Programação dinâmica
Condições de aplicabilidade

A programação dinâmica aplica-se a problemas que apresentam as características seguintes:

Subestrutura óptima (*Optimal substructure*)

- Um problema tem subestrutura óptima se uma sua solução óptima é construída com recurso a soluções óptimas de subproblemas

Subproblemas repetidos (*Overlapping subproblems*)

- Existem subproblemas repetidos quando os subproblemas de um problema têm subproblemas em comum

Programação dinâmica
Aplicação

1. Caracterização de uma solução óptima
2. Formulação recursiva do cálculo do valor de uma solução óptima
3. Cálculo iterativo do valor de uma solução óptima, por tabelamento
4. Construção de uma solução óptima

Produto de matrizes

Cálculo do produto de uma sequência de matrizes (*Matrix-chain multiplication*)

Problema

Dada uma sequência de matrizes a multiplicar

$$A_1\, A_2\, \ldots\, A_n, \quad n > 0$$

com dimensões

$$p_0 \times p_1 \quad p_1 \times p_2 \quad \ldots \quad p_{n-1} \times p_n$$

por que ordem efectuar os produtos de modo a minimizar o número de multiplicações entre elementos das matrizes?

(NOTA 1: A matriz A_i tem dimensão $p_{i-1} \times p_i$)

(NOTA 2: O produto de matrizes é uma operação associativa)

Produto de matrizes

Cálculo do produto de duas matrizes (1)

$$A\ (p \times q) \quad \times \quad B\ (q \times r) \quad = \quad C\ (p \times r)$$

$$\begin{array}{cc} & \\ i & \begin{bmatrix} a_{i1} & a_{i2} & \ldots & a_{iq} \end{bmatrix} \end{array} \times \begin{array}{c} j \\ \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{qj} \end{bmatrix} \end{array} = \begin{array}{cc} & j \\ i & \begin{bmatrix} c_{ij} \end{bmatrix} \end{array}$$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{iq}b_{qj} = \sum_{k=1}^{q} a_{ik}b_{kj}$$

No cálculo de cada elemento de C, são efectuadas q multiplicações (escalares)

Produto de matrizes

Cálculo do produto de duas matrizes (2)

MATRIX-MULTIPLY(A[1..p, 1..q], B[1..q, 1..r])

```
let C[1..p,1..r] be a new matrix
for i <- 1 to p do
  for j <- 1 to r do
    C[i,j] <- 0
    for k <- 1 to q do
      C[i,j] <- C[i,j] + A[i,k] * B[k,j]
return C
```

Número de multiplicações

Se A e B são matrizes com dimensões $p \times q$ e $q \times r$, respectivamente, no cálculo de $C = AB$, o número de multiplicações efectuadas entre elementos das matrizes é

$$p \times q \times r$$

(C tem $p \times r$ elementos e são efectuadas q multiplicações para o cálculo de cada um)

Produto de uma sequência de matrizes

Exemplo

Sejam A_1, A_2 e A_3 matrizes com dimensões

$$10 \times 100,\ 100 \times 5 \text{ e } 5 \times 50$$

Ordens de avaliação possíveis para o produto $A_1 A_2 A_3$

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

Número de multiplicações

$(A_1 A_2) A_3$

$$10 \times 100 \times 5 + 10 \times 5 \times 50 = 5000 + 2500 = 7500$$

$A_1 (A_2 A_3)$

$$100 \times 5 \times 50 + 10 \times 100 \times 50 = 25000 + 50000 = 75000$$

Produto de uma sequência de matrizes

Colocação de parêntesis

Formulação alternativa

Como colocar parêntesis no produto $A_1 A_2 \ldots A_n$ de modo a realizar o menor número de multiplicações possível?

