Análise e Síntese de Algoritmos

Introdução:

O Sr. João Caracol pediu para fazer uma auditoria à sua rede, esta consiste em, encontrar o número de sub-redes existentes na rede, o identificador de cada sub-rede, o número de routers da rede que se forem removidos resultariam no aumento do número de sub-redes, e o número de routers da maior sub-rede resultante da remoção destes.

No nosso programa, considerarmos a rede como uma grafo. Cada vértice do grafo representa uma ligação da rede, ou seja, um router. Este pode estar ligado a todos os routers da rede, criando uma rede robusta de ligações, ou então estar ligado apenas a um certo número de routers, fazendo assim parte de uma sub-rede. Os routers que, se removidos, resultariam no aumento do número de sub-redes são denominados pontos de articulação do grafo.

Descrição da solução:

Para a representação da rede, utilizamos um grafo, definido na nossa estrutura de dados, como uma lista de adjacências por cada vértice. Um vértice (router), possui atributos, os quais permitem diferenciar um identificador de uma sub-rede e um ponto de articulação do grafo.

Para contabilizar o número de sub-redes, de uma rede, o respetivo identificador e pontos de articulação associados, recorremos ao algoritmo de Tarjan.

De seguida, para determinar o número de routers da maior sub-rede, gerada pela remoção de todos os pontos de articulação da rede, recorremos ao algoritmo Depth-first search (DFS).

Sara Machado | Nº 86923
Ricardo Grade | Nº 90774

Análise Teórica:

No nosso programa, alocamos um vetor com memória suficiente para armazenar V vértices, os quais são representados pela estrutura router. Sendo que em cada estrutura, se aloca a quantidade de nós igual ao número de ligações (bidirecionais) de cada router, pelo que, por cada ligação serão alocados 2 nós, dando assim um total de 2E nós.

Assim sendo, no pior caso, a complexidade em termos de memória é:

$$O(V) + O(2E) = O(V + 2E) = O(V + E).$$

Ao longo do nosso programa, existem algumas tarefas que consomem algum tempo de processador. Tais como:

- Inicializar as listas de adjacências dos routers (O(V)).
- Preencher as listas de adjacências dos routers, com as respetivas ligações (O(E)).
- Executar o algoritmo de Tarjan (O(V + E)).
- Executar o algoritmo DFS (O(V + E)).
- Imprimir os identificadores das sub-redes e contabilizar o número de pontos de articulação do grafo (O(V)).
- Libertar a memória utilizada pelas listas de adjacências dos routers (O(V)).

Assim sendo, no pior caso, a complexidade em termos de tempo é:

$$O(V) + O(E) + O(V + E) + O(V+ E) + O(V) + O(V) = O(5V + 3E) = O(V + E).$$

Análise Experimental:

Dadas as complexidades do nosso programa, em termos de memória e de tempo, serem ambas O(V + E), é de esperar que variando o número de vértices e arestas, esta variação se traduza de forma linear, relativamente à quantidade de memória utilizada e ao tempo de processador utilizado.

Ao analisarmos os gráficos, que se seguem, não nos deparamos com apenas uma variação linear. Dado que o nosso script gerador de testes varia para um determinado número de vértices, o número de sub-redes, que por sua vez influência o número de arestas, o número de vértices e arestas não será, sempre, proporcional. Como o número de sub-redes (R) é gerado a partir de uma determinada proporção de vértices, isto irá originar várias variações lineares distintas, uma para cada número de sub-redes.

Apesar de, não termos apenas uma variação linear com os diversos testes que fizemos, gerámos uma regressão linear, uma para cada contexto, memória e tempo, que representa a sua evolução tendencial no nosso programa.

