

Projeto de algoritmos

Profa Mariana Kolberg

(material adaptado dos slides do prof. Edson)

Projeto de Algoritmos

- Técnicas que possibilitam a elaboração de algoritmos com complexidade polinomial para problemas que normalmente teriam complexidade exponencial através do uso de um pouco de memória extra.

Divisão e Conquista

- Resolve os problemas combinando as soluções para subproblemas
 - Particionam o problema em subproblemas independentes
 - Resolvem os subproblemas **recursivamente**
 - Combinam suas soluções

Programação Dinâmica x Divisão e Conquista

- Assim como a técnica de dividir para conquistar, combina a solução de subproblemas
- Porém, os subproblemas não são independentes
 - Subproblemas compartilham subsubproblemas
- Porque um algoritmo dividir para conquistar não é adequado?
 - Pois ele trabalharia mais que o necessário, resolvendo repetidamente os subsubproblemas comuns

Algoritmos Gulosos

- Se aplicam a problemas de otimização em que diversas escolhas devem ser feitas a fim de se chegar a uma solução ótima
- Idéia: fazer cada escolha de maneira ótima para as condições locais
- Exemplo troco de moedas.

Programação dinâmica x Algoritmos gulosos

- Assim como a técnica algoritmos gulosos:
 - Possui subestrutura ótima
 - É aplicada a problemas de otimização em que uma série de escolhas deve ser feita, a fim de se alcançar uma solução ótima
- Porém, programação dinâmica é útil quando não é fácil chegar a uma seqüência ótima de decisões sem testar todas as seqüências possíveis para então escolher a melhor
- Porque um algoritmo guloso não é adequado?
 - Não é possível encontrar uma função gulosa
- É eficaz quando um dado subproblema pode surgir a partir de mais de um conjunto parcial de escolhas
 - Técnica chave é armazenar a solução para cada um dos subproblemas

Programação Dinâmica

- 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
 - mecanismo que evita aquelas seqüências que sabidamente não podem resultar em seqüências ótimas

Programação Dinâmica

- Sua utilização resulta, em geral, em algoritmos mais eficientes que os algoritmos mais diretos.
 - Em alguns casos, o algoritmo direto tem complexidade exponencial, enquanto que o algoritmo desenvolvido por programação dinâmica é polinomial.
 - Outras vezes, a complexidade continua exponencial, mas de ordem mais baixa.

Programação Dinâmica

- Pode ser aplicada em diversos problemas :
 - multiplicação de várias matrizes;
 - caminhos de custo mínimo em grafos orientados;
 - projeto de sistemas confiáveis;
 - casamento de strings;
 - problema do caixeiro viajante;
 - problema de linha de montagem;
 - entre outros

Programação Dinâmica

- Desenvolvimento em 4 etapas
 1. Caracterizar a estrutura de uma solução ótima (subestrutura ótima)
 2. Definir recursivamente o valor de uma solução ótima
 3. Calcular o valor de uma solução ótima em um processo de baixo para cima (bottom-up)
 4. Construir uma solução ótima a partir das informações calculadas*

* Pode ser omitida se apenas o valor da solução ótima for exigido

* Pode ser necessário guardar outras informações na etapa 3

Programação Dinâmica

Multiplicação de cadeias de matrizes

Consiste em determinar a seqüência ótima de multiplicações de n matrizes

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

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

Este cálculo exige $p \cdot q \cdot r$ multiplicações.

Considere o seguinte exemplo

$$M := \begin{matrix} M_1 & \times & M_2 & \times & M_3 & \times & M_4 & \times & M_5 \\ 100 \times 3 & & 3 \times 10 & & 10 \times 50 & & 50 \times 30 & & 30 \times 5 \end{matrix}$$

Programação Dinâmica

Multiplicação de cadeias de matrizes

$$M \equiv \begin{matrix} M_1 & & M_2 & & M_3 & & M_4 & & M_5 \\ 100 \times 3 & & 3 \times 10 & & 10 \times 50 & & 50 \times 30 & & 30 \times 5 \end{matrix}$$

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

A quantidade de operações é dada por

$$(100 \times 10 \times 3) + (100 \times 10 \times 50) + (100 \times 50 \times 30) + (100 \times 30 \times 5) = 218000 \text{ operações}$$

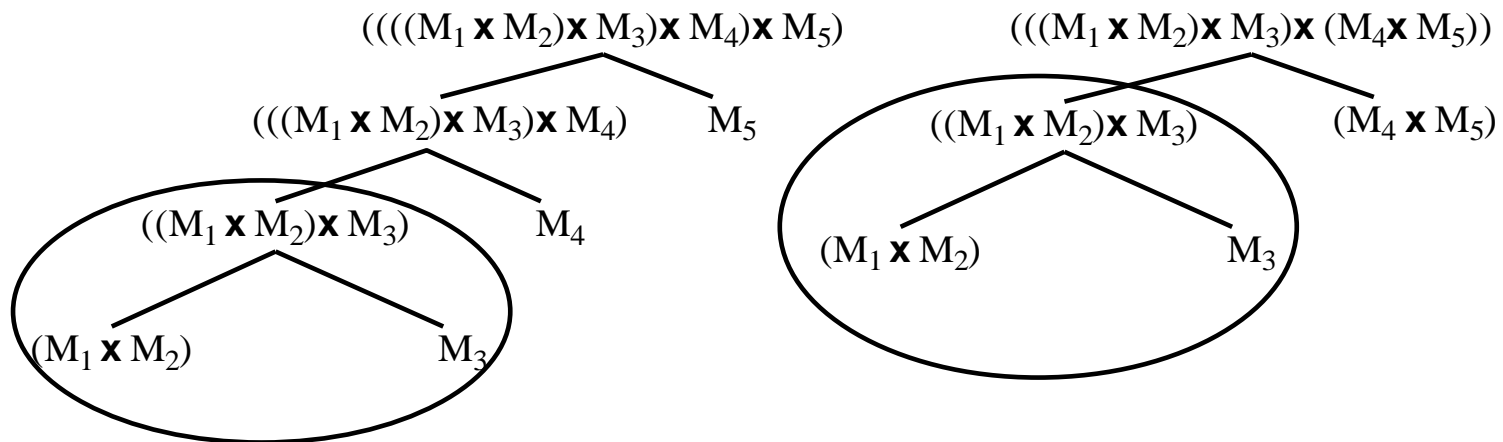
- Segunda maneira $M_1 \times \{M_2 \times [(M_3 \times M_4) \times M_5]\}$

A quantidade de operações é dada por

$$(10 \times 50 \times 30) + (10 \times 30 \times 5) + (3 \times 10 \times 5) + (100 \times 3 \times 5) = 18150 \text{ operações}$$

Programação Dinâmica

Multiplicação de cadeias de matrizes



$(((((M_1 \times M_2) \times M_3) \times M_4) \times M_5) \times M_6) \times M_7)$

$(((((M_1 \times M_2) \times M_3) \times M_4) \times M_5) \times (M_6 \times M_7))$

$((((M_1 \times M_2) \times M_3) \times (M_4 \times M_5)) \times (M_6 \times M_7))$

$((M_1 \times (M_2 \times M_3)) \times ((M_4 \times M_5) \times (M_6 \times M_7)))$

Programação Dinâmica

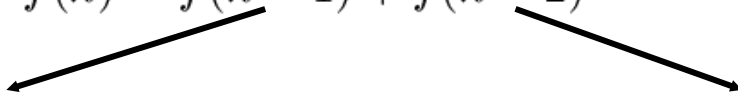
Multiplicação de cadeias de matrizes

- O algoritmo direto tem **complexidade exponencial** no número de matrizes (testaria todas as combinações possíveis)
- Usando a programação dinâmica encontramos um algoritmo de **complexidade polinomial**

Multiplicação de Matrizes

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

Série de Fibonacci

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


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

Programação Dinâmica

Multiplicação de cadeias de matrizes

- Como minimizar ou reduzir a redundância de trabalho ?
 - Devemos resolver os problemas menores e utilizá-los para resolver os maiores

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

Programação Dinâmica

Multiplicação de cadeias de matrizes

- Dado o problema $M := M_1 \times M_2 \times \dots \times M_n$

Considere o subproblema (ou subsequência)

$${}_i M_j = \begin{matrix} M_i & \times & M_{i+1} & \times & \dots & \times & M_j \\ b_{i-1} \times b_i & & b_i \times b_{i+1} & & \dots & & b_{j-1} \times b_j \end{matrix}$$

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

Considere ${}_i m_i = 0$, para $i=1, \dots, n$

- O objetivo é saber a ordem da multiplicação de custo mínimo
 - O tempo de encontrar esta ordem é compensado quando for se fazer as multiplicações.

