Análise e Síntese de Algoritmos

2018/2019

2º Projeto

Grupo AL008

Daniel Gonçalves – 91004

Gabriel Almeida – 89446

Introdução:

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

Este projeto consiste numa rede de transporte, constituída por fornecedores, postos de abastecimento e por um hipermercado. Numa ligação entre cada dois vértices (podendo ser fornecedor, posto de abastecimento ou hipermercado) consegue-se definir uma capacidade que consiste na quantidade máxima de mercadoria que pode ser transportada nessa ligação. Os postos de abastecimento têm uma quantidade máxima que conseguem movimentar, assim como os fornecedores têm uma quantidade máxima que conseguem produzir.

O programa consegue obter a maior quantidade de mercadoria que a rede consegue transportar e os postos de abastecimento e os caminhos que devem ser melhorados de forma a aumentar a capacidade da rede, através de um *input* do número de fornecedores, do número de postos de abastecimento, das capacidades de cada fornecedor e de cada posto de abastecimento e as ligações entre cada, com a devida capacidade.

Descrição da solução:

O programa foi implementado em linguagem C.

Desenvolvemos duas estruturas para a utilização destes dados, uma lista ligada em que cada elemento da própria contém as informações de cada vértice (correspondente a um fornecedor, posto de abastecimento ou hipermercado), assim como as ligações que dele partem e a ele chegam, criando assim uma lista de adjacências, e outra lista ligada para utilizarmos como fila (*queue*).

Para resolvermos o problema, adicionamos um novo vértice (*source*), cujas ligações eram apenas para os fornecedores e a capacidade da ligação corresponde à capacidade do fornecedor. Adicionamos também vértices u', tal que o vértice u é posto de abastecimento, e de tal forma que a capacidade de um posto de abastecimento (u) corresponde à ligação entre esse posto e o novo

vértice (u'). Portanto as ligações que começam nesse posto (u) são alteradas de forma a que comecem no novo vértice (u').

Para calcular o fluxo máximo, utilizamos o algoritmo Push-Relabel com uma fila. Este algoritmo utiliza métodos de pré-fluxo, começando por inicializar a *source* com o simétrico da soma das capacidades dos arcos que começam na *source* (o que equivale à soma da capacidade dos fornecedores) e assim, o fluxo entre as ligações do *source* aos fornecedores é máximo e os fornecedores possuem excesso igual à sua capacidade original. De seguida, vai a cada vértice que tem excesso e tenta escoar esse excesso para os outros vértices, precisando assim de subir a altura desse vértice, de modo a estar mais alto que o vértice para o qual pretendemos escoar o excesso. Assim sendo, como não tentamos escoar o excesso do *target* (neste caso o Hipermercado), o fluxo máximo terá de ser igual ao excesso do *target*.

De seguida, para calcularmos quais as ligações e postos de abastecimento que devemos melhorar, calculamos qual o corte mínimo mais próximo do *target* (o Hipermercado). Para tal, invertemos o sentido de todos os arcos da rede residual. Assim, conseguimos utilizar uma BFS para marcar todos os vértices que conseguimos atingir através da rede residual, partindo do *target*. Seguidamente, escolhemos as ligações que partem dum vértice atingível para um vértice não-atingível e essas ligações fazem obrigatoriamente parte do corte mínimo, que será o mais próximo do *target*, pois a BFS utiliza os caminhos mais curtos. Por fim, verificamos se essa ligação pode ser melhorada, sendo de um posto de abastecimento, entre fornecedores e postos, entre postos e postos ou entre postos e o hipermercado.

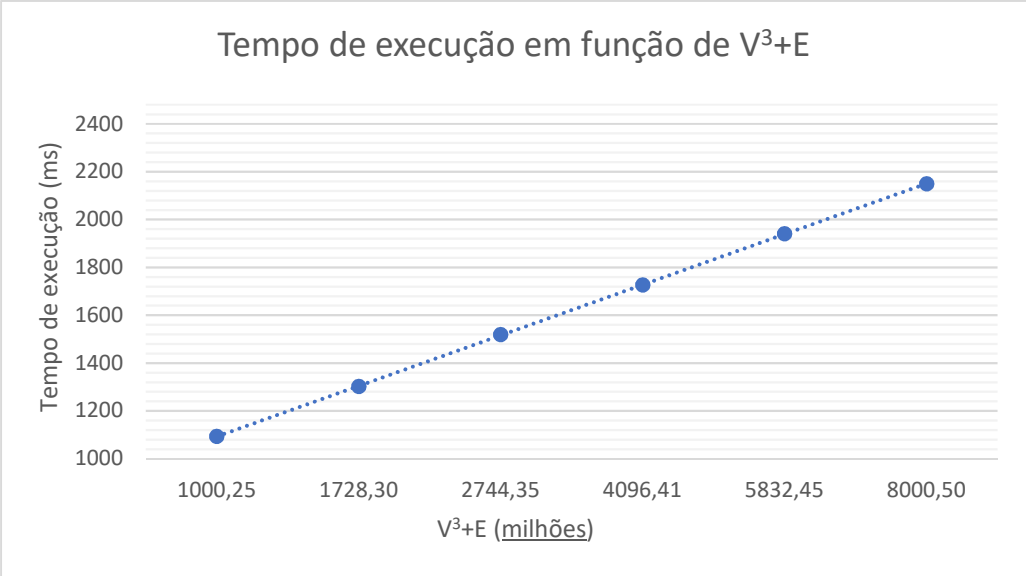
Análise Teórica:

Sendo que V é o número de vértices (número de fornecedores, postos de abastecimento e o hipermercado) e E o número de ligações, calculamos que a complexidade temporal do primeiro algoritmo (Relabel-To-Front) seja $O(V^3)$ e a do segundo (BFS) seja O(V+E). Tendo o programa uma complexidade de $O(V^3+E)$.

Avaliação Experimental dos Resultados:

Para estas experiências utilizamos um computador com processador Intel Core i7-4700MQ 2.7GHz, com 8GB de memória, com o sistema operativo Ubuntu 18.1.

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^3+E, com valores entre 1 milhões e 9 milhões de V^3+E.



V	1000	1200	1400	1600	1800	2000
E (milhares)	251	301.2	351.4	410.6	451.8	502
V^3+E (milhões)	1000.25	1728.30	2744.35	4096.41	5832.45	8000.50
Tempo (ms)	1093	1301	1519	1726	1941	2148

Como se pode verificar no gráfico, a complexidade do programa é linear a $O(V^3+E)$.

Referências:

Utilizamos a seguinte referência para a realização do projeto:

- **Introduction to Algorithms, Third Edition**: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein