

Programação dinâmica

CLRS 15.2–15.3

- = “recursão-com-tabela”
- = transformação inteligente de recursão em iteração

Multiplicação iterada de matrizes

Se A é $p \times q$ e B é $q \times r$ então AB é $p \times r$.

$$(AB)[i,j] = \sum_k A[i,k] B[k,j]$$

MULT-MAT (p, A, q, B, r)

```
1  para  $i \leftarrow 1$  até  $p$  faça
2      para  $j \leftarrow 1$  até  $r$  faça
3           $AB[i,j] \leftarrow 0$ 
4          para  $k \leftarrow 1$  até  $q$  faça
5               $AB[i,j] \leftarrow AB[i,j] + A[i,k] \cdot B[k,j]$ 
```

Número de multiplicações escalares = $p \cdot q \cdot r$

Multiplicação iterada

Problema: Encontrar **número mínimo** de multiplicações escalares necessário para calcular produto $A_1 A_2 \cdots A_n$.

$$\begin{array}{ccccccc} p[0] & & p[1] & & p[2] & \dots & p[n-1] & & p[n] \\ & A_1 & & A_2 & & \dots & & A_n & \end{array}$$

cada A_i é $p[i-1] \times p[i]$ ($A_i[1..p[i-1], 1..p[i]]$)

Exemplo: $A_1 \cdot A_2 \cdot A_3$

	10	A_1	100	A_2	5	A_3	50	
$((A_1 A_2) A_3)$								7500
$(A_1 (A_2 A_3))$								75000
								multiplicações escalares
								multiplicações escalares

Soluções ótimas contêm soluções ótimas

Se

$$(A_1 A_2)(A_3((A_4 A_5) A_6))$$

é **ordem ótima** de multiplicação então

$$(A_1 A_2) \quad \text{e} \quad (A_3((A_4 A_5) A_6))$$

também são **ordens ótimas**.

Soluções ótimas contêm soluções ótimas

Se

$$(A_1 A_2)(A_3((A_4 A_5) A_6))$$

é **ordem ótima** de multiplicação então

$$(A_1 A_2) \quad \text{e} \quad (A_3((A_4 A_5) A_6))$$

também são **ordens ótimas**.

Decomposição: $(A_i \cdots A_k)(A_{k+1} \cdots A_j)$

$m[i, j] =$ **número mínimo** de multiplicações escalares
para calcular $A_i \cdots A_j$

Recorrência

$m[i, j] =$ número mínimo de multiplicações escalares
para calcular $A_i \cdots A_j$

Recorrência

$m[i, j]$ = número mínimo de multiplicações escalares
para calcular $A_i \cdots A_j$

se $i = j$ então $m[i, j] = 0$

se $i < j$ então

$$m[i, j] = \min_{i \leq k < j} \{ m[i, k] + p[i-1]p[k]p[j] + m[k+1, j] \}$$

Exemplo:

$$m[3, 7] = \min_{3 \leq k < 7} \{ m[3, k] + p[2]p[k]p[7] + m[k+1, 7] \}$$

Algoritmo recursivo

Recebe $p[i-1..j]$ e devolve $m[i,j]$

REC-MAT-CHAIN (p, i, j)

```
1  se  $i = j$ 
2    então devolva 0
3   $m[i, j] \leftarrow \infty$ 
4  para  $k \leftarrow i$  até  $j - 1$  faça
5     $q_1 \leftarrow \text{REC-MAT-CHAIN}(p, i, k)$ 
6     $q_2 \leftarrow \text{REC-MAT-CHAIN}(p, k + 1, j)$ 
7     $q \leftarrow q_1 + p[i - 1]p[k]p[j] + q_2$ 
8    se  $q < m[i, j]$ 
9      então  $m[i, j] \leftarrow q$ 
10 devolva  $m[i, j]$ 
```

Consumo de tempo?