Programação Dinâmica

Multiplicação de cadeias de matrizes

$${}_2M_3 = M_2 \times M_3$$

$b_1 \times b_2 \quad b_2 \times b_3$

$$M_2 \times M_3$$

$3 \times 5 \quad 5 \times 40$



$${}_2M_3$$

$\textcircled{3} \times \textcircled{40}$
 $b_1 \quad b_3$

$M_1 \times M_2 \times M_3$ $10 \times 3 \quad 3 \times 5 \quad 5 \times 40$				
vetor B				
	10	3	5	40
posição	0	1	2	3

A matriz ${}_2M_3$ é uma matriz 3×40 , ou seja, $b_1 \times b_3$

Portanto, uma matriz ${}_iM_j$ é uma matriz $b_{i-1} \times b_j$

$${}_iM_j = M_i \times M_{i+1} \times \dots \times M_j$$

$b_{i-1} \times b_i \quad b_i \times b_{i+1} \quad \dots \quad b_{j-1} \times b_j$

Programação Dinâmica

Multiplicação de cadeias de matrizes

O cálculo de $_i M_j$ com custo mínimo $_i m_j$ pode ser decomposto em dois subproblemas. Considere $i \leq k < j$, logo

$$_i M_k = M_i \times M_{i+1} \times \dots \times M_k \quad {}_{(k+1)} M_j = M_{k+1} \times M_{k+2} \times \dots \times M_j$$

Onde

$_i M_k$ tem custo mínimo $_i m_k$ e dimensões $b_{i-1} \times b_k$

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

O custo associado ao cálculo de $_i M_k \times {}_{(k+1)} M_j$, é dado por

$$(_i m_k + {}_{(k+1)} m_j) + (b_{i-1} \times b_k \times b_j).$$

O custo mínimo é dado por

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

Programação Dinâmica

Multiplicação de cadeias de matrizes

Considere o produto das seguintes matrizes

$$M = \begin{matrix} M_1 & \times & M_2 & \times & M_3 \\ 2 \times 30 & & 30 \times 20 & & 20 \times 5 \end{matrix}$$

Inicialmente temos, $m_i = 0$, para $i=1,2$ e 3 (solução trivial).

O produto de 2 matrizes pode ser feito das seguintes maneiras

$$\begin{matrix} M_1 & \times & M_2 & & M_2 & \times & M_3 \\ 2 \times 30 & & 30 \times 20 & & 30 \times 20 & & 20 \times 5 \end{matrix}$$

Programação Dinâmica

Multiplicação de cadeias de matrizes

Considere o produto das seguintes matrizes

$$M = \begin{matrix} M_1 & & M_2 & & M_3 \\ 2 \times 30 & \times & 30 \times 20 & \times & 20 \times 5 \end{matrix}$$

Inicialmente temos, $m_i = 0$, para $i=1, 2$ e 3 (solução trivial).

O produto de 2 matrizes pode ser feito das seguintes maneiras

$$\begin{matrix} M_1 & \times & M_2 \\ 2 \times 30 & & 30 \times 20 \end{matrix}$$

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

$$\begin{matrix} M_2 & \times & M_3 \\ 30 \times 20 & & 20 \times 5 \end{matrix}$$

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

Programação Dinâmica

Multiplicação de cadeias de matrizes

O produto de 3 matrizes pode ser feito das seguintes maneiras

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

Vimos que o custo mínimo é dado por

$$m_{ij} = \min_{i \leq k < j} \{ (m_{ik} + m_{kj}) + (b_{i-1} \times b_k \times b_j) \}$$

Qual é o valor de m_{13} ?

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

Programação Dinâmica

Multiplicação de cadeias de matrizes

O produto de 3 matrizes pode ser feito das seguintes maneiras

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

Vimos que o custo mínimo é dado por

$$m_{ij} = \min_{i \leq k < j} \{ (m_{ik} + m_{kj}) + (b_{i-1} \times b_k \times b_j) \}$$

Qual é o valor de m_{13} ?

