados foram apurados utilizando uma máquina virtual com as seguintes características:

* Sistema Operativo: GNU/Linux - Xubuntu;
* RAM: 1 GB;

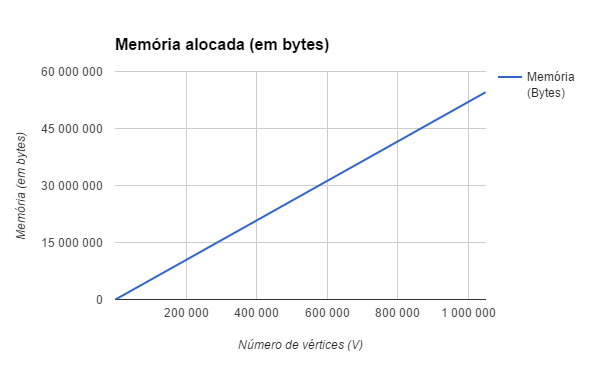
Os testes tiveram como *input* vários ficheiros gerados, utilizando o gerador de *inputs* fornecido pelo corpo docente.

O tempo de execução é linear, relativamente ao número de vértices e arestas do grafo.



A ferramenta utilizada foi o comando *time,* existente na *bash* do GNU/Linux. Uma vez que a precisão desta ferramenta vai apenas até ao milésimo do segundo, só é possível obter uma amostra com elevado rigor a partir de 8 000 vértices. De facto, para valores de inferiores a 4 000, o tempo de execução situa-se muito próximo dos 4 milissegundos.

A memória alocada é linear, relativamente ao número de vértices e arcos do *input*.



Foi utilizado o programa *valgrind* para efetuar as alocações de memória e corrigir eventuais fugas (*memory leaks*).

## **Referências**

**Graphs with topological sort:** <http://stackoverflow.com/questions/8099804/pre-requisite-for-graphs-with-unique-topological-sort>

**DFS**:

Slides disponibilizados na página *web* da cadeira **Análise e Síntese de Algoritmos**