Consumo de tempo

$T(n)$ = número comparações entre q e $m[\star, \star]$
na linha 8 quando $n := j - i + 1$

$$T(1) = 0$$

$$\begin{aligned} T(n) &= \sum_{h=1}^{n-1} (T(h) + T(n-h) + 1) = 2 \sum_{h=2}^{n-1} T(h) + (n-1) \\ &= 2(T(2) + \dots + T(n-1)) + (n-1) \text{ para } n \geq 2 \end{aligned}$$

Consumo de tempo

$T(n)$ = número comparações entre q e $m[\star, \star]$
na linha 8 quando $n := j - i + 1$

$$T(n) = 2(T(2) + \dots + T(n-1)) + (n-1)$$

Considere a mesma fórmula para $n-1$:

$$T(n-1) = 2(T(2) + \dots + T(n-2)) + (n-2)$$

e subtraia a primeira da segunda:

Consumo de tempo

$T(n)$ = número comparações entre q e $m[\star, \star]$
na linha 8 quando $n := j - i + 1$

$$T(n) = 2(T(2) + \dots + T(n-1)) + (n-1)$$

Considere a mesma fórmula para $n-1$:

$$T(n-1) = 2(T(2) + \dots + T(n-2)) + (n-2)$$

e subtraia a primeira da segunda:

$$T(n) - T(n-1) = 2T(n-1) + 1.$$

Logo $T(n) = 3T(n-1) + 1$.

Consumo de tempo

$T(n)$ = número comparações entre q e $m[\star, \star]$
na linha 8 quando $n := j - i + 1$

$$T(n) = 2(T(2) + \dots + T(n-1)) + (n-1)$$

Considere a mesma fórmula para $n-1$:

$$T(n-1) = 2(T(2) + \dots + T(n-2)) + (n-2)$$

e subtraia a primeira da segunda:

$$T(n) - T(n-1) = 2T(n-1) + 1.$$

Logo $T(n) = 3T(n-1) + 1$.

Fácil verificar que $T(n) \geq \frac{3^{n-1}-1}{2}$ para $n \geq 1$.

Recorrência

n	1	2	3	4	5	6	7	8
$T(n)$	0	1	4	13	40	121	364	1093
$3^{n-1} - 1$	0	2	8	26	80	242	728	2186

Prova: Para $n = 1$, $T(1) = 0 = (1 - 1)/2$.

Recorrência

n	1	2	3	4	5	6	7	8
$T(n)$	0	1	4	13	40	121	364	1093
$3^{n-1} - 1$	0	2	8	26	80	242	728	2186

Prova: Para $n = 1$, $T(1) = 0 = (1 - 1)/2$.

Para $n \geq 2$,

$$\begin{aligned}T(n) &= 3T(n-1) + 1 \\&\stackrel{\text{hi}}{=} 3\left(\frac{3^{n-1} - 1}{2}\right) + 1 \\&= \frac{3^n - 3}{2} + 1 = \frac{3^n - 3 + 2}{2} \\&= \frac{3^n - 1}{2}.\end{aligned}$$

Conclusão

$$T(n) \geq \frac{3^{n-1}-1}{2} \text{ para } n \geq 1.$$

O consumo de tempo do algoritmo REC-MAT-CHAIN é $\Omega(3^n)$.

Resolve subproblemas muitas vezes

$$p[0] = 10 \quad p[1] = 100 \quad p[2] = 5 \quad p[3] = 50$$

```
REC-MAT-CHAIN(p, 1, 3)
  REC-MAT-CHAIN(p, 1, 1)
  REC-MAT-CHAIN(p, 2, 3)
    REC-MAT-CHAIN(p, 2, 2)
    REC-MAT-CHAIN(p, 3, 3)
  REC-MAT-CHAIN(p, 1, 2)
    REC-MAT-CHAIN(p, 1, 1)
    REC-MAT-CHAIN(p, 2, 2)
  REC-MAT-CHAIN(p, 3, 3)
```

Número mínimo de mults = 7500

Resolve subproblemas muitas vezes

REC-MAT-CHAIN(p, 1, 5)	REC-MAT-CHAIN(p, 4, 4)	REC-MAT-CHAIN(p, 1, 1)
REC-MAT-CHAIN(p, 1, 1)	REC-MAT-CHAIN(p, 5, 5)	REC-MAT-CHAIN(p, 2, 4)
REC-MAT-CHAIN(p, 2, 5)	REC-MAT-CHAIN(p, 1, 2)	REC-MAT-CHAIN(p, 2, 2)
REC-MAT-CHAIN(p, 2, 2)	REC-MAT-CHAIN(p, 1, 1)	REC-MAT-CHAIN(p, 3, 4)
REC-MAT-CHAIN(p, 3, 5)	REC-MAT-CHAIN(p, 2, 2)	REC-MAT-CHAIN(p, 3, 3)
REC-MAT-CHAIN(p, 3, 3)	REC-MAT-CHAIN(p, 3, 5)	REC-MAT-CHAIN(p, 4, 4)
REC-MAT-CHAIN(p, 4, 5)	REC-MAT-CHAIN(p, 3, 3)	REC-MAT-CHAIN(p, 2, 3)
REC-MAT-CHAIN(p, 4, 4)	REC-MAT-CHAIN(p, 4, 5)	REC-MAT-CHAIN(p, 2, 2)
REC-MAT-CHAIN(p, 5, 5)	REC-MAT-CHAIN(p, 4, 4)	REC-MAT-CHAIN(p, 3, 3)
REC-MAT-CHAIN(p, 3, 4)	REC-MAT-CHAIN(p, 5, 5)	REC-MAT-CHAIN(p, 4, 4)
REC-MAT-CHAIN(p, 3, 3)	REC-MAT-CHAIN(p, 3, 4)	REC-MAT-CHAIN(p, 1, 2)
REC-MAT-CHAIN(p, 4, 4)	REC-MAT-CHAIN(p, 3, 3)	REC-MAT-CHAIN(p, 1, 1)
REC-MAT-CHAIN(p, 5, 5)	REC-MAT-CHAIN(p, 4, 4)	REC-MAT-CHAIN(p, 2, 2)
REC-MAT-CHAIN(p, 2, 3)	REC-MAT-CHAIN(p, 5, 5)	REC-MAT-CHAIN(p, 3, 4)
REC-MAT-CHAIN(p, 2, 2)	REC-MAT-CHAIN(p, 1, 3)	REC-MAT-CHAIN(p, 3, 3)
REC-MAT-CHAIN(p, 3, 3)	REC-MAT-CHAIN(p, 1, 1)	REC-MAT-CHAIN(p, 4, 4)
REC-MAT-CHAIN(p, 4, 5)	REC-MAT-CHAIN(p, 2, 3)	REC-MAT-CHAIN(p, 1, 3)
REC-MAT-CHAIN(p, 4, 4)	REC-MAT-CHAIN(p, 2, 2)	REC-MAT-CHAIN(p, 1, 1)
REC-MAT-CHAIN(p, 5, 5)	REC-MAT-CHAIN(p, 3, 3)	REC-MAT-CHAIN(p, 2, 3)
REC-MAT-CHAIN(p, 2, 4)	REC-MAT-CHAIN(p, 1, 2)	REC-MAT-CHAIN(p, 2, 2)
REC-MAT-CHAIN(p, 2, 2)	REC-MAT-CHAIN(p, 1, 1)	REC-MAT-CHAIN(p, 3, 3)
REC-MAT-CHAIN(p, 3, 4)	REC-MAT-CHAIN(p, 2, 2)	REC-MAT-CHAIN(p, 1, 2)
REC-MAT-CHAIN(p, 3, 3)	REC-MAT-CHAIN(p, 3, 3)	REC-MAT-CHAIN(p, 1, 1)
REC-MAT-CHAIN(p, 4, 4)	REC-MAT-CHAIN(p, 4, 5)	REC-MAT-CHAIN(p, 2, 2)
REC-MAT-CHAIN(p, 2, 3)	REC-MAT-CHAIN(p, 4, 4)	REC-MAT-CHAIN(p, 3, 3)
REC-MAT-CHAIN(p, 2, 2)	REC-MAT-CHAIN(p, 5, 5)	REC-MAT-CHAIN(p, 4, 4)
REC-MAT-CHAIN(p, 3, 3)	REC-MAT-CHAIN(p, 1, 4)	REC-MAT-CHAIN(p, 5, 5)

Programação dinâmica

Cada subproblema

$$A_i \cdots A_j$$

é resolvido **uma só** vez.

Em que ordem calcular os componentes da tabela m ?

Para calcular $m[2,6]$ preciso de ...

Programação dinâmica

Cada subproblema

$$A_i \cdots A_j$$

é resolvido **uma só** vez.

Em que ordem calcular os componentes da tabela m ?

Para calcular $m[2,6]$ preciso de ...

$m[2,2]$, $m[2,3]$, $m[2,4]$, $m[2,5]$ e de
 $m[3,6]$, $m[4,6]$, $m[5,6]$, $m[6,6]$.

Programação dinâmica

Cada subproblema

$$A_i \cdots A_j$$

é resolvido **uma só** vez.

Em que ordem calcular os componentes da tabela m ?

Para calcular $m[2,6]$ preciso de ...

$m[2,2]$, $m[2,3]$, $m[2,4]$, $m[2,5]$ e de
 $m[3,6]$, $m[4,6]$, $m[5,6]$, $m[6,6]$.

Calcule todos os $m[i,j]$ com $j - i + 1 = 2$,
depois todos com $j - i + 1 = 3$,
depois todos com $j - i + 1 = 4$,
etc.

Programação dinâmica

	1	2	3	4	5	6	7	8	<i>j</i>
1	0								
2		0	★	★	★	??			
3			0			★			
4				0		★			
5					0	★			
6						0			
7							0		
8								0	

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	??					
2		0					
3			0				
4				0			
5					0		
6						0	
i							

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000					
2		0					
3			0				
4				0			
5					0		
6						0	

$$m[1,1] + p[1-1]p[1]p[2] + m[1+1,2] = 0 + 2000 + 0 = 2000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000					
2		0	??				
3			0				
4				0			
5					0		
6						0	
i							

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000					
2		0	6000				
3			0				
4				0			
5					0		
6						0	

$$m[2,2] + p[2-1]p[2]p[3] + m[2+1,3] = 0 + 6000 + 0 = 6000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000					
2		0	6000				
3			0	??			
4				0			
5					0		
6						0	
i							

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000					
2		0	6000				
3			0	6000			
4				0			
5					0		
6						0	

$$m[3,3] + p[3-1]p[3]p[4] + m[3+1,4] = 0 + 6000 + 0 = 6000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000					
2		0	6000				
3			0	6000			
4				0	??		
5					0		
6						0	
i							

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000					
2		0	6000				
3			0	6000			
4				0	4500		
5					0		
6						0	

$$m[4,4] + p[4-1]p[4]p[5] + m[4+1,5] = 0 + 4500 + 0 = 4500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 <i>j</i>
1	0	2000				
2		0	6000			
3			0	6000		
4				0	4500	
5					0	??
6 <i>i</i>						0

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000					
2		0	6000				
3			0	6000			
4				0	4500		
5					0	4500	
6						0	

$$m[5,5] + p[5-1]p[5]p[6] + m[5+1,6] = 0 + 4500 + 0 = 4500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	??				
2		0	6000				
3			0	6000			
4				0	4500		
5					0	4500	
6						0	
i							

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	9000				
2		0	6000				
3			0	6000			
4				0	4500		
5					0	4500	
6						0	

$$m[1,1] + p[1-1]p[1]p[3] + m[1+1,3] = 0 + 3000 + 6000 = 9000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000				
2		0	6000				
3			0	6000			
4				0	4500		
5					0	4500	
6						0	

$$m[1,2] + p[1-1]p[2]p[3] + m[2+1,3] = 2000 + 6000 + 0 = 8000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000				
2		0	6000	??			
3			0	6000			
4				0	4500		
5					0	4500	
6						0	
i							