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

Para k=1 temos

$$m_{13} = m_{11} + m_{23} + 2 \times 30 \times 5 = 300 + 3000 = 3300 \quad M_1 \times (M_2 \times M_3)$$

Para k=2 temos

$$m_{13} = m_{12} + m_{33} + 2 \times 20 \times 5 = 1200 + 200 = \mathbf{1400} \quad (M_1 \times M_2) \times M_3$$

Programação Dinâmica

Multiplicação de cadeias de matrizes

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

Este processo assemelha-se ao preenchimento de uma matriz

	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

Programação Dinâmica

Multiplicação de cadeias de matrizes

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

Este processo assemelha-se ao preenchimento de uma matriz

	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

$$M_1 \times (M_2 \times M_3) = 3300$$

Programação Dinâmica

Multiplicação de cadeias de matrizes

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

Este processo assemelha-se ao preenchimento de uma matriz

	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

$$M_1 \times (M_2 \times M_3) = 3300$$

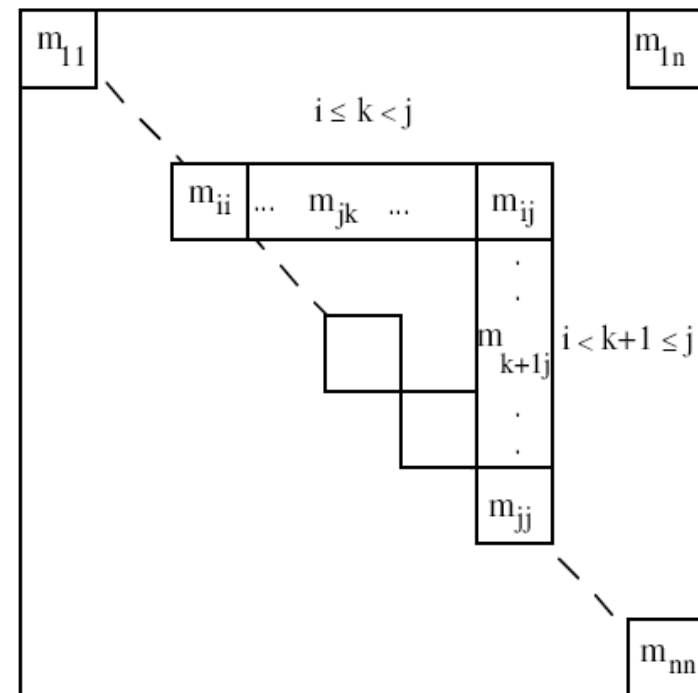
$$(M_1 \times M_2) \times M_3 = 1400$$

Programação Dinâmica

Multiplicação de cadeias de matrizes

O produto de n matrizes

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



Programação Dinâmica

- Caracterizar a estrutura de uma solução ótima
 - Se para encontrar a ordem ótima da multiplicação das matrizes $_iM_j$ dividimos em $_iM_k$ e $_{k+1}M_j$, então tanto $_iM_k$ quanto $_{k+1}M_j$ devem representar uma ordem ótima também
- Definir recursivamente o valor de uma solução ótima
 - $$_iM_j = \begin{cases} 0 & \text{se } i = j \\ \min\{_iM_k + _{k+1}M_j + b_{i-1} \times b_k \times b_j\} & \text{se } i < j \end{cases}$$
- Calcular o valor de uma solução ótima em um processo de baixo para cima (bottom-up)
 - Começa pelo cálculo do custo da multiplicação da menor cadeia de matrizes: tamanho 1,2,3...n
 - A medida que a cadeia de matrizes cresce, são utilizados os custos já calculados anteriormente

Programação Dinâmica

Multiplicação de cadeias de matrizes

- Função: Multi_Mat($b:D$) \rightarrow IN {custo mínimo de produto de matrizes}
1. para i de 1 até n faça $m[i, i] \leftarrow 0$; {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. $j \leftarrow i+u$; $\{u=j-i\}$;
 5. $m[i, j] \leftarrow \min_{i \leq k < j} \{ (m[i, k] + m[k+1, j]) + (b[i-1] \times b[k] \times b[j]) \}$;
 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}

