Análise e Síntese de Algoritmos

2015/2016

Relatório do Primeiro Projeto

Grupo 40

81186 - Stéphane Duarte | 81858 - João Oliveira

INTRODUÇÃO

No âmbito da cadeira de Análise e Síntese de Algoritmos, foi-nos proposto um projeto cujo objetivo era identificar quais as pessoas fundamentais numa rede para a transmissão de informação, isto é: uma pessoa *p* é considerada fundamental se o único caminho para a partilha de informação entre outras duas pessoas r e s passa necessariamente por p (onde $p \neq r$ e $p \neq s$).

Deste modo, o problema vai ser encarado como um grafo não dirigido no qual terá de ser aplicado um algoritmo de procura pelos vértices fundamentais do grafo, recorrendo ao auxílio do algoritmo DFS (procura em profundidade).

DESCRIÇÃO DA SOLUÇÃO

Este problema é encarado como um grafo não dirigido, em que cada vértice representa uma pessoa e cada aresta representa uma ligação entre pessoas (onde existe transmissão de informação).

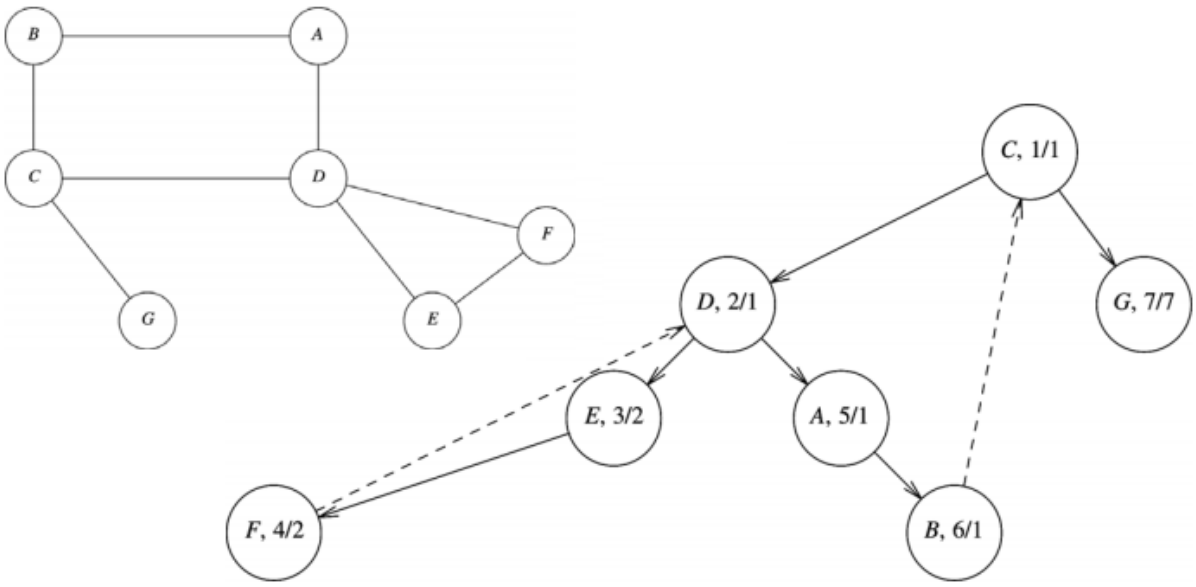
A solução foi implementada em C++ para facilitar a implementação de ADTs, usando as bibliotecas list e vector.

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

Utilizando o algoritmo DFS, cada vértice V é numerado à medida que é visitado (visittime(V)). É também numerado com um lowtime(V) que representa a distância do vértice com menor distância que o sub-grafo com raiz em V pode alcançar.

A partir daqui há duas hipóteses que tornam um vértice num ponto fundamental:

1. Se for a raiz do grafo e tiver mais que um filho;
2. Se não for a raiz do grafo mas tiver um filho W tal que lowtime(W) ≥ visittime(V). Para entender este ponto, podemos questionar-nos: há algum vértice W, filho de V, que não consegue atingir um vértice visitado antes de V? Se sim, significa que V é a raiz de um sub-grafo que não tem nenhuma ligação ao grafo anterior, então retirar o V ao grafo iria separá-lo em dois, ou seja, V é um vértice fundamental.



