Análise e Síntese de Algoritmos

2015/2016

Relatório do Segundo Projeto

Grupo 40

81186 - Stéphane Duarte | 81858 - João Oliveira

INTRODUÇÃO

No âmbito da cadeira de Análise e Síntese de Algoritmos, foi-nos proposto um projeto cujo objetivo é descobrir qual a melhor localidade para organizar um encontro de uma empresa, isto é, o ponto de encontro em que a perda total da empresa, considerando todas as filiais da empresa e os respetivos custos nas deslocações entre aldeias, é minimizada.

Deste modo, o problema vai ser encarado como um grafo dirigido pesado, no qual terá de ser aplicado um algoritmo de procura do caminho mais curto a partir de cada vértice correspondente a uma filial.

O valor dos arcos representa a subtração da receita efetuada ao valor perdido na viagem, ou seja, quanto menor for o valor dos arcos, maior será a receita para a empresa. No fim da execução do algoritmo, será escolhida a localidade cuja somatória do custo da viagem de todas as filiais até si seja menor, preferencialmente negativo, pois significaria obtenção de receita.

DESCRIÇÃO DA SOLUÇÃO

Este problema é encarado como um grafo dirigido pesado, em que cada vértice representa uma localidade ou uma filial e cada arco representa uma ligação entre filiais e/ou localidades (o qual tem o valor do custo de cada ligação).

A solução foi implementada em C, com recurso à implementação de uma estrutura para o grafo e outra para os arcos. A estrutura do grafo é constituída por um número de vértices (V), um número de arcos (E), e um *array* de arcos do grafo. Cada arco é representado pela sua origem (u), pelo seu destino (v) e pelo seu peso (w). A função *graphInit* é a função que permite inicializar o grafo.

O algoritmo calcula, então, o caminho mais curto de cada filial a cada localidade, devolvendo o respetivo custo. Neste caso, recorremos ao algoritmo de Bellman-Ford, já que é possível existirem arcos com pesos negativos.

Como input, é recebido:

1. Uma linha com 3 valores:
    a. O primeiro valor corresponde ao número de localidades (N), que determina o número de vértices do grafo.
    b. O segundo valor corresponde ao número de filiais (F), um número que varia entre 1 e N.
    c. O terceiro valor corresponde ao número de ligações entre localidades (C), que determina o número de arestas do grafo.
2. Uma linha com F valores inteiros correspondentes à identificação das filiais;
3. C linhas com 3 valores inteiros (u, v, w), onde u e v variam entre 1 e N, correspondendo assim à localidade de origem e à localidade de destino (origem e destino do arco), respetivamente, e w posso assumir qualquer valor inteiro (positivo, zero ou negativo), correspondendo ao custo da viagem (peso do arco).

Os valores obtidos em 1.a) e 1.c) são utilizados na criação do grafo. O valor obtido em 1.b) é utilizado para guardar a identificação das filiais, input recebido através do ponto 2. São ainda criados um vetor *custo* de tamanho N, para guardar o custo do caminho mais curto de uma filial a cada localidade no decorrer do algoritmo, um vetor *custototal* de tamanho N para guardar o somatório dos custos dos caminhos mais curtos das filiais às aldeias e uma matriz FxN para guardar todos estes valores. Através do ponto 3 são criadas C arestas no grafo, com os respetivos valores de origem, destino e peso.

É, de seguida, aplicado F vezes o algoritmo de Bellman-Ford ao grafo, que recebe um identificador de uma filial como vértice *source* diferente em cada uma das execuções. Em cada iteração o algoritmo calcula o caminho mais curto da filial a todas as localidades, e preenche o respetivo custo no vetor *custo*. Este custo é calculado com a seguinte fórmula: se o custo até à origem do arco existe e, somado ao peso do arco, é menor que o custo já calculado para o destino, então o custo do destino é o custo da origem mais o peso do arco. Para isto, percorre cada arco N-1 vezes. O algoritmo verifica ainda em cada uma destas iterações se o valor dos arcos foi relaxado. Se isto não se verificar em nenhum dos casos, então o algoritmo termina o ciclo. Isto permite otimizar o tempo de execução. No fim de cada execução do algoritmo a uma respetiva filial, são atualizados o vetor *custototal* ao qual são somados os valores do vetor *custo* e a matriz para onde são copiados os valores do vetor *custo* para a respetiva linha da filial que foi alvo do algoritmo.