Programação Dinâmica

Multiplicação de cadeias de matrizes

Função: $\text{Multi_Mat}(b:D) \rightarrow \text{IN}$

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

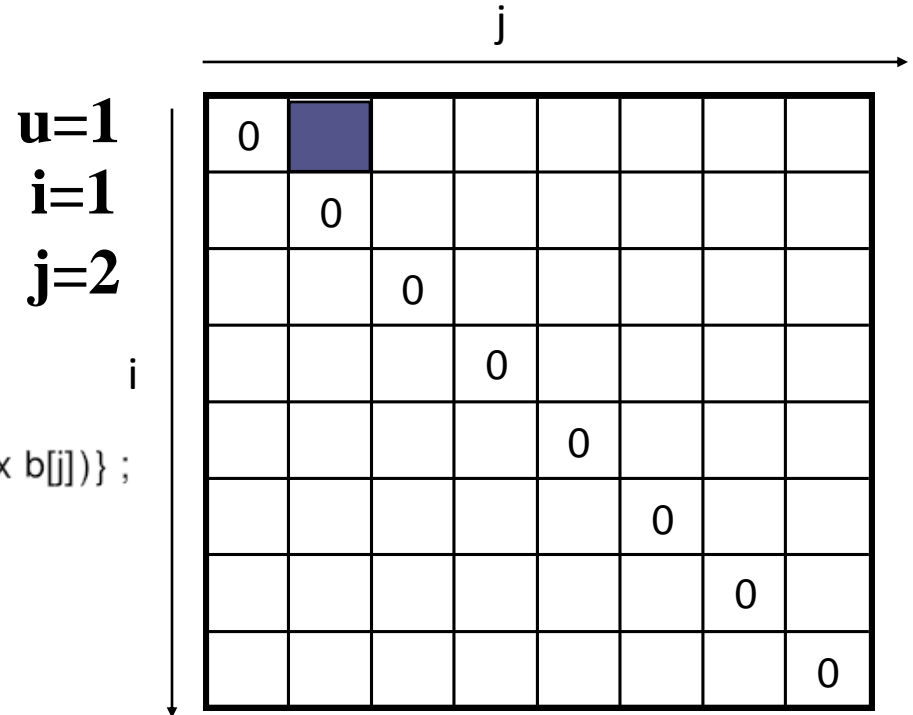
0							
	0						
		0					
			0				
				0			
					0		
						0	
							0

