Multiplicação de matrizes

Este é um algoritmo famoso na matemática. Dadas as matrizes A (de i linhas por j colunas) e B (de k linhas por m colunas), onde $j = k$, obter a matriz C, multiplicação de A por B, de i linhas por m colunas, a partir de $C[i][m] = \sum_{x=1}^{j} A[i][x] \times B[x][m]$

Para implementá-lo, tem-se o

```
 1: função MM (A[i][j], B[j][m]:real) : (C[i][m]:real)
 2: inteiro AUX
 3: inteiro x1, x2, x3
 4: para x1 = 1 até i
 5:     para x2 = 1 até m
 6:         AUX ← 0
 7:         para x3 = 1 até j
 8:             AUX ← AUX + A[x1][x3] × B[x3][x2]
 9:         fim{para}
10:         C[x1][x2] ← AUX
11:     fim{para}
12: fim{para}
13: devolva C
14: fim{função}
```

Como se pode ver no algoritmo, o número de colunas de A precisa ser igual ao número de linhas de B, se esta igualdade não ocorrer a multiplicação não pode ser realizada. O tamanho da matriz resultado da multiplicação é o número de linhas de A e o número de colunas de B. Acompanhe no quadro

matriz A	matriz B	matriz $A \times B$
i, j	j, k	i, k

Antes de continuar com este exercício, implemente a multiplicação entre estas duas matrizes: A, formada por (5 linhas por 6 colunas)

```
13   5   6  46  48  26
74  27  87  55  52  34
18  44  11  38  12  36
23  92  66  60  61  78
40  71  90  72   7  82
```

e a matriz B, formada por (6 linhas por 8 colunas)

```
88  94  43  75  90  89  28  68
32  56   8  25  64  12  35  83
41  48  29  23  78  50  11  44
27  61  16   3   2  69  79   6
99  55  30  17  81  39  15  71
62  47  82  22  98  38  33  52
```

A multiplicação dessas duas matrizes deve ser (5 linhas por 8 colunas)

```
 9156  8458  5081  2764  8486  7551  5817  6599
19684 20457 11149 10023 22828 18375 10221 16891
 7889  9354  5365  3813  9870  7138  6535  8312
20169 21163 12825  8476 25811 15934 12819 20851
17203 20687 12984  8984 23911 17269 13094 17766
```

Se definirmos uma operação elementar neste procedimento como sendo a multiplicação de um elemento de A por outro de B e depois a soma desse produto em uma variável de trabalho, pode-se afirmar que a quantidade de operações é $i \times j \times k$ ou mais resumidamente como sendo $O(ijk)$. No exemplo acima, foram $5 \times 6 \times 8$ operações ou 240 operações.

Múltiplas multiplicações matriciais

Agora suponha a existência de uma série de multiplicações, como por exemplo

$$M_{1[10,100]} \times M_{2[100,1000]} \times M_{3[1000,3]}$$

Como a multiplicação matricial é associativa, tanto faz, realizar a multiplicação como

$$(M_{1[10,100]} \times M_{2[100,1000]}) \times M_{3[1000,3]}$$

ou como

$$M_{1[10,100]} \times (M_{2[100,1000]} \times M_{3[1000,3]})$$

entretanto, a quantidade de trabalho, nem de longe é a mesma. Em $(M_{1[10,100]} \times M_{2[100,1000]}) \times M_{3[1000,3]}$ são necessárias $10 \times 100 \times 1000 = 1.000.000$ de operações dentro do parênteses e depois mais 300.000 operações fora do parênteses, totalizando 1.300.000 operações.

Entretanto, na segunda parentização, as operações são 300.000 dentro do parênteses e mais 3.000 operações totalizando 303.000 operações, que é muito menos do que 1.300.000.

Veja um segundo exemplo em

$$M = M_{1[10,20]} \times M_{2[20,50]} \times M_{3[50,1]} \times M_{4[1,100]}$$

onde as dimensões de cada matriz estão entre colchetes.

Calculando a expressão como $M_1 \times (M_2 \times (M_3 \times M_4))$ são necessárias 125.000 operações, enquanto calculando $(M_1 \times (M_2 \times (M_3)) \times M_4$ são requeridas apenas 2.200 operações.

A avaliação de todas as possibilidades de operações entre as matrizes é um processo exponencial, como se pode ver em

$$X_{n+1} = \frac{1}{n+1} \left(\begin{array}{c} 2n \\ n \end{array} \right)$$

onde $X(n)$ é o número de maneiras que se pode parentizar um conjunto de n símbolos. Este número cresce e se torna impraticável quando n é grande, veja na tabela a seguir:

1 matriz	não há o que calcular
2 matrizes	1 possibilidade
3 matrizes	2 possibilidades
4 matrizes	5
5	14
6	42
7	132
8	429
9	1.430
10	4.862
15	2.674.440
20	1.767.263.190
30	1.002.242.216.651.368
40	680.425.371.729.975.700.000

Usando a programação dinâmica pode-se estudar o comportamento do problema através de um algoritmo $O(n^3)$. Seja m_{ij} o custo mínimo de computar $M_i \times M_{i+1} \times ... \times M_j$ para $1 \leq i \leq j \leq n$. Claramente $m_{ij} = 0$ se $i = j$ e $m_{ij} =$ o menor dos valores de $m_{ik} + m_{k+1,j} + r_{i-1} \times r_k \times r_j$ se $j > i$ variando k nos limites $i \leq k < j$. Aqui r_k é o $k - esimo$ elemento no vetor de dimensões das matrizes.

