Complexidade de Algoritmos

Edson Prestes

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Programação Dinâmica

A programação dinâmica costuma ser aplicada a problemas de otimização resultando, em geral, em algoritmos mais eficientes que os mais diretos.

Esse método é útil quando não é fácil chegar a uma seqüência ótima de decisões sem testar todas as seqüências possíveis para então escolher a melhor.

A cada passo são eliminadas subsoluções que certamente não farão parte da solução ótima do problema.

Ele reduz drasticamente o número total de seqüências viáveis através de um mecanismo que evita aquelas seqüências que sabidamente não podem resultar em seqüências ótimas.

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Problema de Multiplicação de Matrizes

Consiste em determinar a **seqüência ótima** de multiplicações de **n** matrizes

$$M := M_1 \times M_2 \times \ldots \times M_n$$

Sabemos que

$$[A \times B]_{ij} := \sum_{k=1}^{q} A_{ik} \cdot B_{kj} \qquad (\text{para } i = 1, \ldots, p \text{ e } j = 1, \ldots, r)$$

Este cálculo exige **p.q.r** multiplicações.

Considere o seguinte exemplo

$$M := \underset{100 \times 3}{M_1} \times \underset{3 \times 10}{M_2} \times \underset{10 \times 50}{M_3} \times \underset{50 \times 30}{M_4} \times \underset{30 \times 5}{M_5}$$

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

1a. Maneira

$$\{[(M_1 \times M_2) \times M_3] \times M_4\} \times M_5$$

A quantidade de operações é dada por

$$(100 \times 10 \times 3) + (100 \times 10 \times 50) + (100 \times 50 \times 30) + (100 \times 30 \times 5)$$

$$= 218000 \text{ operações}$$

2a. Maneira

$$M_1 \times \{M_2 \times [(M_3 \times M_4) \times M_5,]\}$$

A quantidade de operações é dada por

$$(10 \times 50 \times 30) + (10 \times 30 \times 5) + (3 \times 10 \times 5) + (100 \times 3 \times 5)$$

$$= 18150 \text{ operações}$$

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Para este caso, o algoritmo direto tem **complexidade exponencial no** número de matrizes

Usando a programação dinâmica encontramos um algoritmo de complexidade **polinomial.**

Multiplicação de Matrizes

$$\{[(M_1 \times M_2) \times (M_3 \times M_4)] \times M_5\} \qquad \{(M_1 \times M_2) \times [M_3 \times (M_4 \times M_5)]\}$$

