**Análise e Síntese de Algoritmos**

**2018/2019**

**1º Projeto**

**Grupo AL008**

**Daniel Gonçalves – 91004**

**Gabriel Almeida – 89446**

## Introdução:

Este relatório estende-se sobre a solução encontrada para o primeiro projeto de ASA no ano 2018/2019.

Este projeto consiste numa rede de routers que estão ligados entre si, ou seja, consegue-se definir caminhos que liguem qualquer par de routers da rede. É possível isso não acontecer, logo a rede está dividida em subconjuntos de routers ligados entre si (sub-redes).

O programa consegue obter o número de sub-redes, os seus maiores identificadores (por ordem crescente), o número de routers que quebram uma sub-rede e qual o número de routers da maior sub-rede resultante da remoção de todos esses routers que quebram uma sub-rede, através de um *input* do número dos routers da rede, da quantidade de ligações entre esses routers e de cada uma dessas ligações, indicando quais routers liga.

## Descrição da solução:

O programa foi implementado em linguagem C.

Desenvolvemos duas estruturas para a utilização destes dados, um vetor em que cada índice contém as informações de cada router e um vetor em que cada índice (que correspondia a um router) continha as ligações desse mesmo para os outros routers, criando assim uma lista de adjacências, de modo a consideramos este problema como um grafo não dirigido.

Para calcular o número de sub-redes, utilizamos um algoritmo que, através duma DFS, acedia a cada router, atribuía um identificador para cada subconjunto e aplicava o mesmo identificador a todos os routers nessa árvore, recursivamente.

Para detetarmos quais routers quebram uma sub-rede, utilizamos o algoritmo de Tarjan para deteção de pontos de articulação (pontos que aumentam o número de componentes fortemente ligadas). Este algoritmo utiliza uma DFS, guardando informações como a profundidade, o menor valor de profundidade dos seus descendentes (conhecido por *lowtime*), o seu predecessor e o número de filhos

de cada vértice. De seguida, verifica se cada vértice é um ponto de articulação se a sua profundidade for menor que o *lowtime* dos seus descendentes ou o se o vértice pelo qual começámos a pesquisa, *start*, não tiver predecessor e tiver 2 ou mais filhos.

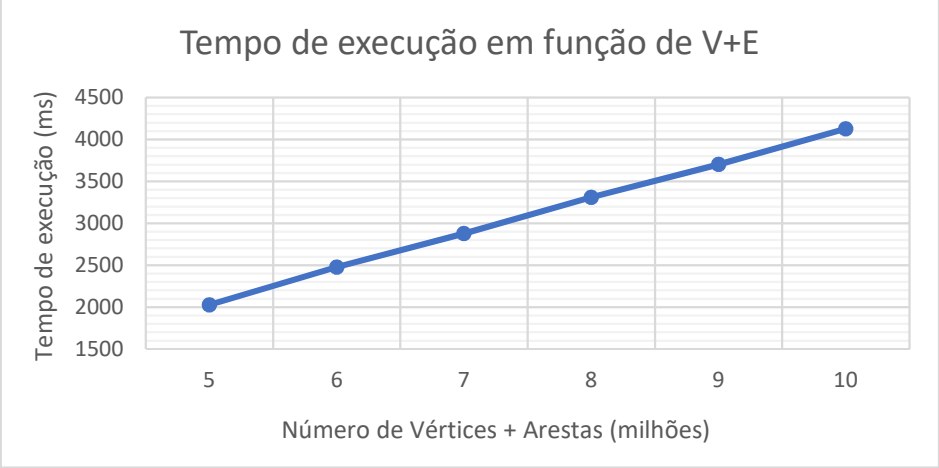
**Análise Teórica:**

Sendo que V é o número de vértices e E o número de arestas, calculamos que a complexidade temporal do primeiro algoritmo (DFS) seja O(V+E) e a do segundo (Tarjan) seja O(V+E). Tendo o programa uma complexidade de O(V+E).

**Avaliação Experimental dos Resultados:**

Para estas experiências utilizamos um computador com processador Intel Core i7-7700HQ 2.8GHz, com 16GB de memória, com o sistema operativo OpenSUSE Leap.

Corremos o nosso programa com os variados ficheiros de input dados para a resolução do problema com o comando *time* do Linux, de modo a calcularmos o tempo utilizado em cada teste, em função de V+E, com valores entre 5 milhões e 10 milhões.




Como se pode verificar no gráfico, a complexidade do programa é mesmo linear, ou seja, O(V+E).

**Referências:**

Utilizamos a seguinte referência para a realização do projeto:


- https://en.wikipedia.org/wiki/Biconnected_component (algoritmo de Tarjan)