Número de colocações de parêntesis distintas

$$\Omega\left(\frac{4^n}{n^{\frac{3}{2}}}\right)$$

Produto de uma sequência de matrizes

Caracterização de uma solução óptima (1)

O produto $A_1 A_2 \dots A_n$ será calculado de uma das formas

$$\begin{array}{l} A_1\,(A_2 \dots A_n) \\ (A_1 A_2)\,(A_3 \dots A_n) \\ (A_1 \dots A_3)\,(A_4 \dots A_n) \\ \qquad\vdots \\ (A_1 \dots A_{n-2})\,(A_{n-1} A_n) \\ (A_1 \dots A_{n-1})\,A_n \end{array}$$

O número m de multiplicações a efectuar para o cálculo de

$$(A_1 \dots A_k)\,(A_{k+1} \dots A_n)$$

para qualquer $1 \leq k < n$, será

$$\mathrm{m}(A_1 \dots A_k) + \mathrm{m}(A_{k+1} \dots A_n) + p_0\, p_k\, p_n$$

Produto de uma sequência de matrizes

Caracterização de uma solução óptima (2)

Procura-se o valor mínimo de

$$\mathrm{m}(A_1 \dots A_n)$$

que depende do valor mínimo de

$$\mathrm{m}(A_1 \dots A_k) \quad \text{e de} \quad \mathrm{m}(A_{k+1} \dots A_n)$$

para algum valor de k

O número mínimo m de multiplicações a efectuar será obtido para o valor de k que minimiza

$$m(A_1 \dots A_k) + m(A_{k+1} \dots A_n) + p_0\, p_k\, p_n$$

Produto de uma sequência de matrizes

Função recursiva

Sequência de matrizes a multiplicar

$$A_1\, A_2\, \ldots\, A_n, \quad n > 0$$

Dimensões das matrizes: $P = (p_0\, p_1 \ldots p_n)$

$m_P(1..n, 1..n)$: $m_P(i, j)$ é o menor número de multiplicações a fazer para calcular o produto $A_i \ldots A_j$

$$m_P(i,j) = \begin{cases} 0 & \text{se} \quad i = j \\ \min\limits_{i \leq k < j} \{ m_P(i,k) + m_P(k+1, j) + p_{i-1} p_k p_j \} & \text{se} \quad i < j \end{cases}$$

Chamada inicial da função

Nº mínimo de multiplicações para a sequência completa: $m_P(1, n)$

Produto de uma sequência de matrizes

Cálculo de $m[i, j]$

m

	1	2	3	4	5
1	0	m_{12}	m_{13}	m_{14}	m_{15}
2		0	m_{23}	m_{24}	m_{25}
3			0	m_{34}	m_{35}
4				0	m_{45}
5					0

Ordem de cálculo

1. Sequências de comprimento 1: m_{11}, m_{22}, m_{33}, m_{44}, m_{55} (Caso base)
2. Sequências de comprimento 2: m_{12}, m_{23}, m_{34}, m_{45}
3. Sequências de comprimento 3: m_{13}, m_{24}, m_{35}
4. Sequências de comprimento 4: m_{14}, m_{25}
5. Sequências de comprimento 5: m_{15}

Produto de uma sequência de matrizes

Cálculo iterativo de $m[1, n]$

Cálculo por comprimento crescente de sequência

MATRIX-CHAIN-ORDER(P)

```
n <- |P| - 1                           // p[0..n]
let m[1..n,1..n] be a new array
for i <- 1 to n do                     // caso base
     m[i, i] <- 0
for ℓ <- 2 to n do                     // ℓ é o comprimento
    for i <- 1 to n - ℓ + 1 do
        j <- i + ℓ - 1                 // |Ai..Aj| = ℓ
        m[i, j] <- +∞
        for k <- i to j - 1 do
            q <- m[i, k] + m[k + 1, j] +
                 p[i - 1] * p[k] * p[j]
            if q < m[i, j] then
                m[i, j] <- q
return m[1, n]
```

Produto de uma sequência de matrizes

Complexidade de MATRIX-CHAIN-ORDER($p_0\ p_1\ \ldots\ p_n$)

Todas as operações executadas têm custo constante

Ciclo 3–4 é executado n vezes

Ciclo 5–12 é executado $n-1$ vezes (variável ℓ)

Ciclo 6–12 é executado $n-\ell+1$ vezes (variável i)

Ciclo 9–12 é executado $\ell-1$ vezes (variável k)

$$\sum_{\ell=2}^{n} \sum_{i=1}^{n-\ell+1} \sum_{k=i}^{i+\ell-2} 1 = \sum_{\ell=2}^{n} \sum_{i=1}^{n-\ell+1} \ell-1 = \sum_{\ell=2}^{n} (n-(\ell-1))(\ell-1) = \sum_{\ell=1}^{n-1} (n-\ell)\ell =$$

$$n \sum_{\ell=1}^{n-1} \ell - \sum_{\ell=1}^{n-1} \ell^2 = n\frac{(n-1)n}{2} - \frac{(n-1)n(2n-1)}{6} = \frac{n^3-n}{6} = \Theta(n^3)$$

