Relatório do 1º Projeto de ASA

2º Semestre – 2017/18

Introdução

Este relatório é no âmbito do 1º projeto de ASA que no nosso caso, foi implementado na linguagem C.

O problema consiste na identificação de sub-redes regionais de uma rede de distribuição de supermercado, de forma a que seja possível enviar produtos de uma região para qualquer outro ponto da rede regional.
O objetivo deste projeto é ajudar o Sr. João a identificar as sub-redes regionais, i.e., encontrar o número de SCC's de um grafo onde cada aresta representa um supermercado, o número de ligações entre elas, e por fim o conjunto dessas ligações.

Descrição da solução

Na nossa solução o grafo é representado através de uma lista de adjacências. Para obter o número de SCC's aplicamos o algoritmo Tarjan.

O algoritmo de Tarjan é uma versão modificada do algoritmo DFS e é usado para encontrar SCCs. Para o aplicarmos, adicionamos à estrutura que caracteriza o vértice do grafo dois inteiros, d e low. d é um inteiro que representa a profundidade do grafo em relação à raiz (o d da raiz é 1). O low representa o menor valor de d atingível a partir do vértice respetivo.

Para este algoritmo precisamos ainda de um stack.
O algoritmo visita um vértice (v) e coloca-o no stack. Depois, percorre os vértices adjacentes deste e, quando encontra um que ainda não foi visitado chama-se recursivamente para esse adjacente.
Caso este já tenha sido visitado e caso o low deste seja menor que o do vértice atual, atualiza-o para ser igual.
Se os valores d e low do vértice forem iguais então este é a raiz de uma SCC e, portanto, retiramos os elementos do stack até encontrar a raiz (o próprio vértice) sendo que os elementos retirados fazem parte da mesma componente fortemente ligada. Caso um vértice não seja atingível pelo vértice de origem selecionado, o seu valor de d não vai estar inicializado e, portanto, vamos recomeçar o algoritmo nesse vértice.

No fim do algoritmo Tarjan, criamos um vetor de arestas cujo tamanho máximo é o nº de arcos do nosso grafo (número máximo de ligações entre SCC's) percorremos todas as arestas do grafo para identificar quais as que ligam SCC's diferentes e adicionamo-las ao vetor, incrementando também uma variável que conta o nº total de ligações encontradas.

Após termos um vetor cujo conteúdo é as ligações ordenamo-lo usando a função qsort da biblioteca stdlib.

Análise Teórica

Ao inicializar o grafo (alocar memória para os seus vértices) percorremos a lista de adjacências ($\Theta(V)$) quando adicionamos os respetivos arcos a essa lista temos uma complexidade de O(E) pois voltamos a percorrer os vértices, mas agora para adicionar as arestas.

- No total, ao inicializar a complexidade é de O(V + E).

Como na nossa solução aplicamos o algoritmo Tarjan simples temos complexidade de O(V + E).

Ao analisar o grafo, após o algoritmo Tarjan, para detetar as ligações entre redes, no pior dos casos temos uma complexidade de O(E) pois percorremos todos os arcos.

Após a obtenção das ligações entre redes ordenamos esse resultado usando o Quicksort, que no pior dos casos é $O(E^2)$ no entanto escolhemos este algoritmo pois em média tem uma complexidade de O(E logE).

Por fim ao fazer print das ligações obtidas a complexidade no pior caso é O(E) e representa o caso em que todas as arestas entre vértices são entre vértices pertencentes a SCC's diferentes.

A complexidade total do nosso programa é O(ElogE +V)

Avaliação Experimental:

Usámos o software gnuplot para gerar os gráficos da nossa análise experimental, e um script que gera grafos com SCC's aleatórias.

No gráfico 1, ao lado está explicita a relação entre o tempo e o número de vértices crescente.

Usámos o script para gerar grafos com um número de vértices crescente entre os 1000 e os 100000. Fixámos o número de arcos em 30000.

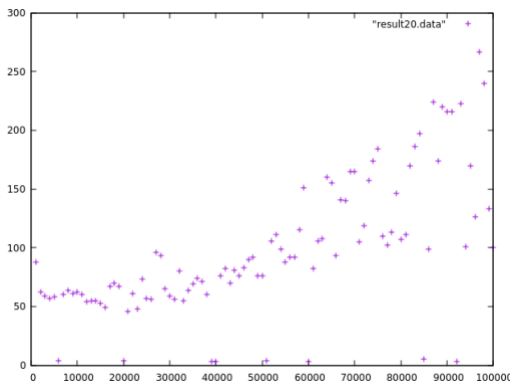

Gráfico 1: Relação entre tempo (eixo y) e número de vértices (eixo x)

Os resultados apresentados neste gráfico (1) estão de acordo com os resultados teóricos sendo que a relação é linear.

Observámos também a relação entre o tempo e o número de vértice + número de arestas (gráfico 2). Este também se encontra de acordo com os resultados teóricos, O(V + Elog(E)).

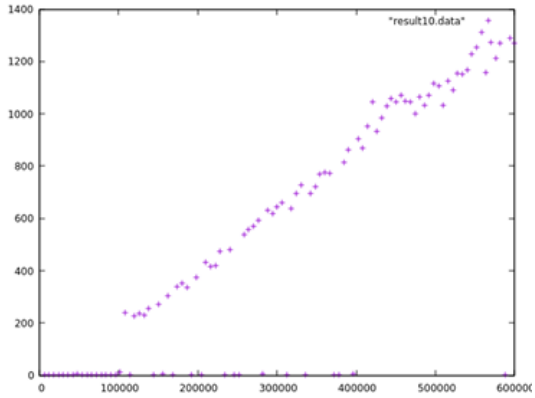

Gráfico 2: Relação entre tempo(eixo y) e número de vértice (constante 100000) + número de arestas (crescente) (eixo x)

A análise experimental mostrou que a complexidade do nosso algoritmo não está relacionada com o número de SCC's (gráfico 3). Nesse gráfico mantemos o número de vértices e arestas e vamos aumentando o número de SCC's.

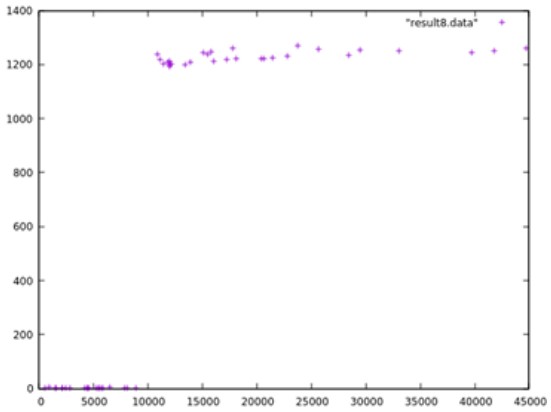

Gráfico 3: Relação entre tempo (eixo y) e número de SCC's (eixo x)

Referências:

https://en.wikipedia.org/wiki/Tarjan%27s_strongly_connected_components_algorithm (visitado a 17-03-2018)
https://fenix.tecnico.ulisboa.pt/downloadFile/1407993358870185/ch22_2.pdf (visitado a 17-03-2018)
https://stackoverflow.com/questions/6105513/need-help-using-qsort-with-an-array-of-structs (visitado a 17-03-2018)
Cormen, Thomas H., Charles Eric. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. Cambridge, MA: MIT, 2009. Print.

Realizado por:
Carolina Carreira, 87641
Miguel Barros, 87691
Grupo 36