ANÁLISE E SÍNTESE DE ALGORITMOS

RELATÓRIO DO 1º PROJETO

O objetivo deste projeto é fazer a implementação de um programa que facilite a realização de uma auditoria a uma rede.
A solução do problema mencionado faz uso de um grafo não dirigido onde os vértices são os routers e as arestas as respetivas ligações entre si. Para identificar as sub-redes e localizar os routers que, se forem atacados ou desligados o número de sub-redes aumentaria, é utilizado um algoritmo de DFS.

Na solução do problema **executam-se os seguintes passos:**

- Leitura do número de routers, número de conexões entre routers e das conexões existentes na rede, presentes num ficheiro de input. Este ficheiro é lido através do standard input. Todos os valores são alocados em memória, criando-se uma lista de adjacências, através do uso das estruturas *router_t* e *node_t*;

- De seguida, para cada um dos routers, verifica-se se o router já foi visitado. Caso contrário, o número de sub-redes é incrementado, executa-se o algoritmo de DFS implementado e, no fim, o ID da sub-rede é guardado na lista de IDs das sub redes;

- Após o passo descrito acima, reinicializa-se os vértices, ou seja, todos os vértices passam a não estar visitados (para se poder executar o algoritmo uma segunda vez), para além disso, o valor da variável *secondDFS* passa a true (explicado mais à frente), o algoritmo de DFS é então executado mais uma vez para determinar o número de routers da maior sub-rede resultante da "remoção" de todos os routers que quebram uma sub-rede (o programa não os remove, apenas os ignora), estes pontos denominam-se por pontos de articulação..

- Por fim, imprime-se a informação pedida pelo enunciado: o número de sub redes, o ID de cada sub rede, o número de pontos de articulação e o número de conexões da maior sub rede.

Como mencionado, foi implementado um algoritmo de DFS na solução, tendo em conta os **vários objetivos** a serem alcançados:

- Em cada chamada da função DFS, verifica-se se o número do router é maior que o ID da sub-rede do qual faz parte, e em caso afirmativo o ID da sub-rede passa a ser o número desse router;

- Atualizam-se as variáveis *low* e *discovery* com o tempo de descoberta do vértice e, para cada um dos vértices ainda não visitados que fazem parte da lista de adjacências do vértice atual, faz-se uma chamada recursiva da função DFS e incrementa-se o número de filhos do vértice atual. Caso o vértice já tenha sido visitado verifica-se se o low do vértice atual (vértice i) é maior que o tempo de descoberta do vértice já visitado (vértice x->num-1) e verifica-se também se o vértice já visitado não é o pai do vértice atual, caso estas duas condições sejam verdadeiras, atualiza-se o low do vértice atual;

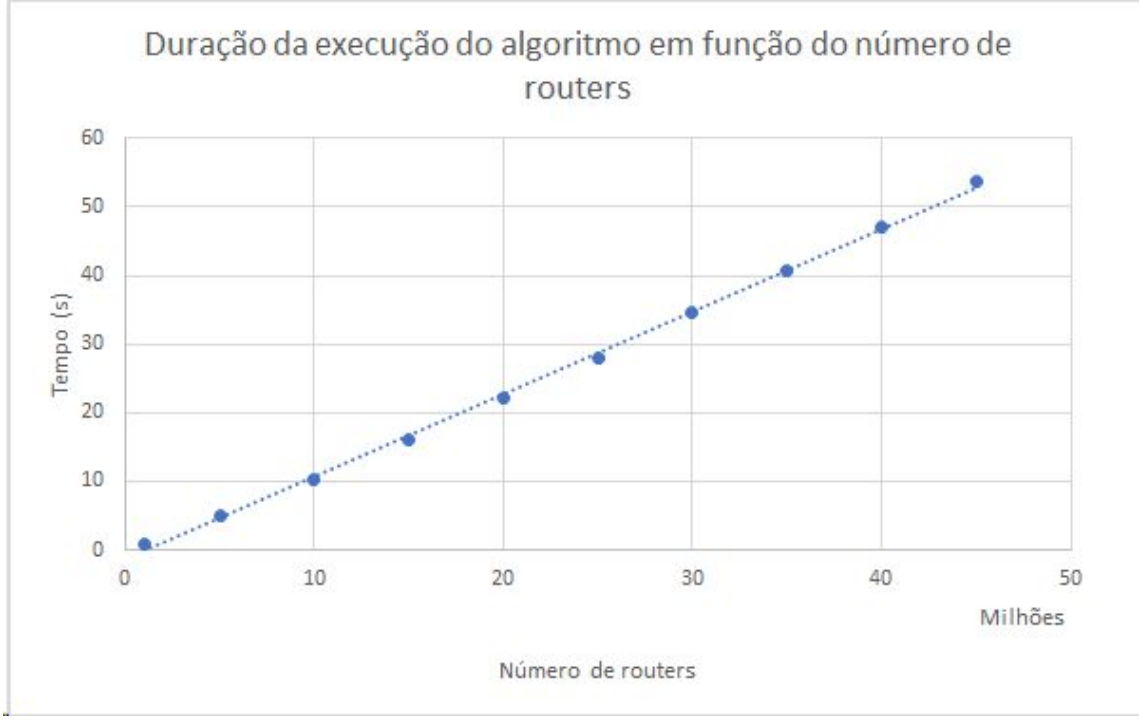
- Após as chamadas recursivas, se o low do vértice atual for maior que o low do seu adjacente atualiza-se o low do vértice atual e posteriormente verifica-se se cada vértice é ponto de articulação (Pontos de articulação: Raíz do grafo com 2 ou mais filhos; Vértice que não seja raíz do grafo e tenha um filho com um low maior que o seu tempo de descoberta).

- Na segunda execução do algoritmo DFS, o valor da variável *secondDFS* é atualizado para true, a fim de executar o algoritmo com um único objetivo: obter o número de routers da maior sub-rede resultante da remoção de todos os pontos de articulação.

O grafo é constituído por uma ou mais componentes. Os vértices que, quando removidos, dão origem ao aumento do número de componentes do grafo são denominados pontos de articulação. Estes podem ser identificados, através de um algoritmo baseado em DFS, sendo que o seu tempo de descoberta mínimo é igual ao menor valor do tempo de descoberta atingível por um arco para trás. Neste caso, o grafo representa a rede inteira de routers, sendo que cada uma das suas componentes representa uma sub rede. Os routers que, quando atacados ou desligados resultam no aumento de sub redes, são representados através de pontos de articulação e descobertos da mesma forma (mencionado acima).

O algoritmo usado começa num dos vértices, a raiz, explorando tanto quanto possível os ramos de um grafo, até retroceder quando chegar a um vértice que não tem filhos. Este algoritmo tem complexidade O(V + E), ou seja, é linear, onde V e E correspondem, respetivamente, ao número de vértices (routers) e arestas (conexões) do grafo (rede).

De forma a comprovar a complexidade teórica executou-se o programa com inputs de diferentes dimensões. O número de routers utilizado inicialmente foi 1 milhão, tendo-se aumentado de 5 em 5 milhões até 45 milhões (do 1º para o 2º input aumentou apenas 4 milhões). O número de conexões aumenta numa razão de 1.5 em relação ao número de routers. Os resultados encontram-se no gráfico abaixo.



Ao observar o gráfico, é possível concluir que o tempo de execução do algoritmo aumenta linearmente em relação ao número de routers e de conexões. Logo, a complexidade teórica O(V + E) é comprovada.