Programação Dinâmica

Multiplicação de cadeias de matrizes

Função: Multi_Mat($b:D$) \rightarrow IN

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

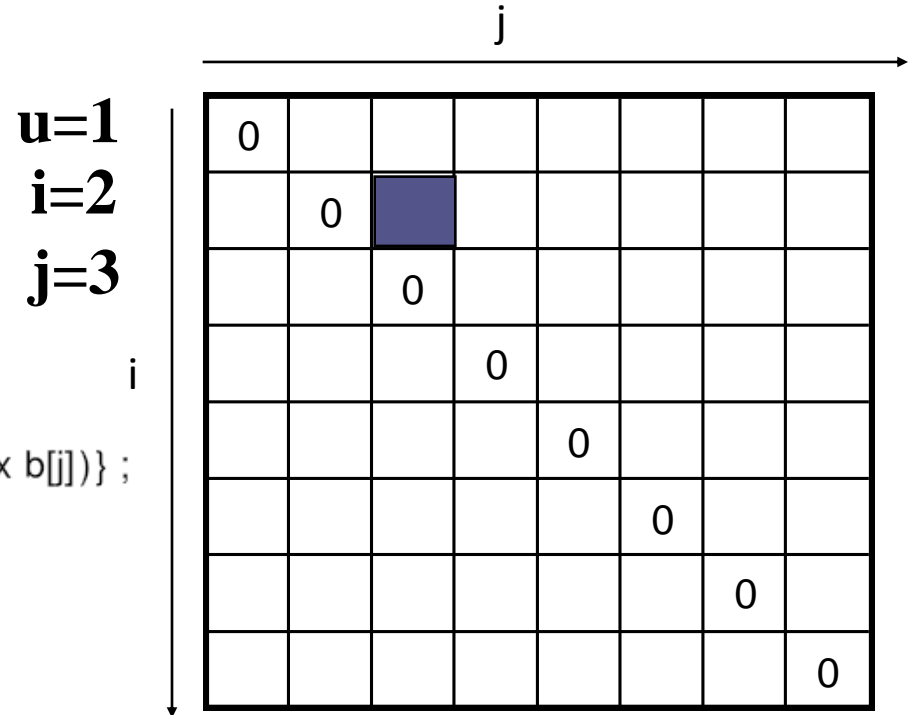


Programação Dinâmica

Multiplicação de cadeias de matrizes

Função: Multi_Mat($b:D$) \rightarrow IN

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

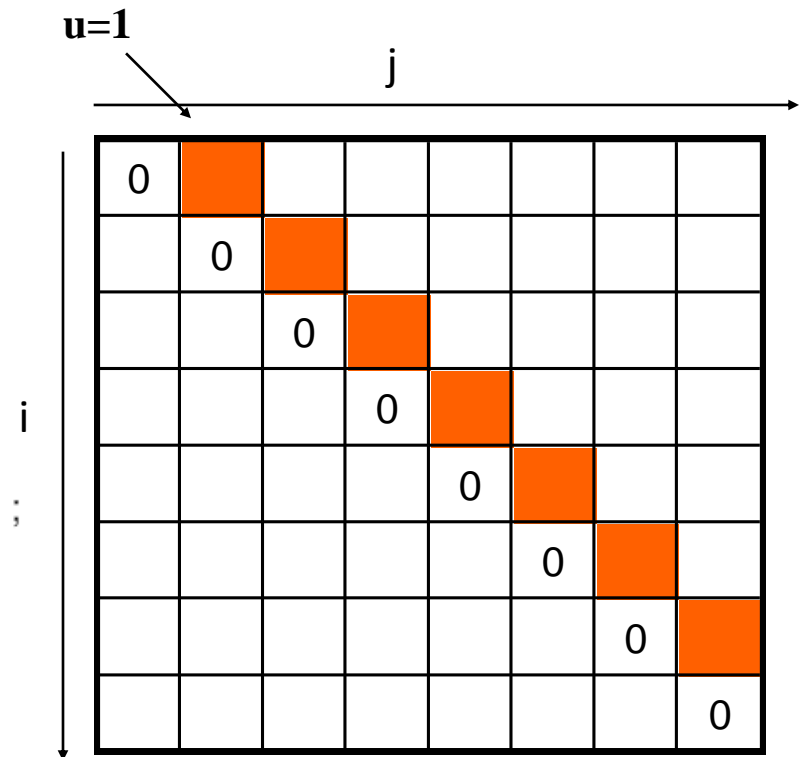


Programação Dinâmica

Multiplicação de cadeias de matrizes

Função: Multi_Mat($b:D$) \rightarrow IN

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

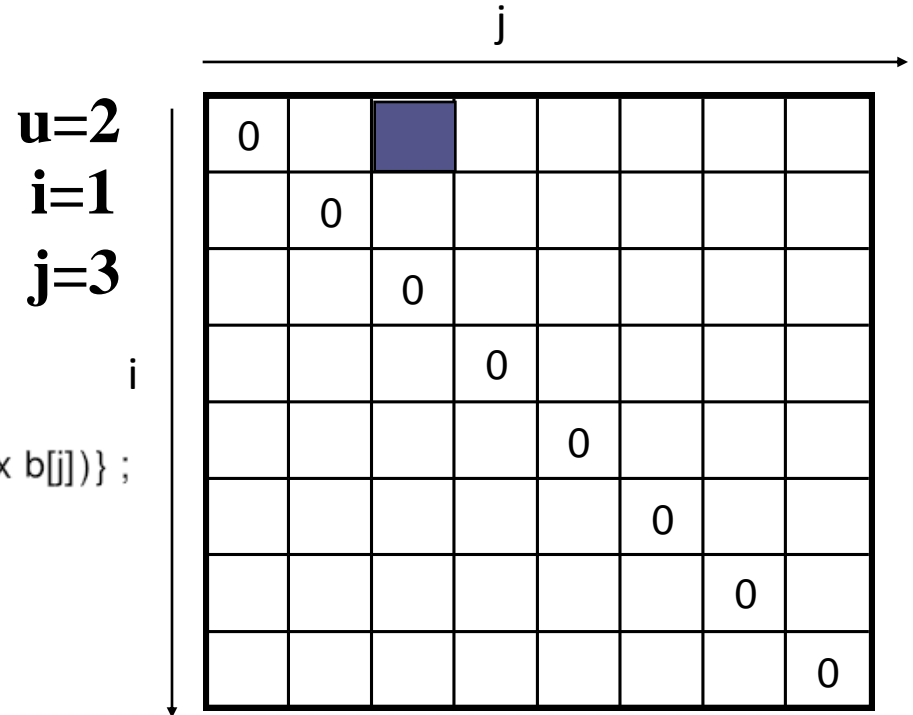


Programação Dinâmica

Multiplicação de cadeias de matrizes

Função: Multi_Mat($b:D$) \rightarrow IN

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

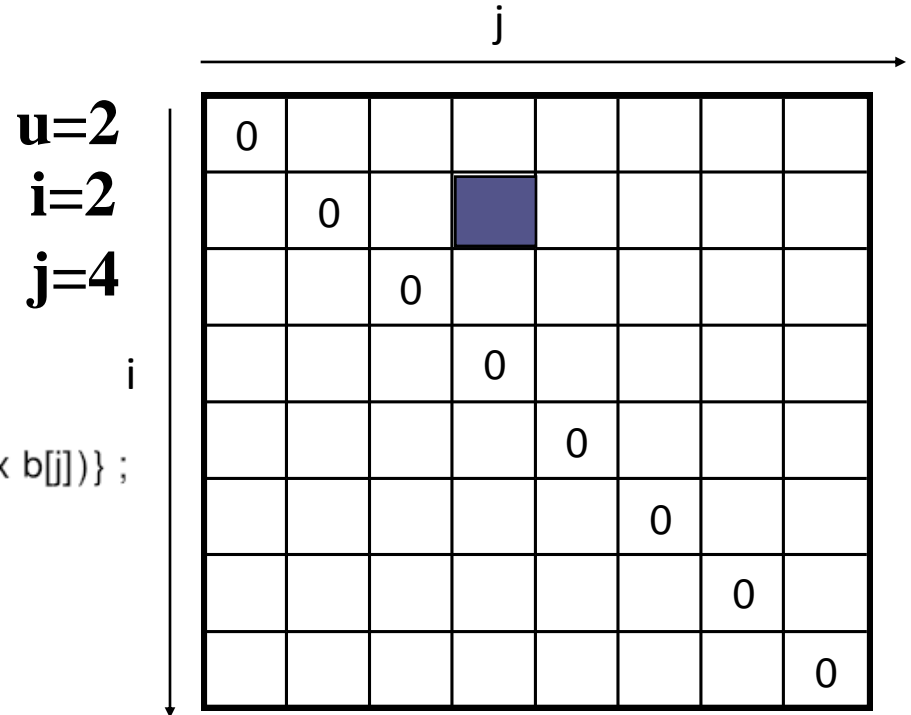


Programação Dinâmica

Multiplicação de cadeias de matrizes

Função: Multi_Mat($b:D$) \rightarrow IN

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





Programação Dinâmica

Multiplicação de cadeias de matrizes

Função: $\text{Multi_Mat}(b:D) \rightarrow \text{IN}$

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

0							
	0						
		0					
			0				
				0			
					0		
						0	
							0

Programação Dinâmica

Multiplicação de cadeias de matrizes

Considere

$$M = \begin{matrix} M_1 & \times & M_2 & \times & M_3 & \times & M_4 \\ 2 \times 30 & & 30 \times 20 & & 20 \times 5 & & 5 \times 10 \end{matrix}$$

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

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

Programação Dinâmica

Multiplicação de cadeias de matrizes

Considere

$$M = \begin{matrix} M_1 & \times & M_2 & \times & M_3 & \times & M_4 \\ 2 \times 30 & & 30 \times 20 & & 20 \times 5 & & 5 \times 10 \end{matrix}$$