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000				
2		0	6000	8000			
3			0	6000			
4				0	4500		
5					0	4500	
6						0	

$$m[2,2] + p[2-1]p[2]p[4] + m[2+1,4] = 0 + 2000 + 6000 = 8000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000				
2		0	6000	8000			
3			0	6000			
4				0	4500		
5					0	4500	
6						0	

$$m[2,3] + p[2-1]p[3]p[4] + m[3+1,4] = 6000 + 3000 + 0 = 9000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000				
2		0	6000	8000			
3			0	6000	??		
4				0	4500		
5					0	4500	
6						0	

i

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000				
2		0	6000	8000			
3			0	6000	13500		
4				0	4500		
5					0	4500	
6						0	

$$m[3,3] + p[3-1]p[3]p[5] + m[3+1,5] = 0 + 9000 + 4500 = 13500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000				
2		0	6000	8000			
3			0	6000	9000		
4				0	4500		
5					0	4500	
6						0	

$$m[3,4] + p[3-1]p[4]p[5] + m[4+1,5] = 6000 + 3000 + 0 = 9000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000				
2		0	6000	8000			
3			0	6000	9000		
4				0	4500	??	
5					0	4500	
6						0	
i							

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000			
2		0	6000	8000		
3			0	6000	9000	
4				0	4500	13500
5					0	4500
6						0

$$m[4,4] + p[4-1]p[4]p[6] + m[4+1,6] = 0 + 9000 + 4500 = 13500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000			
2		0	6000	8000		
3			0	6000	9000	
4				0	4500	13500
5					0	4500
6						0

$$m[4,5] + p[4-1]p[5]p[6] + m[5+1,6] = 4500 + 13500 + 0 = 18000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	??			
2		0	6000	8000			
3			0	6000	9000		
4				0	4500	13500	
5					0	4500	
6						0	
i							

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000			
2		0	6000	8000			
3			0	6000	9000		
4				0	4500	13500	
5					0	4500	
6						0	

$$m[1,1] + p[1-1]p[1]p[4] + m[1+1,4] = 0 + 1000 + 8000 = 9000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000			
2		0	6000	8000			
3			0	6000	9000		
4				0	4500	13500	
5					0	4500	
6						0	

$$m[1,2] + p[1-1]p[2]p[4] + m[2+1,4] = 2000 + 2000 + 6000 = 10000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000			
2		0	6000	8000			
3			0	6000	9000		
4				0	4500	13500	
5					0	4500	
6						0	

$$m[1,3] + p[1-1]p[3]p[4] + m[3+1,4] = 8000 + 3000 + 0 = 11000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000			
2		0	6000	8000	??		
3			0	6000	9000		
4				0	4500	13500	
5					0	4500	
6						0	

i

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000			
2		0	6000	8000	12000		
3			0	6000	9000		
4				0	4500	13500	
5					0	4500	
6						0	

$$m[2,2] + p[2-1]p[2]p[5] + m[2+1,5] = 0 + 3000 + 9000 = 12000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000			
2		0	6000	8000	12000		
3			0	6000	9000		
4				0	4500	13500	
5					0	4500	
6						0	

$$m[2,3] + p[2-1]p[3]p[5] + m[3+1,5] = 6000 + 4500 + 4500 = 15000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000			
2		0	6000	8000	9500		
3			0	6000	9000		
4				0	4500	13500	
5					0	4500	
6						0	

$$m[2,4] + p[2-1]p[4]p[5] + m[4+1,5] = 8000 + 1500 + 0 = 9500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000			
2		0	6000	8000	9500		
3			0	6000	9000	??	
4				0	4500	13500	
5					0	4500	
6						0	

i

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000		
2		0	6000	8000	9500	
3			0	6000	9000	31500
4				0	4500	13500
5					0	4500
6						0

$$m[3,3] + p[3-1]p[3]p[6] + m[3+1,6] = 0 + 18000 + 13500 = 31500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000		
2		0	6000	8000	9500	
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6						0

$$m[3,4] + p[3-1]p[4]p[6] + m[4+1,6] = 6000 + 6000 + 4500 = 16500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000		
2		0	6000	8000	9500	
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6						0

$$m[3,5] + p[3-1]p[5]p[6] + m[5+1,6] = 9000 + 9000 + 0 = 18000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000	??		
2		0	6000	8000	9500		
3			0	6000	9000	16500	
4				0	4500	13500	
5					0	4500	
6						0	
i							

