Análise e Síntese de Algoritmos

1º Projeto

Grupo 91

João Bernardo – 86443

Pedro Antunes – 86493

Introdução

O presente relatório aborda a solução encontrada ao problema proposto pelo corpo docente no primeiro projeto do 2º Semestre do ano curricular 2017/2018.

O problema proposto aborda a rede de distribuição de uma cadeia de supermercados. É-nos pedido que identifiquemos sub-redes regionais de modo a que cada um desses supermercados tenha sempre todos os produtos num curto espaço de tempo.

Recebendo um input com o número de vértices (pontos de distribuição da rede), o número de ligações entre eles (ligações na rede de distribuição) e uma lista dessas mesmas ligações, o programa deve dividir a rede de distribuição em sub-redes regionais de forma a que numa região seja possível qualquer ponto de distribuição enviar produtos para qualquer outro ponto da rede regional.

Descrição da Solução

A implementação do programa foi elaborada em linguagem C.

Depois de o programa receber o input descrito anteriormente, é alocado um array, *pairList,* de estruturas *pair*. Esta última estrutura simboliza um par de pontos na rede distribuição, representando a variável x o ponto de origem e a variável y o ponto de destino, sendo cada *pair* um arco na rede de distribuição. Posteriormente, este array é ordenado pelos pontos de origem e de seguida pelos pontos de destino utilizando o quicksort.

Após esta ordenação é alocado um array de inteiros, *locator,* em que cada entrada indica o índice na *pairList* na qual se iniciam os vizinhos de cada vértice. Por exemplo, se na terceira entrada do *locator* estiver o inteiro 6, significa que os vizinhos do vértice 3 se iniciam no índice 6 da *pairList*.

No seguimento do programa, é alocado outro array de vértices (estruturas *vertex*), chamado *vertexes.* Cada *vertex* possui um valor do tipo inteiro, que identifica esse mesmo vértice, um booleano que indica se está na pilha, utilizada pelo Tarjan mais à frente, e dois inteiros, um que representa o tempo de descoberta e outro que representa o low, essencial no algoritmo de Tarjan.

Posteriormente, é alocado também um array de inteiros, *id,* que servirá para associar a cada vértice o valor mínimo da SCC correspondente (a sub-rede regional é identificada pelo ponto de distribuição com menor identificador que pertence à sub-rede). De seguida, é executado o algoritmo de Tarjan para identificar as SSCs. Depois de executado este algoritmo, substituímos o valor de cada vértice na *pairList* pelo seu valor no array *id* (que será o número do vértice com o menor valor da sua SCC) e, após efetuar este passo, reordenamos a *pairList* utilizando o mesmo método que referimos no segundo parágrafo (quicksort).

Para finalizar o programa, o output será o número de SCCs encontradas, o número de ligações entre estas e uma lista dessas ligações.

Análise Teórica

Sendo V o número de vértices e E o número de arcos do grafo, estima-se que a complexidade total da solução utilizada seja linear:

→ complexidade temporal: O(V+E)

→ complexidade espacial: O(V)

Em detalhe, temos as seguintes complexidades temporais para as diversas funções e passos do programa:

O(V) para a inicialização da array de vértices, *vertexes*, e para a inicialização da array identificador da SCC de cada vértice, *id*.

O(E logE) para a função *qsort* (*quicksort*).

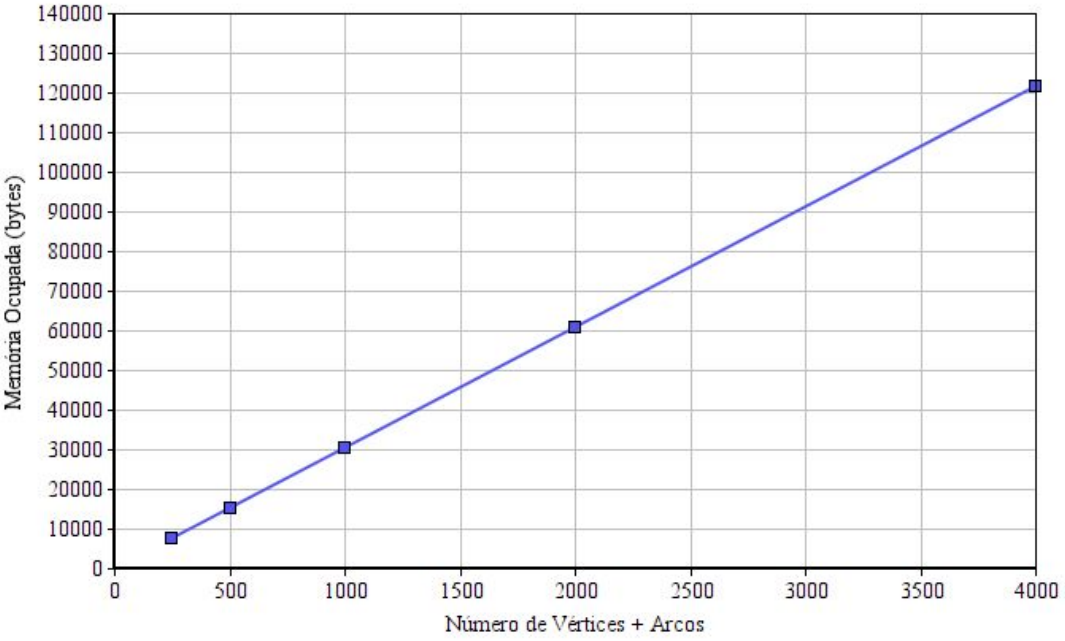
O(V+E) para a função *tarjan*.

