

# Programação Dinâmica

- Multiplicação de Cadeia de Matrizes.
  - Dada uma cadeia de matrizes,  $A_1A_2\dots A_n$ , com  $n$  matrizes, queremos determinar o produto  $A_1*A_2*...*A_n$  efetuando o menor número de operações aritméticas possível.

A multiplicação de matrizes é associativa, e assim todas as parentizações produzem o mesmo produto. Um produto de matrizes será sempre executado duas matrizes por vez e assim pode ser totalmente parentizado, tanto se for uma única matriz ou se for o produto de dois produtos de sub cadeias de matrizes. Por exemplo, a cadeia de matrizes  $(A_1, A_2, A_3, A_4)$  pode ser totalmente parentizada das seguintes maneiras:

$$(A_1 (A_2 (A_3 A_4) ) ) ,$$

$$(A_1 ( (A_2 A_3) A_4 ) ) ,$$

$$( (A_1 (A_2 A_3) ) A_4 ) ,$$

$$( ( (A_1 A_2) A_3) A_4 ) , \text{ ou}$$

$$( (A_1 A_2) (A_3 A_4) )$$

O algoritmo padrão para cálculo de multiplicação de matrizes abaixo determina um custo de  $A.\text{linhas} * B.\text{colunas} * A.\text{colunas}$  multiplicações de escalar.

```
algoritmo multiplicação_matrizes(A, B)
  se A.colunas != B.linhas então
    erro("dimensões incompatíveis")
  senão
    seja C uma nova matriz A.linhas x B.colunas
    para i = 1 até A.linhas
      para j = 1 até B.colunas
        Cij = 0
        para k = 1 até A.colunas
          Cij = Cij + Aik * Bkj
    retorne(C)
```

Para ilustrar os diferentes custos incorridos por diferentes parentizações de um produto de matriz, considere uma cadeia  $(A_1, A_2, A_3)$  de três matrizes, com as dimensões  $10 \times 100$ ,  $100 \times 5$  e  $5 \times 50$ , respectivamente. Se multiplicarmos de acordo com os parênteses temos:

$((A_1 A_2) A_3)$ :  $(A_1 A_2) \Rightarrow 10 \times 100 \times 5 = 5000$  resultando uma matriz  $10 \times 5$ , que ao multiplicar por  $A_3$  temos mais  $10 \times 5 \times 50 = 2500$ , resultando em 7500 multiplicações de escalares.

$(A_1 (A_2 A_3))$ :  $(A_2 A_3) \Rightarrow 100 \times 5 \times 50 = 25000$  resultando uma matriz  $100 \times 50$ , que ao multiplicar por  $A_1$  temos mais  $10 \times 100 \times 50 = 50000$ , resultando em 75000 multiplicações de escalares.

Enunciamos o problema de multiplicação de cadeias de matrizes da seguinte forma:

**Dada uma cadeia  $(A_1, A_2, \dots, A_n)$  de  $n$  matrizes, onde para  $i = 1, 2, \dots, n$ , a matriz  $A_i$  tem dimensão  $p_{i-1} \times p_i$ , coloque o produto  $A_1 * A_2 * \dots * A_n$  entre parênteses de forma a minimizar o número de multiplicações escalares.**

## 1- Caracterizar uma subestrutura ótima:

Como a multiplicação de matrizes é feita duas a duas, a multiplicação de uma cadeia  $A_1 \dots A_n$ , de qualquer forma que for feita terá como última operação a multiplicação de um par de dois produtos de subcadeias, por exemplo  $(A_1 \dots A_k) * (A_{k+1} \dots A_n)$ , logo podemos imaginar que os dois produtos de subcadeias  $A_1 A_k$  e  $A_{k+1} A_n$  sendo os que necessitam menor número de multiplicações de escalares para serem computados resultarão na maneira mais barata de fazer a multiplicação de toda a cadeia. Ou seja, **A melhor maneira de multiplicar uma cadeia de matrizes é multiplicando seus dois melhores subprodutos, que são, recursivamente, seus dois melhores subprodutos.**

## 2- Uma Solução Recursiva

Para o problema de multiplicação de uma cadeia de matrizes, escolhamos como nossos subproblemas determinar o custo mínimo de colocar entre parênteses

$$A_i A_{i+1} \dots A_j \text{ para } 1 \leq i \leq j \leq n.$$

**Seja  $m[i, j]$  o número mínimo de multiplicações escalares** necessárias para calcular a **matriz  $A_{i..j}$** ; para o problema completo, a maneira de menor custo para calcular  $A_{1..n}$  seria, portanto,  $m[1, n]$ .

Podemos **definir  $m[i, j]$**  recursivamente como segue:

**Se  $i = j$** , o problema é trivial; a cadeia consiste em apenas uma matriz  $A_{i..i} = A_i$ , de modo que nenhuma multiplicação escalar é necessária para calcular o produto. Assim,  **$m[i, i] = 0$**  para  $i = 1, 2, \dots, n$ .