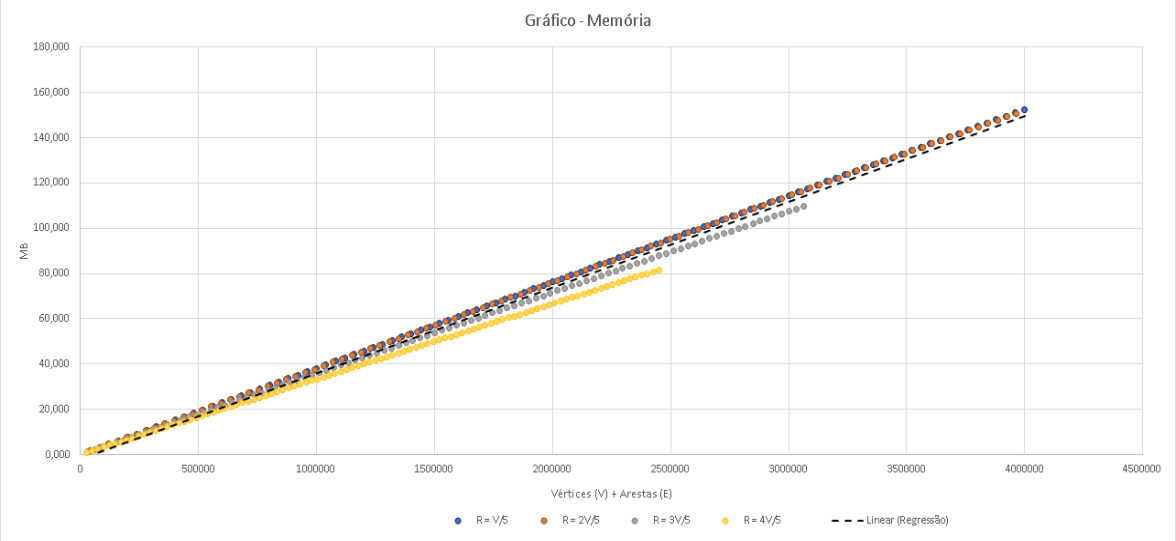

Gráfico 1: Representa a evolução da memória utilizada, pelo nosso programa, em 400 testes.

As razões pelas quais os gráficos associados a cada R, não serem coincidentes, deve-se ao facto de, para determinados valores de X, existirem várias combinações de Vi e Ei, tais que Vi + Ei = X, e ao facto das dimensões das estruturas de dados, associadas ao armazenamento de vértices e arestas, terem tamanhos diferentes. Pelo que O(X) produzirá valores diferentes, para cada combinação de Vi e Ei.

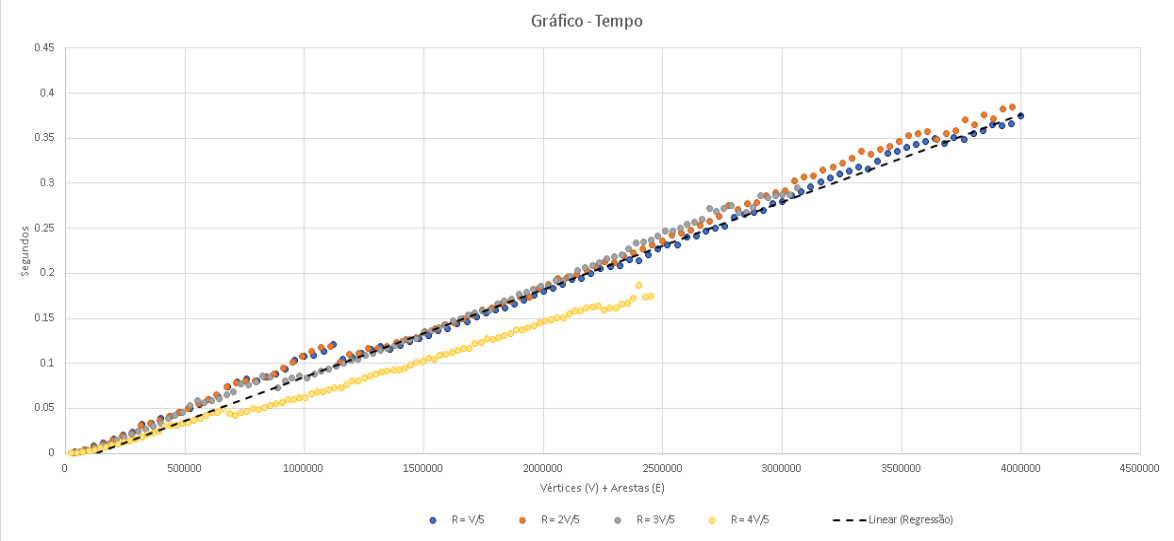

Gráfico 2: Representa a evolução do tempo de execução, do nosso programa, em 400 testes.

A evolução dos testes associados a cada R, não é completamente linear. O que se deve ao facto de os testes terem sido realizados numa máquina virtual, do estado de atividade no computador não ter sido regular, da duração prolongada dos múltiplos testes, entre outras razões que escapam ao nosso controlo.