O(E) para a função *pair_scc*, uma vez que tem de percorrer todos os arcos da rede de distribuição e para as funções *print_nr_lig* e *print_ligs* pelo mesmo motivo.

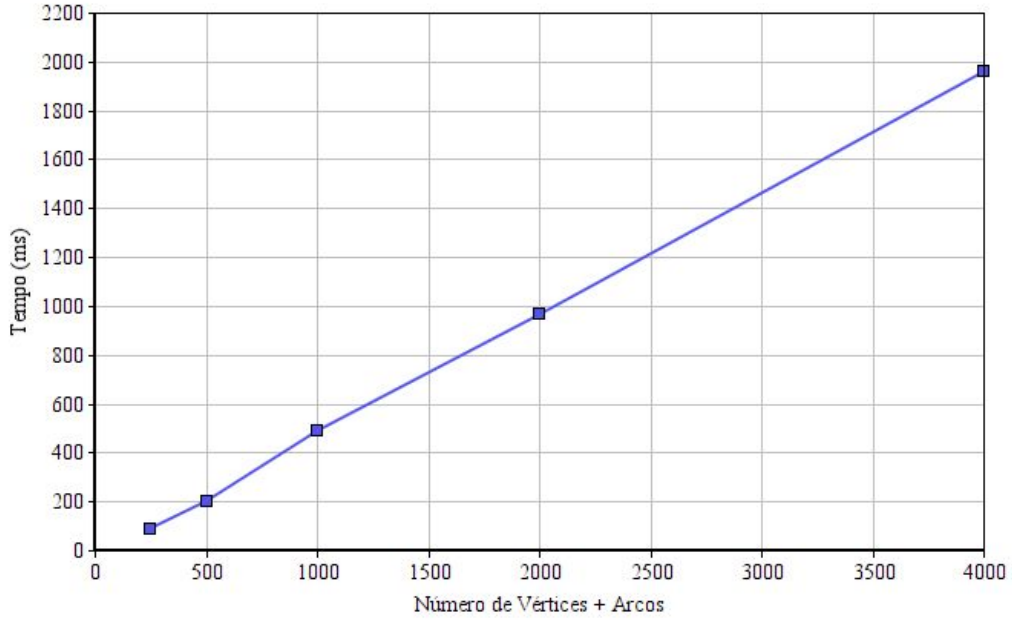
Avaliação Experimental dos Resultados

A plataforma de testes foi um computador com processador Intel® Core™ i7-7700HQ CPU @ 2.80GHz com 16GB de memória a correr o Windows 10.

Executámos o programa com os ficheiros de input disponibilizados na página da cadeira, utilizando as funcionalidades *time* e *Valgrind* do Linux para obter o tempo (em ms) e a memória ocupada por cada teste (em bytes), respectivamente.



Pelo gráfico podemos concluir que o algoritmo desenvolvido é, de facto, linear, dado que a memória ocupada cresce linearmente com o número de vértices e arcos.



Com base no gráfico, constatamos que o tempo de execução e o valor de V+ E são linearmente proporcionais, ou seja, à medida que o número de vértices e arcos de um grafo aumenta, o tempo de execução aumenta proporcionalmente. Assim, é possível verificar que a complexidade do algoritmo é O(V+E).

Valores experimentais usados na elaboração dos gráficos:

V + E (milhares)	Bytes (milhares)	Tempo (ms)
250	7,608	90
500	15,208	201
1,000	30,408	489
2,000	60,808	965
4,000	121,608	1959

Referências

Os websites/obras consultados para a realização do projeto foram os seguintes:

Introduction to Algorithms, Third Edition: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein September 2009 ISBN-10: 0-262- 53305-7; ISBN-13: 978-0-262-53305-8

Tarjan's strongly connected components algorithm: wikipedia

Valgrind user manual - 9. Massif: a heap profiler: valgrind

Detect memory leaks with Memcheck tool provided by Valgrind: ibm