Série de Fibonacci

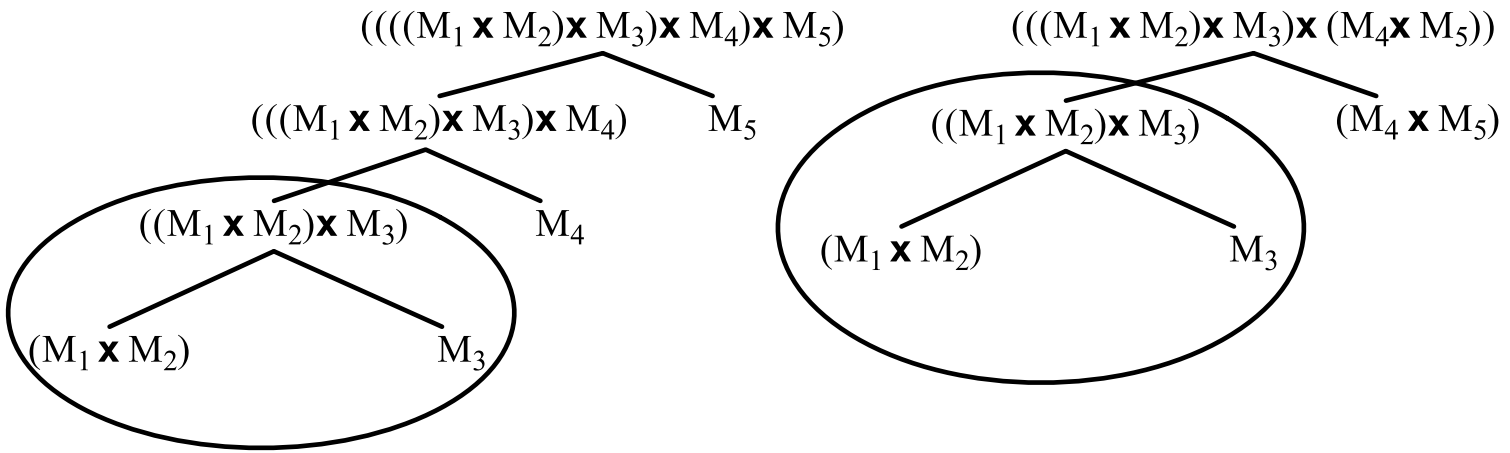
$$f(n) = f(n-1) + f(n-2)$$

$$f(n-1) = f(n-2) + f(n-3) \qquad f(n-2) = f(n-3) + f(n-4)$$

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Multiplicação de Matrizes



((((((M1 x M2)x M3)x M4)x M5)x M6)x M7)

((((M1 x M2)x M3)x M4)x M5)x(M6x M7))

(((M1 x M2)x M3)x (M4x M5))x (M6x M7))

((M1 x (M2x M3))x ((M4x M5)x(M6x M7)))

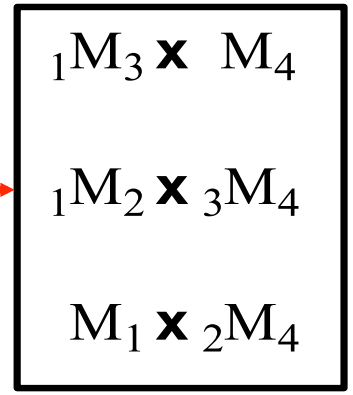
Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Como minimizar ou reduzir a redundância de trabalho ?

Devemos resolver os problemas menores e utilizá-los para resolver os maiores

	$M_1 \times M_2 \times M_3$	$M_2 \times M_3 \times M_4$	$M_1 \times M_2 \times M_3 \times M_4$
$(M_1 \times M_2)$	$(M_1 \times M_2) \times M_3$	$(M_2 \times M_3) \times M_4$	$((M_1 \times M_2) \times M_3) \times M_4$
$(M_2 \times M_3)$	$M_1 \times (M_2 \times M_3)$	$M_2 \times (M_3 \times M_4)$	$(M_1 \times (M_2 \times M_3)) \times M_4$
$(M_3 \times M_4)$			$M_1 \times ((M_2 \times M_3) \times M_4)$
			$M_1 \times (M_2 \times (M_3 \times M_4))$
			$(M_1 \times M_2) \times (M_3 \times M_4)$



Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Dado o problema

$$M := M_1 \times M_2 \times \ldots \times M_n$$

Considere o subproblema (ou subseqüência)

$$ {}_iM_j = \underset{b_{i-1} \times b_i}{M_i} \times \underset{b_i \times b_{i+1}}{M_{i+1}} \times \ldots \times \underset{b_{j-1} \times b_j}{M_j}$$

Com $1 \leq i < j \leq n$ e custo mínimo dado por $\mathbf{{}_im_j}$.

Considere ${}_im_i = 0$, para i=1,…, n

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

O cálculo de ${}_iM_j$ com custo mínimo ${}_im_j$ pode ser decomposto em dois subproblemas. Considere $i \leq k < j$, logo

$$ {}_iM_k = M_i \text{ x } M_{i+1} \text{ x } \ldots \text{ x } M_k \qquad {}_{(k+1)}M_j = M_{k+1} \text{ x } M_{k+2} \text{ x } \ldots \text{ x } M_j $$

Onde

${}_iM_k$ tem custo mínimo ${}_im_k$ e dimensões b_{i-1} x b_k

${}_{(k+1)}M_j$ tem custo mínimo ${}_{(k+1)}m_j$ e dimensões b_k x b_j

O custo associado ao cálculo de $\mathbf{{}_iM_k}$ x $\mathbf{{}_{(k+1)}M_j}$, é dado por

$$ ({}_im_k + {}_{(k+1)}m_j) + (b_{i-1} \text{ x } b_k \text{ x } b_j). $$

O custo mínimo é dado por

$$ {}_im_j = \min_{i \leq k < j} \{({}_im_k + {}_{(k+1)}m_j) + (b_{i-1} \text{ x } b_k \text{ x } b_j)\} $$

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Considere o produto das seguintes matrizes

$$M = \underset{2x30}{M_1} \text{ x } \underset{30x20}{M_2} \text{ x } \underset{20x5}{M_3}$$

Inicialmente temos, $_{\text{¡}}\mathbf{m_i} = 0$, para i=1,2 e 3.