Para calcular  **$m[i, j]$  quando  $i < j$** , aproveitamos a estrutura de uma solução ótima da etapa 1. Vamos supor que, para colocar parênteses de forma ótima, dividimos o produto  $A_i A_{i+1} \dots A_j$  entre  $A_k$  e  $A_{k+1}$ , onde  $i \leq k < j$ . Então,  **$m[i, j]$  é igual ao custo mínimo para calcular os subprodutos  $A_{i..k}$  e  $A_{k+1..j}$** , mais o custo de multiplicar essas duas matrizes juntas. Lembrando que cada matriz  $A_i$  é  $p_{i-1}$  por  $p_i$ , vemos que calcular o produto matricial  $A_{i..k} A_{k+1..j}$  leva  $p_{i-1} p_k p_j$  multiplicações escalares. Assim, obtemos

$$m[i, j] = m[i, k] + m[k+1, j] + p_{i-1} p_k p_j$$

algoritmo multCadeiaMatrizes(Ai..j)

se  $i=j$  então retorne (0)

senão

$m = \infty$

para  $k=i$  até  $j$  faça

$q = \text{multCadeiaMatrizes}(A_{i..k}) +$   
 $\text{multCadeiaMatrizes}(A_{k+1..j}) +$   
 $p_{i-1} * p_k * p_j$

se  $q < m$  então

$m = q$

retorne (  $\min(\text{multCadeiaMatrizes}(A_{i..k}) +$

### 3- Uma implementação bottom-up

Implementaremos o método tabular, bottom-up, no procedimento MATRIX-CHAIN-ORDER, que aparece abaixo. Este procedimento assume que a matriz  $A_i$  tem dimensões  $p_{i-1}, p_i$  para  $i = 1, 2, \dots, n$ . Sua entrada é uma sequência  $p = (p_0, p_1, \dots, p_n)$ , onde  $p.length = n + 1$ . O procedimento usa uma tabela auxiliar  $m[1..n, 1..n]$  para armazenar os custos  $m[i, j]$  e outra tabela auxiliar  $s[1..n-1, 2..n]$  que registra o índice de  $k$  que atingiu o custo ótimo no cálculo de  $m[i, j]$ . Usaremos a tabela  $s$  para construir uma solução ótima.

```

algoritmo MATRIX-CHAIN-ORDER(p)
  n = p.length+1
  sejam m[1.. n, 1.. n] e s[1.. n-1, 2..n] novos vetores
  para i = 1 até n
    m[i, i] = 0
  fim_para
  para l = 2 até n
    para i = 1 até n - l + 1
      j = i + l - 1
      m[i, j] = ∞
      para k = i até j - 1
        q = m[i, k] + m[k+1, j] + pi-1*pk*pj
        se q < m[i, j] então
          m[i, j] = q
          s[i, j] = k
      fim_se
    fim_para
  fim_para
  retorne(m e s)
fim_algoritmo

```

#### Etapa 4: Construindo uma solução ótima

Embora MATRIX-CHAIN-ORDER determine o número ótimo de multiplicações escalares necessárias para calcular um produto de cadeia de matrizes, ele não mostra diretamente como multiplicar as matrizes. A tabela  $s[1..n-1, 2..n]$  nos dá as informações de que precisamos para fazer isso. Cada entrada  $s[i,j]$  registra um valor de  $k$  tal que uma parentização ótima de  $A_i A_{i+1} \dots A_j$  divide o produto entre  $A_k$  e  $A_{k+1}$ . Assim, sabemos que a multiplicação final da matriz no cálculo de  $A_{1..n}$  de forma ótima é  $A_{1..s[1,n]} A_{s[1,n]+1..n}$ . Podemos determinar as multiplicações de matriz anteriores recursivamente, uma vez que  $s[1, s[1, n]]$  determina a última multiplicação de matriz ao calcular  $A_{1..s[1,n]}$  e  $s[s[1,n]+1, n]$  determina a última multiplicação de matriz ao calcular  $A_{s[1,n]+1..n}$ . O procedimento recursivo a seguir imprime uma parentização ótima de  $(A_i, A_{i+1}, \dots, A_j)$ , dada a tabela  $s$  computada por MATRIX-CHAIN-ORDER e os índices  $i$  e  $j$ . A chamada inicial PRINT-OPTIMAL-PARENS( $s, 1, n$ ) imprime uma parentização ótima de  $(A_1, A_2, \dots, A_n)$ .

```
algoritmo PRINT-OPTIMAL-PARENS(s, i, j)
  se i == j então
    escreva "A"i
  senão
    escreva "("
    PRINT-OPTIMAL-PARENS(s, i, s[i,j])
    PRINT-OPTIMAL-PARENS(s, s[i, j]+1, j)
    escreva ")"
  fim_se
fim_algoritmo
```