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000	11000		
2		0	6000	8000	9500		
3			0	6000	9000	16500	
4				0	4500	13500	
5					0	4500	
6						0	

$$m[1,1] + p[1-1]p[1]p[5] + m[1+1,5] = 0 + 1500 + 9500 = 11000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000	11000		
2		0	6000	8000	9500		
3			0	6000	9000	16500	
4				0	4500	13500	
5					0	4500	
6						0	

$$m[1,2] + p[1-1]p[2]p[5] + m[2+1,5] = 2000 + 3000 + 9000 = 14000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000	11000		
2		0	6000	8000	9500		
3			0	6000	9000	16500	
4				0	4500	13500	
5					0	4500	
6						0	

$$m[1,3] + p[1-1]p[3]p[5] + m[3+1,5] = 8000 + 4500 + 4500 = 17000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000	10500		
2		0	6000	8000	9500		
3			0	6000	9000	16500	
4				0	4500	13500	
5					0	4500	
6						0	

$$m[1,4] + p[1-1]p[4]p[5] + m[4+1,5] = 9000 + 1500 + 0 = 10500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 <i>j</i>
1	0	2000	8000	9000	10500	
2		0	6000	8000	9500	??
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6 <i>i</i>						0

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000	10500	
2		0	6000	8000	9500	22500
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6						0

$$m[2,2] + p[2-1]p[2]p[6] + m[2+1,6] = 0 + 6000 + 16500 = 22500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000	10500	
2		0	6000	8000	9500	22500
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6						0

$$m[2,3] + p[2-1]p[3]p[6] + m[3+1,6] = 6000 + 9000 + 13500 = 28500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000	10500	
2		0	6000	8000	9500	15500
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6						0

$$m[2,4] + p[2-1]p[4]p[6] + m[4+1,6] = 8000 + 3000 + 4500 = 15500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000	10500	
2		0	6000	8000	9500	14000
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6						0

$$m[2,5] + p[2-1]p[5]p[6] + m[5+1,6] = 9500 + 4500 + 0 = 14000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	j
1	0	2000	8000	9000	10500	??	
2		0	6000	8000	9500	14000	
3			0	6000	9000	16500	
4				0	4500	13500	
5					0	4500	
6						0	
i							

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000	10500	17000
2		0	6000	8000	9500	14000
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6						0

$$m[1,1] + p[1-1]p[1]p[6] + m[1+1,6] = 0 + 3000 + 14000 = 17000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000	10500	17000
2		0	6000	8000	9500	14000
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6						0

$$m[1,2] + p[1-1]p[2]p[6] + m[2+1,6] = 2000 + 6000 + 16500 = 24500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000	10500	17000
2		0	6000	8000	9500	14000
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6						0

$$m[1,3] + p[1-1]p[3]p[6] + m[3+1,6] = 8000 + 9000 + 13500 = 30500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000	10500	16500
2		0	6000	8000	9500	14000
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6						0

$$m[1,4] + p[1-1]p[4]p[6] + m[4+1,6] = 9000 + 3000 + 4500 = 16500$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6 j
1	0	2000	8000	9000	10500	15000
2		0	6000	8000	9500	14000
3			0	6000	9000	16500
4				0	4500	13500
5					0	4500
6						0

$$m[1,5] + p[1-1]p[5]p[6] + m[5+1,6] = 10500 + 4500 + 0 = 15000$$

Simulação

$p[0]=10$ $p[1]=10$ $p[2]=20$ $p[3]=30$ $p[4]=10$ $p[5]=15$ $p[6]=30$

	1	2	3	4	5	6	<i>j</i>
1	0	2000	8000	9000	10500	15000	
2		0	6000	8000	9500	14000	
3			0	6000	9000	16500	
4				0	4500	13500	
5					0	4500	
6						0	

i

Algoritmo de programação dinâmica

Recebe $p[0..n]$ e devolve $m[1, n]$.