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

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

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

Seqüência ótima $M = ((M_1 \times M_2) \times M_3) \times M_4$

Programação Dinâmica

Multiplicação de cadeias de matrizes

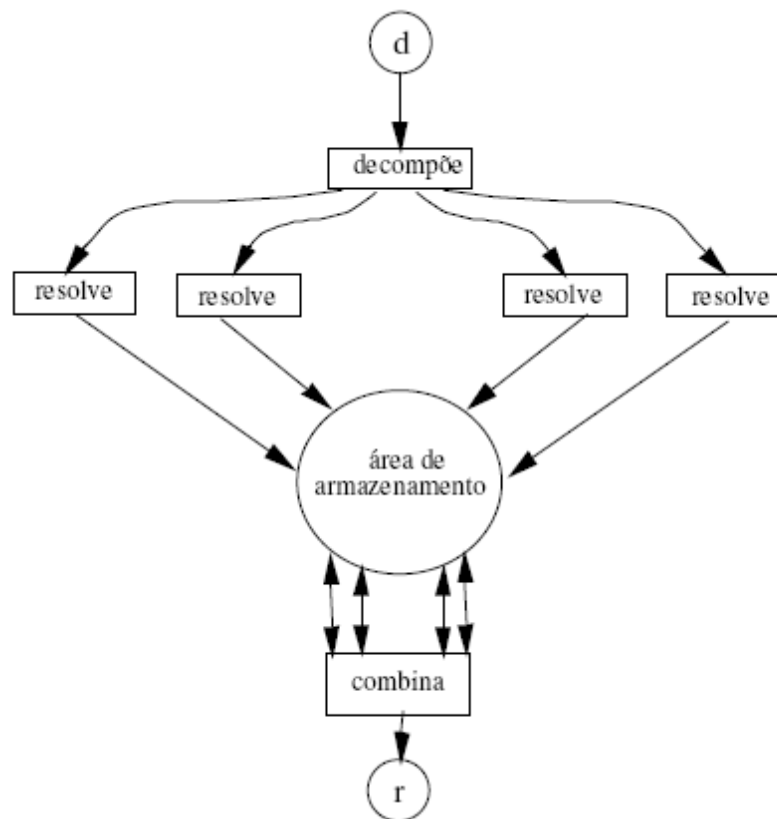
Idéias básicas da programação dinâmica

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**.

Programação Dinâmica

Multiplicação de cadeias de matrizes



Programação Dinâmica

Multiplicação de cadeias de matrizes

Algoritmo: Programação Dinâmica

Função Alg_PD($d:D$) $\rightarrow R$ {Algoritmo de programação dinâmica (abstrato)}

{*Entrada-saída*: saída $r:R$ é resposta ótima para entrada $d:D$ }

Incز_PD ; Iter_PD ; Fnl_PD {estrutura geral}
 onde {componentes}

Incز_PD := {inicialização}
 entrada é decomposta em partes mínimas;
 partes mínimas dão diretamente partes da saída

Iter_PD: itera corpo Crp_PD {iteração}
 escolhe elemento da parte da entrada;
 combina parte da entrada com elemento e ;
 atualiza parte da saída

Fnl_PD: {finalização}
 extraí a saída final para a entrada original

Programação Dinâmica

Multiplicação de cadeias de matrizes

Multi_Mat	Alg_PD
<u>sorte</u> D := vetor b:D de naturais	<u>sorte</u> D {entrada}
1. <u>para</u> i <u>de</u> 1 <u>até</u> n <u>faca</u> m[i, i] ← 0;	IncZ_PD {inicialização}
2. <u>para</u> u <u>de</u> 1 <u>até</u> n-1 <u>faca</u>	Iter_PD {iteração}
3. <u>para</u> i <u>de</u> 1 <u>até</u> n-u <u>faca</u>	Crp_PD {corpo da iteração}
4. j ← i+u;	
5. m[i, j] ← melhor valor;	{atualiza parte de saída}
6. <u>fim-para</u>	
7. <u>fim-para</u>	{fim da iteração}
8. <u>retorne-saída</u> (m[1, n]);	Fnl_G {finalização}

Programação Dinâmica

Multiplicação de cadeias de matrizes

- Função: $\text{Multi_Mat}(b:D) \rightarrow \mathbb{N}$ {custo mínimo de produto de matrizes}
1. para i de 1 até n faça $m[i, i] \leftarrow 0$; {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. $j \leftarrow i + u$; $\{u = j - i\}$;
 5. $m[i, j] \leftarrow \min_{i \leq k < j} \{ (m[i, k] + m[k + 1, j]) + (b[i - 1] \times b[k] \times b[j]) \}$;
 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}