Grafo original e representação do grafo com os valores visittime e lowtime

Como input, é recebido:

1. Uma linha com dois valores:
    a. O primeiro valor corresponde ao número de pessoas (N), ou seja, o número de vértices do grafo. Este valor é utilizado para a criação do grafo e para a criação dos vetores dinâmicos necessários para a execução do algoritmo.

   b. O segundo valor corresponde ao número de ligações (L) existentes entre as pessoas. Este valor vai ser utilizado no ponto 2.
2. L linhas com dois elementos numéricos A e B entre 1 e N, representando assim a ligação de A a B e, consequentemente, de B a A (grafo não dirigido).

Após receber estes dados, é aplicado o algoritmo ao grafo, que atualiza o vetor de booleanos que reconhece se um vértice é ou não fundamental. Depois são calculados o menor e maior valor dos identificadores dos vértices fundamentais, recorrendo ao uso de um contador e de duas variáveis.

Como output, obtém-se:

1. Uma linha com o número de pessoas fundamentais na rede.
2. Uma linha com o menor, seguido do maior, valor do identificador das pessoas.

   NOTA: No caso do número de pessoas fundamentais na rede ser 0, então os identificadores apresentados pelo ponto 2 serão -1 e -1.

ANÁLISE TEÓRICA

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

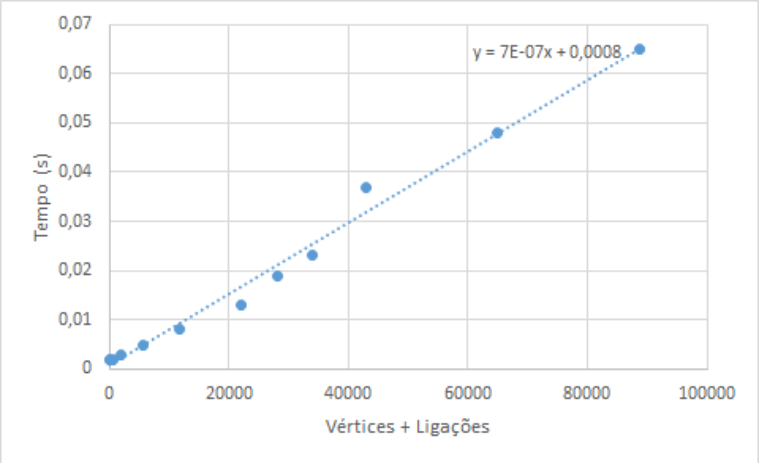
1. Inicialização do grafo: O(V);
2. Ciclo para gerar as ligações: O(L);
3. Ciclo para inicializar os vetores: O(V);
4. Algoritmo DFS: O(V+L);
5. Ciclo para atualizar os valores mínimo e máximo e o contador: O(V).

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

AVALIAÇÃO EXPERIMENTAL DOS RESULTADOS

Casos de teste:

V	L	Tempo (s)
8	7	0.002
10	9	0.002
200	500	0.002
400	1500	0.003
500	5000	0.005
5000	6800	0.008
10000	12000	0.013
13000	15000	0.019
16000	18000	0.023
20000	22803	0.037
25000	40000	0.048
30000	58783	0.065



Como anteriormente referido, a complexidade do programa é, teoricamente, O(V+L). Foram então feitos testes experimentais para verificar o valor da complexidade.

Após 12 testes, foi verificado que à medida que a soma vértices+ligações aumentava, o tempo de execução aumentava linearmente, como se pode ver no gráfico acima.

Deste modo, fica comprovado que a complexidade do programa é, de facto, O(V+L).

REFERÊNCIAS

http://www.eecs.wsu.edu/~holder/courses/CptS223/spr08/slides/graphapps.pdf

http://www.geeksforgeeks.org/biconnected-components/

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