ANÁLISE E SINTESE DE ALGORTIMOS

1 º Projeto 2018/2019 | Grupo 014 | 90704 | LEIC-T

Introdução

O problema abordado neste projeto corresponde a ajudar na auditoria a uma certa rede de routers, de modo a conseguir extrair quatro propriedades:

1. O número de sub-redes. Caso não se consiga estabelecer um caminho entre todos os pontos da rede então esta está dividida sub-redes, ou seja, não for possível estabelecer um caminho entre quaisquer dois pontos.
2. O identificador de cada sub-rede. Encontrar para cada sub-rede existente o seu identificador, em que uma sub-rede é identificada pelo router com maior identificador que lhe pertence.
3. O número de routers que ao serem desligados resultariam no aumento de sub-redes.
4. O tamanho da maior sub-rede resultante da remoção de todos os routers que ao serem desligados quebram uma sub-rede.

A situação acima descrita foi resolvida tendo em conta uma representação em grafo em que os vértices são routers e as arestas as ligações entre estes, o problema resume-se maioritariamente a identificar componentes fortemente ligados e os pontos de articulação deste grafo.

Descrição da solução

Este projeto foi desenvolvido usando a linguagem C.

- Estruturas de dados:

  O grafo que representa a situação descrita foi implementado tendo por base uma lista de adjacências. O tamanho desta lista é igual ao número de routers existentes mais uma unidade, pois foi ignorada a primeira posição, index igual a 0, de modo a que cada router fosse guardado no index igual ao seu identificador. Esta lista contém dados do tipo Node, que são caraterizados por terem toda a informação de cada vértice ao longo da execução do algoritmo de exploração do grafo, incluindo o identificador, o tempo de descoberta, se é ou não um ponto de articulação, o seu low (o valor mínimos que pode ser atingido a partir de cada vértice) e um ponteiro para um dos seus vértices adjacentes.

  A cada posição desta lista foram sendo adicionados os routers adjacentes a cada um, tendo por base listas simplesmentes ligadas. De modo a ter menos dispêndio de memória foi criada uma nova estrutura de dados, Neighbour, que apenas contém o identificador de cada router e um ponteiro para o seguinte.

- Algoritmo de exploração do grafo e identificação de pontos de articulação:

  A ideia base do algoritmo é fazer uma DFS(depth-first search).
  Procura-se na lista de adjacências o primeiro vértice que ainda não tenha sido visitado e aplica-se a função searchGraph() a esse vértice, marcando-se como visitado.
  Para cada vértice adjacente, se este ainda não tiver sido visitado então chama-se novamente searchGraph com este novo vértice e assim sucessivamente (recursividade) comparando e atualizando se necessário, no retorno, os valores de low de cada vértice. Se o um dos vértices adjacentes já tiver sido visitado o valor de low do é atualizado caso necessário.

  Um vértice é considerado um ponto de articulação em duas situações:

  1. É a raíz da árvore e é o pai de dois vértices, ou seja, foi a partir da raíz que dois vértices foram descobertos e encontram-se ligados a esta por arcos de árvore.
  2. Não é a raíz da árvore e o seu tempo de descoberta é menor ou igual ao tempo low de pelo menos um dos seus filhos.

  À medida que o algoritmo avança e novos vértices são descobertos testa-se cada vértice para a possibilidade de ser um ponto de articulação. Caso se encontre um ponto de articulação a variável correspondente é incrementada e o vértice em questão fica marcado com a variável articulationPoint a 1.

- Passagem final para identificar a maior sub-rede caso os pontos de articulação sejam removidos:

  É feita, no fim, uma exploração final, também com base em recursão, de modo a encontrar a maior sub-rede resultante de retirar todos os pontos de articulação do grafo. A variável correspondente é incrementada à medida que se vão descobrindo novos vértices, até chegar a um beco ou a um novo ponto de articulação.
  No final da exploração de cada sub-rede o valor da variável com o tamanho máximo é substituido com o novo valor, caso este seja maior.

Análise Teórica

O programa começa com a alocação de memória para a lista de adjacências e a inicialização de cada uma das suas posições, o que tem uma complexidade de O(V) em que V é o numero de vértices do grafo. A seguir, é feita a leitura do input e a inserção dos arcos que fazem a ligação entre os vértices. Assumindo a leitura do input em tempo constante este processo tem um custo de O(E), em que E é o número de arestas no

grafo. Isto acontece porque o custo de inserção de uma aresta é O(1) pois a ordem dos vértices adjacentes é irrelevante. Concluindo o processo de leitura de input e construção da representação do grafo tem uma complexidade de O(V+E).

O próprio algoritmo de exploração do grafo tem por base uma DFS que garante que cada vértice e cada aresta são visitados uma única vez, garantindo uma complexidade O(V+E). Apesar de serem executadas algumas operações de comparação ou de verificação da existência de pontos de articulação considera-se que não influenciam de modo significativo a complexidade do algoritmo.