O produto de 2 matrizes pode ser feito das seguintes maneiras

$$\underset{2x30}{M_1} \text{ x } \underset{30x20}{M_2}$$

$_1m_2 = 2 \times 30 \times 20 = 1200$

$$\underset{30x20}{M_2} \text{ x } \underset{20x5}{M_3}$$

$_2m_3 = 30 \times 20 \times 5 = 3000$

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

O produto de 3 matrizes pode ser feito das seguintes maneiras

$M_1 \text{x} (M_2 \text{x} M_3)$ $\qquad\qquad$ $(M_1 \text{x} M_2) \text{x} M_3$

Vimos que o custo mínimo é dado por

$$ {}_i m_j = \min_{i \le k < j} \{ ({}_i m_k + {}_{(k+1)} m_j) + (b_{i-1} \times b_k \times b_j) \} $$

Temos 2 valores possíveis para k, k=1 e k=2.

Para k=1 temos

${}_1m_3 = {}_1m_1 + {}_2m_3 + 2 \text{ x } 30 \text{ x } 5 = 300 + 3000 = 3300$ $\quad M_1 \text{x} (M_2 \text{x} M_3)$

Para k=2 temos

${}_1m_3 = {}_1m_2 + {}_3m_3 + 2 \text{ x } 20 \text{ x } 5 = 1200 + 200 =$ **1400** $\quad (M_1 \text{x} M_2) \text{x} M_3$

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Este processo assemelha-se ao preenchimento de uma matriz

	1	2	3
1	$_1m_1 = 0$		
2		$_2m_2 = 0$	
3			$_3m_3 = 0$

	1	2	3
1	0	$_1m_2 = 1200$	
2		0	$_2m_3 = 3000$
3			0

	1	2	3
1	0	1200	$_1m_3 = 1400$
2		0	3000
3			0

$M_1 \text{x} (M_2 \text{x} M_3) = 3300$

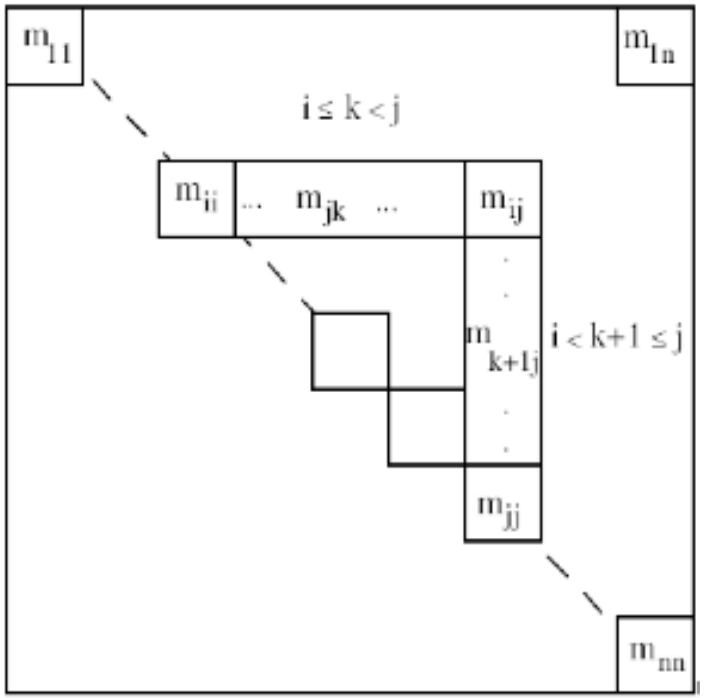
$(M_1 \text{x} M_2) \text{x} M_3 = 1400$

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

O produto de n matrizes

- A diagonal é inicializada, $_{i}m_{i} = 0$
- Os valores para as diagonais superiores são calculados;
- Após o processo, o resultado final encontra-se no canto superior direito da matriz



Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Função: Multi_Mat(b : D) → IN {custo mínimo de produto de matrizes}