MATRIX-CHAIN-ORDER (p, n)

```
1  para  $i \leftarrow 1$  até  $n$  faça
2       $m[i, i] \leftarrow 0$ 
3  para  $\ell \leftarrow 2$  até  $n$  faça
4      para  $i \leftarrow 1$  até  $n - \ell + 1$  faça
5           $j \leftarrow i + \ell - 1$ 
6           $m[i, j] \leftarrow \infty$ 
7          para  $k \leftarrow i$  até  $j - 1$  faça
8               $q \leftarrow m[i, k] + p[i - 1]p[k]p[j] + m[k + 1, j]$ 
9              se  $q < m[i, j]$ 
10                 então  $m[i, j] \leftarrow q$ 
11 devolva  $m[1, n]$ 
```

Correção e consumo de tempo

Linhas 3–10: tratam subcadeias $A_i \cdots A_j$ de comprimento ℓ

Correção e consumo de tempo

Linhas 3–10: tratam subcadeias $A_i \cdots A_j$ de comprimento ℓ

Consumo de tempo: ???

Correção e consumo de tempo

Linhas 3–10: tratam subcadeias $A_i \cdots A_j$ de comprimento ℓ

Consumo de tempo: $O(n^3)$ (três loops encaixados)

Correção e consumo de tempo

Linhas 3–10: tratam subcadeias $A_i \cdots A_j$ de comprimento ℓ

Consumo de tempo: $O(n^3)$ (três loops encaixados)

Curioso verificar que consumo de tempo é $\Omega(n^3)$:

Número de execuções da linha 8:

Correção e consumo de tempo

Linhas 3–10: tratam subcadeias $A_i \cdots A_j$ de comprimento ℓ

Consumo de tempo: $O(n^3)$ (três loops encaixados)

Curioso verificar que consumo de tempo é $\Omega(n^3)$:

Número de execuções da linha 8:

ℓ	i	execs linha 8
2	$1, \dots, n-1$	$(n-1) \cdot 1$
3	$1, \dots, n-2$	$(n-2) \cdot 2$
4	$1, \dots, n-3$	$(n-3) \cdot 3$
...	...	...
$n-1$	$1, 2$	$2 \cdot (n-2)$
n	1	$1 \cdot (n-1)$
total		$\sum_{h=1}^{n-1} h(n-h)$

Consumo de tempo

$$\begin{aligned}\text{Para } n \geq 6, \quad \sum_{h=1}^{n-1} h(n-h) &= \\&= n \sum_{h=1}^{n-1} h - \sum_{h=1}^{n-1} h^2 \\&= n \frac{1}{2} n(n-1) - \frac{1}{6} (n-1)n(2n-1) \quad (\text{CLRS p.1060}) \\&\geq \frac{1}{2} n^2 (n-1) - \frac{1}{6} 2n^3 \\&\geq \frac{1}{2} n^2 \frac{5n}{6} - \frac{1}{3} n^3 \\&= \frac{5}{12} n^3 - \frac{1}{3} n^3 \\&= \frac{1}{12} n^3.\end{aligned}$$

Consumo de tempo é $\Omega(n^3)$

Conclusão

O consumo de tempo do
algoritmo **MATRIX-CHAIN-ORDER** é $\Theta(n^3)$.

Versão recursiva eficiente

MEMOIZED-MATRIX-CHAIN-ORDER (p, n)

- 1 **para** $i \leftarrow 1$ até n **faça**
- 2 **para** $j \leftarrow 1$ até n **faça**
- 3 $m[i, j] \leftarrow \infty$
- 4 **devolva** LOOKUP-CHAIN ($p, 1, n$)

Versão recursiva eficiente

LOOKUP-CHAIN (p, i, j)

```
1  se  $m[i, j] < \infty$ 
2      então devolva  $m[i, j]$ 
3  se  $i = j$ 
4      então  $m[i, j] \leftarrow 0$ 
5      senão para  $k \leftarrow i$  até  $j - 1$  faça
6           $q \leftarrow$  LOOKUP-CHAIN ( $p, i, k$ )
7               $+ p[i-1]p[k]p[j]$ 
8               $+$  LOOKUP-CHAIN ( $p, k+1, j$ )
9          se  $q < m[i, j]$ 
10             então  $m[i, j] \leftarrow q$ 
11 devolva  $m[i, j]$ 
```

Versão recursiva eficiente

LOOKUP-CHAIN (p, i, j)

```
1  se  $m[i, j] < \infty$ 
2      então devolva  $m[i, j]$ 
3  se  $i = j$ 
4      então  $m[i, j] \leftarrow 0$ 
5      senão para  $k \leftarrow i$  até  $j - 1$  faça
6           $q \leftarrow$  LOOKUP-CHAIN ( $p, i, k$ )
7              +  $p[i-1]p[k]p[j]$ 
8              + LOOKUP-CHAIN ( $p, k+1, j$ )
9          se  $q < m[i, j]$ 
10             então  $m[i, j] \leftarrow q$ 
11 devolva  $m[i, j]$ 
```

Quanto tempo consome?