Dados todos os passos anteriormente descritos como terminados, é determinado, através do vetor *custototal* qual a localidade cujo custo é menor.

Como output, é dado:

1. Uma linha com o identificador da localidade cuja soma do custo da deslocação de todas as filiais à mesma é o menor possível e o respetivo custo, separados por um espaço em branco.
2. Uma linha com os custos das deslocações de cada filial até à localidade elegida, apresentados pela mesma ordem do input e separados por um espaço.

NOTA: No caso de não ser possível arranjar um ponto de encontro, o output consiste numa única linha com a letra N.

DESCRIÇÃO DA SOLUÇÃO

Após cuidada análise ao código, é possível determinar o tempo de execução de cada ciclo e, consequentemente, do programa. Considere-se N o número de vértices do grafo, F o número de filiais e C o número de arcos.

1. Preenchimento do vetor *filiais*: O(F);
2. Ciclo para obter os arcos: O(C);
3. Ciclo para executar o algoritmo Bellman-Ford: O(N);
4. Bellman-Ford: O(NC);
    a. Inicialização do vetor *custo*: O(N);
    b. Ciclo que percorre todos os arcos N-1 vezes: O(NC);
5. Ciclo para atualizar a matriz: O(NF);
6. Verificação do custo mínimo: O(NF);
7. Impressão de resultados: O(F);

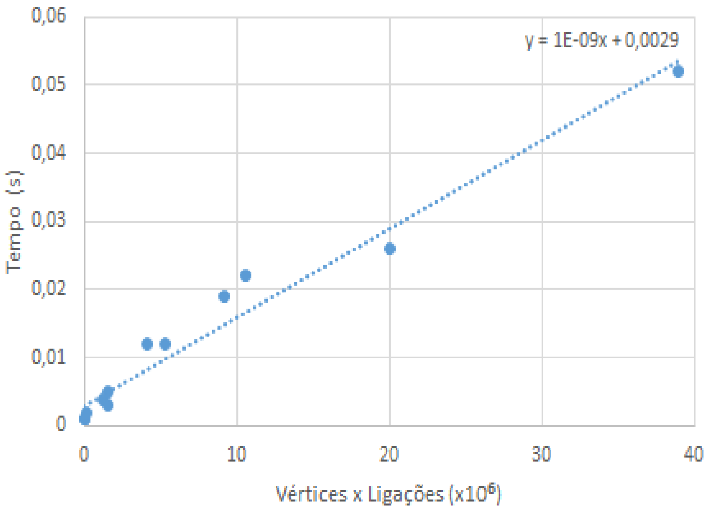
Tendo em conta as complexidades apresentadas anteriormente, concluímos que a complexidade total do programa será: F + C + N + N + NC + NF + NF + F = O(NC).[1]

[1] Tem-se em conta que F é significativamente menor que N.

AVALIAÇÃO EXPERIMENTAL DOS RESULTADOS

Casos de teste:

N	C	Tempo(s)
2	2	0.001
5	6	0.001
5	8	0.001
6	11	0.001
100	800	0.002
100	1000	0.002
250	5000	0.004
300	5000	0.005
500	2975	0.003
800	5080	0.012
100	5262	0.012
1250	7356	0.019
1500	7019	0.022
2000	10000	0.026
3000	13000	0.052



Como anteriormente referido, a complexidade do programa é, teoricamente, O(NC). Foram então feitos testes experimentais para verificar este valor.

Após 14 testes, foi verificado que, à medida que a multiplicação de vértices(N) pelas ligações(C) aumentava, o tempo de execução aumentava linearmente, como se pode ver no gráfico acima.

Deste modo, fica comprovado que a complexidade do programa é, de facto, O(NC).

REFERÊNCIAS

http://www.geeksforgeeks.org/dynamic-programming-set-23-bellman-ford-algorithm/

Introduction to Algorithms, Third Edition: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein September 2009