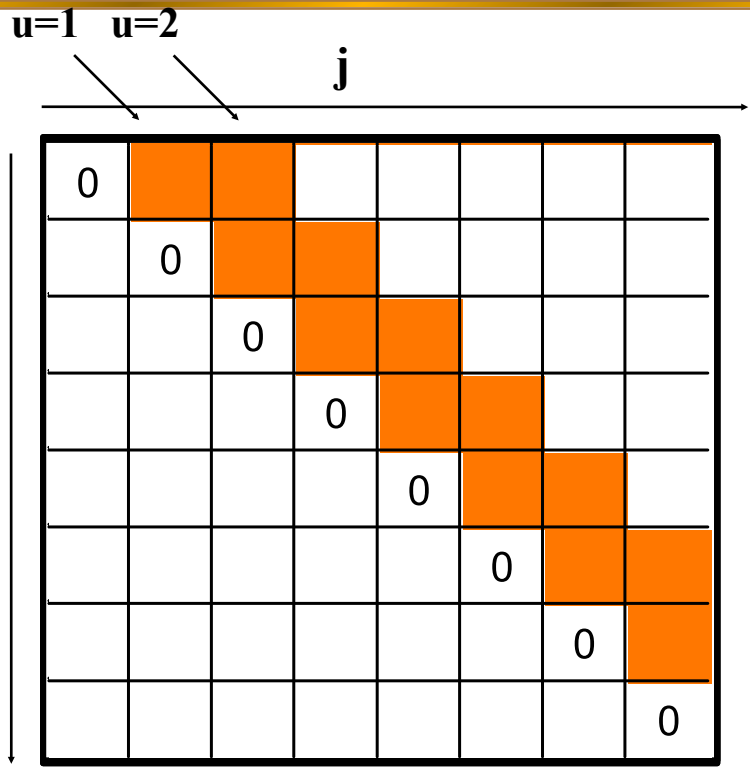
1. para i de 1 até n faça m[i, i] ← 0 ; {inicializa diagonal principal}
2. para u de 1 até n - 1 faça {deslocamento da diagonal: 7}
3. para i de 1 até n - u faça {posição na diagonal: 6}
4. j ← i + u ; {u = j - i} ;
5. $m[i,j] \leftarrow \min_{i \le k < j} \{(m[i,k] + m[k+1,j]) + (b[i-1] \times b[k] \times b[j])\}$;
6. fim-para {3: i de 1 até n - u}
7. fim-para {2: u de 1 até n - 1}
8. retorne-saída(m[1, n]) ; {dá saída extraída}
9. fim-Função {fim do algoritmo Multi_Mat: custo mínimo de produto}

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Função: Multi_Mat(b:D) → IN

1. para i de 1 até n faça m[i,i] ← 0;
2. para u de 1 até n-1 faça
3. para i de 1 até n-u faça
4. j ← i+u; {u=j-i};
5. $m[i,j] \leftarrow \min_{i \le k < j} \{(m[i,k]+m[k+1,j])+(b[i-1]xb[k]\ x\ b[j])\}$;
6. fim-para
7. fim-para
8. retorne-saída(m[1,n]);
9. fim-Função



Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Considere

$$M = \underset{2x30}{M_1} \text{ x } \underset{30x20}{M_2} \text{ x } \underset{20x5}{M_3} \text{ x } \underset{5x10}{M_4}$$

Calcule a sequência ótima de multiplicações e o preenchimento da matriz usada na programação dinâmica. Lembre-se da linha 5.

5. $m[i,j] \leftarrow \min_{i \le k < j} \{(m[i,k] + m[k+1,j]) + (b[i-1] x b[k] x b[j])\}$

	1	2	3	4
1	0	1200	1400	1500
2		0	3000	4500
3			0	1000
4				0

Seqüência ótima $M = ((M_1 \text{ x } M_2) \text{x } M_3) \text{x } M_4$

Complexidade de Algoritmos

Projeto e Análise de Algoritmos

Idéias básicas da programação dinâmica

Objetiva construir uma resposta ótima através da combinação das respostas obtidas para partes menores do problema (subproblemas).

- Inicialmente, a entrada é decomposta em partes mínimas e resolvidas.

- A cada passo, **os resultados parciais são combinados** dando respostas para os subproblemas cada vez maiores, até que se obtenha uma resposta para o problema original.

- A decomposição **é feita uma única vez** e, além disso, os casos menores são tratados antes dos maiores.

- Este método é chamado **ascendente**, ao contrário dos métodos **recursivos**, que são chamados **descendentes**.