Versão recursiva eficiente

LOOKUP-CHAIN (p, i, j)

```
1  se  $m[i, j] < \infty$ 
2      então devolva  $m[i, j]$ 
3  se  $i = j$ 
4      então  $m[i, j] \leftarrow 0$ 
5  senão para  $k \leftarrow i$  até  $j - 1$  faça
6       $q \leftarrow$  LOOKUP-CHAIN ( $p, i, k$ )
7          +  $p[i-1]p[k]p[j]$ 
8          + LOOKUP-CHAIN ( $p, k+1, j$ )
9      se  $q < m[i, j]$ 
10         então  $m[i, j] \leftarrow q$ 
11 devolva  $m[i, j]$ 
```

Quanto tempo consome?

Como encontrar uma parentização ótima?

Ingredientes de programação dinâmica

- ▶ **Subestrutura ótima**: soluções ótimas contém soluções ótimas de subproblemas.
- ▶ **Subestrutura**: decomponha o problema em subproblemas menores e, com sorte, mais simples.
- ▶ **Bottom-up**: combine as soluções dos problemas menores para obter soluções dos maiores.
- ▶ **Tabela**: armazene as soluções dos subproblemas em uma tabela, pois soluções dos subproblemas são consultadas várias vezes.
- ▶ **Número de subproblemas**: para a eficiência do algoritmo é importante que o número de subproblemas resolvidos seja 'pequeno'.
- ▶ **Memoized**: versão *top-down*, recursão com tabela.

Exercício

O algoritmo **MATRIX-CHAIN-ORDER** determina o **número mínimo** de multiplicações escalares necessário para calcular produto $A_1 A_2 \cdots A_n$.

Na aula, mencionamos uma maneira de obter uma parentização ótima a partir dos cálculos feitos, usando para isso um dado a mais que podemos guardar no decorrer do algoritmo.

Faça os ajustes sugeridos na aula, de modo a guardar esse dado extra, e devolvê-lo junto com o valor $m[1, n]$.

Faça uma rotina que recebe a informação extra armazenada pelo algoritmo acima e imprime uma parentização ótima das matrizes $A_1 A_2 \cdots A_n$.

Exercícios

Exercício 13.A [CLRS 15.2-1]

Encontre a maneira ótima de fazer a multiplicação iterada das matrizes cujas dimensões são $(5, 10, 3, 12, 5, 50, 6)$.

Exercício 13.B [CLRS 15.2-5]

Mostre que são necessários exatamente $n - 1$ pares de parênteses para especificar exatamente a ordem de multiplicação de $A_1 \cdot A_2 \cdots A_n$.

Exercício 13.C [CLRS 15.3-2]

Desenhe a árvore de recursão para o algoritmo [MERGESORT](#) aplicado a um vetor de 16 elementos. Por que a técnica de programação dinâmica não é capaz de acelerar o algoritmo?

Mais exercícios

Exercício 13.D [CLRS 15.3-5 expandido]

Considere o seguinte algoritmo para determinar a ordem de multiplicação de uma cadeia de matrizes $A_1, A_2, \dots, A_n$ de dimensões $p_0, p_1, \dots, p_n$: primeiro, escolha k que minimize p_k ; depois, determine recursivamente as ordens de multiplicação de $A_1, \dots, A_k$ e $A_{k+1}, \dots, A_n$. Esse algoritmo produz uma ordem que minimiza o número total de multiplicações escalares? E se k for escolhido de modo a maximizar p_k ? E se k for escolhido de modo a minimizar p_k ?

Exercício 13.E

Prove que o número de execuções da linha 9 em `MATRIX-CHAIN-ORDER` é $O(n^3)$.