ANÁLISE E SÍNTESE DE ALGORITMOS

1º PROJECTO 2016/2017
GRUPO 046
ANDREIA ROGÉRIO Nº78557

DESCRIÇÃO DO PROJETO

No problema a tratar pretende-se ordenar cronologicamente um grupo de fotografias, que aqui são descritas como nós de grafos. Entre vários pares de fotografias existe uma relação cronológica estabelecida tal que a direcção dos arcos será do antecessor em direcção ao predecessor.

Dependendo das relações fornecidas no input, a ordenação poderá ser possível, e o programa deverá fornecê-la nesse caso, ou não possível, devido a um número insuficiente de relações disponíveis, ou a relações que se contradizem cronologicamente.

IMPLEMENTAÇÃO DO PROGRAMA

A implementação do programa foi feita na linguagem C, com recurso às bibliotecas <stdio.h> e <stdlib.h>.

A resolução do problema passou pela utilização de grafos dirigidos, representados por uma lista de adjacência. A implementação desta lista foi feita com base num vector primário para representar cada um dos vértices, cada um com a sua respectiva lista simplesmente ligada para representar os arcos que se dirigem a outros vértices. Para inicializar o grafo foram incluídas funções de criação e inserção de arcos.

Para analisar o grafo procedeu-se a uma procura em profundidade (DFS) que chama recursivamente a função DFS_visit. Deste modo, cada nodo alterna entre o estado WHITE se nunca foi visitado (estado inicial de todos), GREY se foi visitado mas ainda não foram percorridos todos os arcos que tem origem no mesmo, e BLACK se estes já foram percorridos. Para guardar os estados foi utilizado um vector denominado mark[]. Quando são visitados denomina-se ao vértice predecessor o pai do vértice visitado e guarda-se essa informação no vector p[].

ESTRUTURA DO PROGRAMA – MODIFICAÇÕES À DFS

Durante a DFS são analisadas várias particularidades que permitem identificar possíveis ordenações, incoerências e insuficiências. Criou-se uma variável global, **output**, que terá o valor de 0 inicialmente, passando a 1 se forem detectadas características de Insuficiência, e a 2 se existir Incoerência, sendo que neste último caso o programa não permite que o output modifique o seu valor.

Para construir o programa estipularam-se os possíveis casos:

- Incoerências:
    - Existência de Ciclos – traduzem-se em encontrar durante a DFS um vértice já visitado mas não finalizado, ou seja, GREY.
- Insuficiências
    - Quando existem vários ‘’ramos’’ que partem de um mesmo vértice;
    - Quando existe mais do que uma ‘’árvore’’ no grafo;

Podem ser controladas através de:

- um contador que guarda o número de vértices sem ‘’filhos’’, sendo que se existir mais do que uma árvore haverá mais do que um vértice assim, tal como em ramos. Traduz-se num apontador na lista de adjacências NULL.
- um vector, p[],que guarda em cada índice i o pai de i, sendo que se existirem vários vértices com o mesmo pai então existiram ramos, e se vários não tiverem pai então existirão várias árvores.
- comparação entre o número de vértices e o número de arestas, que se for tal que V>E+1 então há Insuficiência;

Os vectores construídos (mark[]e p[]) fazem uso de memória dinâmica, uma vez que são criados enquanto variáveis globais, mas o seu tamanho depende do input.

FUNÇÃO MAIN

A função principal do programa, GRAPHorder, chama a função DFS, a qual modifica o valor da variável output. Posteriormente, verifica qual o estado da variável **output** e responde em conformidade.

A função depende dos valores de input fornecidos para criar o grafo. De notar que os valores enviados para a estrutura de dados para os valores das arestas são u-1 e v-1. Isto deve-se ao facto de o programa esperar como input N vértices de valores de 1 a N, e a estrutura de dados interna funcionar com N vértices de valores 0 a N-1.

COMPLEXIDADE DO PROGRAMA

- Inicialização do grafo: O(|V|)
- Inserção das arestas: O(|E|)
  Subtotal: O(|V|+|E|)

- Função DFS modificada: O(|V|+|E|)
  (Uma vez que é aplicada a uma lista de adjacências e garante-se que todos os vértices e arestas são apenas percorridos uma só vez)
- Função DFS aplicada ao vector de pais: O(|V|+|E|)
- Função Print: tempo constante;

Total: O(|V|+|E|) – complexidade linear

FALHAS DO PROGRAMA

Verificou-se que o programa apresenta falhas, especificamente em situações de arcos para a frente. Isto ocorre porque nessas situações os vértices têm mais do que um vértice adjacente, embora não se trate da situação de vários ''ramos'', anteriormente descrita.

Para corrigir esse problema removeu-se a condição de número mínimo de vértices adjacentes, substituindo-a pela reformulação do vértice pai. Deste modo, quando um vértice **i** encontra um vértice **j** que já foi anteriormente visitado e finalizado, **i** passará a ser o pai de **j** se o pai de **j** ainda não estiver finalizado (BLACK), uma vez que essa situação será indicadora que o vértice **i** procede o pai do vértice **j**.

Esta alteração não foi bem sucedida por motivos de compilação, pelo que não foi implementada na versão final submetida.

Igualmente por questões de compilação não foi possível analisar o vector de pais como pretendido, sendo que as situações em que vários vértices têm o mesmo pai não são detectadas. Deste modo, como resultado final de ordenação efectua-se uma DFS ao vector de pais, que tem bons resultados no caso de não existirem arcos para a frente.

AVALIAÇÃO EXPERIMENTAL DOS RESULTADOS

Para verificar a linha de tendência linear sugerida pela análise à complexidade, analisou-se a variação do tempo de execução do algoritmo criado de acordo com o tamanho do input. Para criar os inputs recorreu-se à ferramenta disponibilizada na página da cadeira.

No entanto, devido às falhas no programa criado não foi possível obter dados experimentais viáveis pois para certos inputs. Mais concretamente, muitos inputs desenhados para construir um grafo de ordenação possível são mal analisados e classificados como uma situação de Insuficiente, terminando com um menor tempo de execução do que inputs de menores dimensões que não são mal classificados.