Mais tarde é feita uma última exploração do grafo para encontrar o tamanho da maior sub-rede resultante da remoção de todos os pontos de articulação, novamente de complexidade O(V+E), pois está na base uma DFS que garante mais uma vez as passagens únicas por todos os vértices e arestas.

No fim é libertada toda a memória alocada nestes processos, tendo novamente um custo de O(V+E), pois é preciso libertar todos os vértices e todas as suas ligações.

Concluindo, a complexidade total deste algoritmo é teóricamente O(V+E).

Análise Experimental

De acordo com os dados recolhidos em cerca de 200 testes com diferentes valores de input, obteram-se os Gráficos 1 e 2 apresentado abaixo. Foram gerados diferentes testes em que os inputs eram diferentes combinações de números de vértices e arestas, sendo que V+E variou entre cerca de 0.5 milhões e 3.775 milhões de unidades no caso do gráfico 1 e entre cerca de 0.5 milhões e 3.15 milhões no caso do gráfico 2.

- Gráfico 1: Cada teste foi realizado 70 vezes e no fim foi calculada a média dos tempos registados. O gráfico contém a relação entre o total de vértices e arestas (V+E) e o tempo que o algoritmo demorou a executar, em segundos.
- Gráfico 2: Cada teste foi realizado com recurso à ferramenta massif do valgrind de modo a conseguir obter o pico máximo de memória durante a execução do algoritmo, em MegaBytes.

De acordo com os dados recolhidos e com os gráficos gerados podemos concluir que a análise teórica é comprovada. Ambos os gráficos podem ser reduzidos a uma reta ou seja, este algoritmo tem uma complexidade O(V+E).

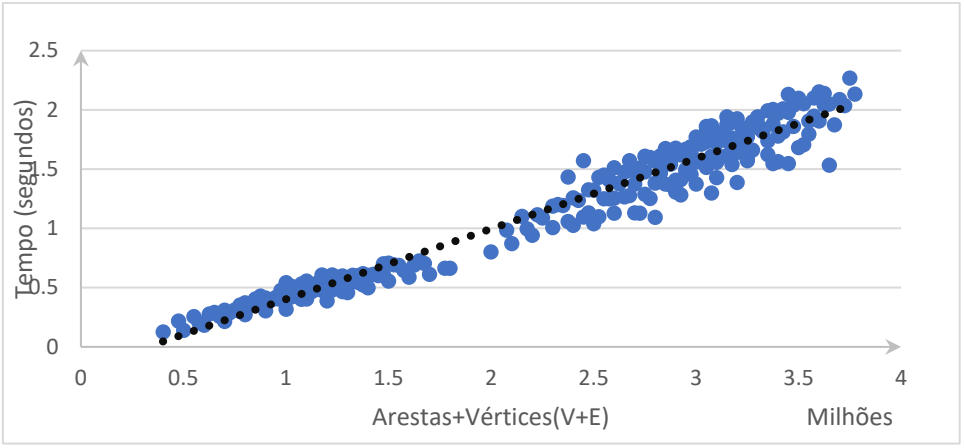

Gráfico 1:Relação entre o numero total de vértices e arestas da rede com o tempo de execução

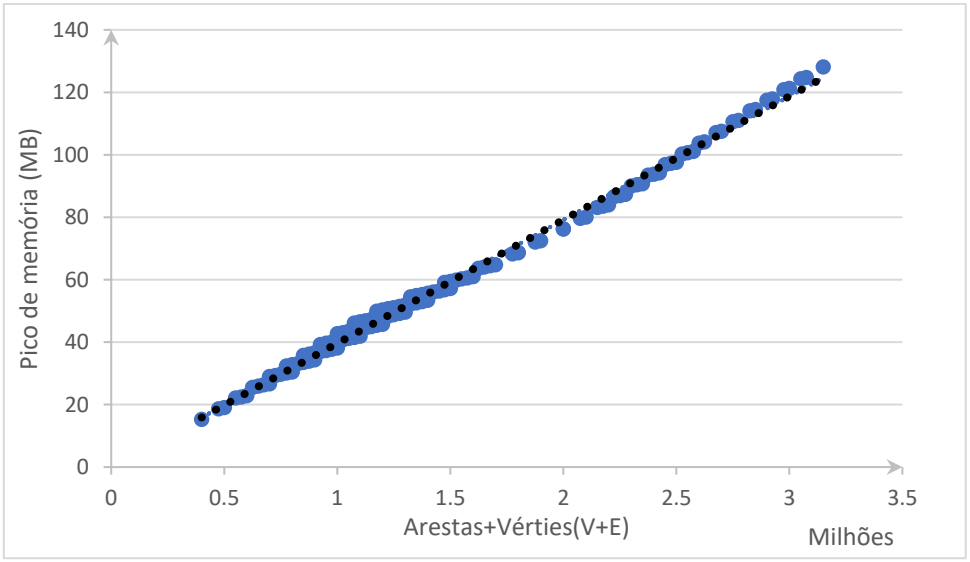

Gráfico 2:Relação entre o total de vértices e de arestas com o pico de mémória utilizada