O termo m_{ik} é o custo mínimo de avaliar $M' = M_i \times M_{i+1} \times ... \times M_k$.

O segundo termo $m_{k+1,j}$ é o custo mínimo de avaliar $M'' = M_{k+1} \times M_{k+2} \times ... \times M_j$.

O terceiro termo é o custo de multiplicar M' por M''. Note que M' é uma matriz $r_{i-1} \times r_k$ e M'' é uma matriz $r_k \times r_j$.

A equação acima determina que m_{ij}, com $j > i$ é o valor mínimo obtido de todos os valores possíveis de k entre i e $j - 1$ da soma desses 3 termos.

A estratégia da programação dinâmica calcula os $m'_{ij}s$ na ordem em que a diferença dos subscritos vai sendo incrementada. Começa-se calculando m_{ii} para todo i. Depois $m_{i,i+1}$ para todo i, depois $m_{i,i+2}$ e assim por diante. Desta forma, os termos m_{ik} e $m_{k+1,j}$ da fórmula acima já estarão disponíveis quando m_{ij} for ser calculado. Veja-se o algoritmo

```
 1: função avaliamats v
 2: n ← tamanho(v)-1
 3: m ← matriz n linhas × n colunas ← 0
 4: q ← matriz n linhas × n colunas ← 0
 5: para h ← 1 até n − 1
 6:     para i ← 1 até n − h
 7:         j ← i + h
 8:         m[i, j] ← ∞
 9:         para k ← i até j − 1
10:             temp ← m[i, k] + m[k + 1, j] + (v[i] × v[1 + k] × v[1 + j])
11:             se temp < m[i, j]
12:                 m[i, j] ← temp
13:                 q[i, j] ← k
14:             fim{se}
15:         fim{para}
16:     fim{para}
17: fim{para}
18: imprima m,q
```

A matriz m contém as quantidades de operações mínimas enquanto a matriz q contém qual operação deve ser feita a cada etapa. A análise de q pode ser feita pelo algoritmo

```
1: função explicaq (q,i,j)
2: se i=j
3:     imprima 'M',i
4: senão
5:     imprima '('
6:     explicaq (q,i,q[i,j])
7:     explicaq (q,(q[i,j]+1),j
8:     imprima ')'
9: fim{se}
```

Para obter a parentização correta, deve-se chamar `explicaq(q,1,6)` (o último número é a quantidade de matrizes).

Eis os resultados do algoritmo para o exemplo acima: (a ← 10 20 50 1 100):

```
0 10000 1200 2200              0 1 1 3
0     0 1000 3000              0 0 2 3
0     0    0 5000              0 0 0 3

((M1(M2M3))M4)
2200 1200 1000
```

Veja mais um exemplo: (c ← 13 25 4 80 1)

```
0 1300 5460 745                0 1 2 1
0    0 8000 420                0 0 2 2
0    0    0 320                0 0 0 3

(M1(M2(M3M4)))
745 420 320
```

Veja-se agora um exemplo maior:
f← 41 90 38 73 3 36 39

```
0 140220 253954 29652 34080 38661   0 1 2 1 4 4
0      0 249660 18582 28302 33324   0 0 2 2 4 4
0      0      0  8322 12426 16980   0 0 0 3 4 4
0      0      0     0  7884 12753   0 0 0 0 4 4
0      0      0     0     0  4212   0 0 0 0 0 5

((M1(M2(M3M4)))(M5M6))
38661 33324 16980 12426 4212
```

☞ Para você fazer

1. Suponha a multiplicação matricial da seguinte sequência de matrizes:

```
M[42x12] X M[12x53] X M[53x20] X M[20x36] X
M[36x11] X M[11x27] X M[27x44] X M[44x10]
```

O objetivo é calcular a sequência de multiplicações que MINIMIZA a quantidade de operações elementares. Assim, implemente os algoritmos acima e responda:
1a. Qual o número de multiplicações elementares do primeiro par de matrizes a ser calculado ?
1b. Qual o número total de operações elementares ao calcular a expressão completa ?

2. Mais um caso,

```
M[29x22] X M[22x75] X M[75x79] X M[79x53] X
M[53x88] X M[88x20] X M[20x65] X M[65x61]
```

2a. Operações do primeiro par ?
2b. Número total de operações ?

1a	1b	2a	2b

Para saber mais:
1. AHO, HOPCROFT, ULLMAN. Alfred, John e Jeffrey **The design and Analysis of Computer Algorithms**. 1974. Addison Wesley, Reading Massachusetts. (UP: 005.12 A286d, pág. 65).
2. ZIVIANI, Nivio. **Projeto de Algoritmos**. 2012. Thomson. (UP: 005.12 Z82p, 3.ed., pág. 51).
3. CORMEN, Thomas, et alli. **Algoritmos**. 2002. Campus. (UP: 005.12 A396alg, pág. 266 a 271).