Complexidade temporal $\Theta(n^3)$

Complexidade espacial $\Theta(n^2)$

Produto de uma sequência de matrizes

Construção da solução

MATRIX-CHAIN-ORDER(P)

```
n <- |P| - 1                                // p[0..n]
let m[1..n,1..n] and s[1..n-1,2..n] be new arrays
for i <- 1 to n do                          // caso base
     m[i, i] <- 0
for ℓ <- 2 to n do                          // ℓ é o comprimento
    for i <- 1 to n - ℓ + 1 do
        j <- i + ℓ - 1                      // |Ai..Aj| = ℓ
        m[i, j] <- +∞
        for k <- i to j - 1 do
            q <- m[i, k] + m[k + 1, j] +
                 p[i - 1] * p[k] * p[j]
            if q < m[i, j] then
                m[i, j] <- q
                s[i, j] <- k                // parte na matriz k
return m and s
```

Produto de uma sequência de matrizes

Exemplo

$$P = (10 \quad 100 \quad 5 \quad 50 \quad 3) \qquad n = 4$$

Matriz `m` (multiplicações)

	1	**2**	**3**	**4**
1	0	5000	7500	5250
2		0	25000	2250
3			0	750
4				0

Matriz `s` (separação)

	2	**3**	**4**
1	1	2	1
2		2	2
3			3

Número mínimo de multiplicações para calcular ...

$$\begin{aligned}
A_1A_2 &= 5000 \\
A_2A_3 &= 25000 \\
A_1A_2A_3 &= 7500 \\
A_2A_3A_4 &= 2250 \\
A_1A_2A_3A_4 &= 5250
\end{aligned}$$

Separação dos produtos

$$\begin{aligned}
A_1 \dots A_2 &= (A_1)(A_2) \\
A_1 \dots A_3 &= (A_1A_2)(A_3) \\
A_2 \dots A_4 &= (A_2)(A_3A_4) \\
A_1 \dots A_4 &= (A_1)(A_2 \dots A_4) \\
&= (A_1)(A_2(A_3A_4))
\end{aligned}$$

Produto de uma sequência de matrizes

Melhor colocação de parêntesis

$s[1..n-1, 2..n]$: $s[i,j]$ é a posição onde a sequência $A_i \ldots A_j$ é dividida: $(A_i \ldots A_{s[i,j]})(A_{s[i,j]+1} \ldots A_j)$

PRINT-OPTIMAL-PARENS(s, i, j)

```
if i = j then
  print "A"_i
else
  print "("
  PRINT-OPTIMAL-PARENS(s, i, s[i, j])
  PRINT-OPTIMAL-PARENS(s, s[i, j] + 1, j)
  print ")"
```

Produto de uma sequência de matrizes

Cálculo iterativo de $m[1, n]$ — Variante 1

Cálculo por linhas

MATRIX-CHAIN-ORDER(P)

```
n <- |P| - 1                          // p[0..n]
let m[1..n,1..n] be a new array
for i <- 1 to n do                    // caso base
     m[i, i] <- 0
for i <- n - 1 downto 1 do
    for j <- i + 1 to n do
        m[i, j] <- +∞
        for k <- i to j - 1 do
            q <- m[i, k] + m[k + 1, j] +
                 p[i - 1] * p[k] * p[j]
            if q < m[i, j] then
                m[i, j] <- q
return m[1, n]
```

Produto de uma sequência de matrizes

Cálculo iterativo de $m[1, n]$ — Variante 2

Cálculo por colunas

MATRIX-CHAIN-ORDER(P)

```
n <- |P| - 1                                 // p[0..n]
let m[1..n,1..n] be a new array
for i <- 1 to n do                           // caso base
     m[i, i] <- 0
for j <- 2 to n do
    for i <- j - 1 downto 1 do
        m[i, j] <- +∞
        for k <- i to j - 1 do
            q <- m[i, k] + m[k + 1, j] +
                 p[i - 1] * p[k] * p[j]
            if q < m[i, j] then
                m[i, j] <- q
return m[1, n]
```