AULA 13

Programação dinâmica

CLRS 15.1–15.3

= “recursão–com–tabela”

= transformação inteligente de recursão em iteração

Programação dinâmica

"Dynamic programming is a fancy name for divide-and-conquer with a table. Instead of solving subproblems recursively, solve them sequentially and store their solutions in a table. The trick is to solve them in the right order so that whenever the solution to a subproblem is needed, it is already available in the table. Dynamic programming is particularly useful on problems for which divide-and-conquer appears to yield an exponential number of subproblems, but there are really only a small number of subproblems repeated exponentially often. In this case, it makes sense to compute each solution the first time and store it away in a table for later use, instead of recomputing it recursively every time it is needed."

I. Parberry, *Problems on Algorithms*, Prentice Hall, 1995.

Números de Fibonacci

$$F_0 = 0 \qquad F_1 = 1 \qquad F_n = F_{n-1} + F_{n-2}$$

n	0	1	2	3	4	5	6	7	8	9
F_n	0	1	1	2	3	5	8	13	21	34

Números de Fibonacci

$$F_0 = 0 \qquad F_1 = 1 \qquad F_n = F_{n-1} + F_{n-2}$$

n	0	1	2	3	4	5	6	7	8	9
F_n	0	1	1	2	3	5	8	13	21	34

Algoritmo recursivo para F_n:

```
FIBO-REC (n)
    se n ≤ 1
        então devolva n
        senão a ← FIBO-REC (n − 1)
              b ← FIBO-REC (n − 2)
              devolva a + b
```

Consumo de tempo

FIBO-REC (n)
1 **se** $n \leq 1$
2 **então devolva** n
3 **senão** $a \leftarrow$ FIBO-REC $(n-1)$
4 $b \leftarrow$ FIBO-REC $(n-2)$
5 **devolva** $a + b$

n	16	32	40	41	42	43	44	45	47
tempo	0.002	0.06	2.91	4.71	7.62	12.37	19.94	32.37	84.50

tempo em segundos.

$F_{47} = 2971215073$

Consumo de tempo

$T(n) :=$ número de somas feitas por FIBO-REC (n)

linha	número de somas
1-2	$= 0$
3	$= T(n-1)$
4	$= T(n-2)$
5	$= 1$

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

Recorrência

$$T(0) = 0$$

$$T(1) = 0$$

$$T(n) = T(n-1) + T(n-2) + 1 \text{ para } n = 2, 3, \ldots$$

A que classe Ω pertence $T(n)$?
A que classe O pertence $T(n)$?

Recorrência

$$T(0) = 0$$

$$T(1) = 0$$

$$T(n) = T(n-1) + T(n-2) + 1 \text{ para } n = 2, 3, \ldots$$

A que classe Ω pertence $T(n)$?
A que classe O pertence $T(n)$?

Solução: $T(n) > (3/2)^n$ para $n \geq 6$.

n	0	1	2	3	4	5	6	7	8	9
T_n	0	0	1	2	4	7	12	20	33	54
$(3/2)^n$	1	1.5	2.25	3.38	5.06	7.59	11.39	17.09	25.63	38.4

Recorrência

Prova: $T(6) = 12 > 11.40 > (3/2)^6$ e $T(7) = 20 > 18 > (3/2)^7$.
Se $n \geq 8$, então

$$
\begin{aligned}
T(n) &= T(n-1) + T(n-2) + 1 \\
&\overset{\text{hi}}{>} (3/2)^{n-1} + (3/2)^{n-2} + 1 \\
&= (3/2 + 1)\,(3/2)^{n-2} + 1 \\
&> (5/2)\,(3/2)^{n-2} \\
&> (9/4)\,(3/2)^{n-2} \\
&= (3/2)^2 (3/2)^{n-2} \\
&= (3/2)^n\,.
\end{aligned}
$$

Logo, $T(n)$ é $\Omega((3/2)^n)$.

Verifique que $T(n)$ é $\mathrm{O}(2^n)$.

Exercícios

Prove que

$$T(n) = \frac{\phi^{n+1} - \hat{\phi}^{n+1}}{\sqrt{5}} - 1 \quad \text{para } n = 0, 1, 2, \ldots$$

onde

$$\phi = \frac{1+\sqrt{5}}{2} \approx 1{,}61803 \quad \text{e} \quad \hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -0{,}61803.$$

Prove que $1 + \phi = \phi^2$.

Prove que $1 + \hat{\phi} = \hat{\phi}^2$.

Mais exercícios

[CLRS 3.2-6] Prove que

$$F_n = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n) \quad \text{para } n = 0, 1, 2, \ldots$$

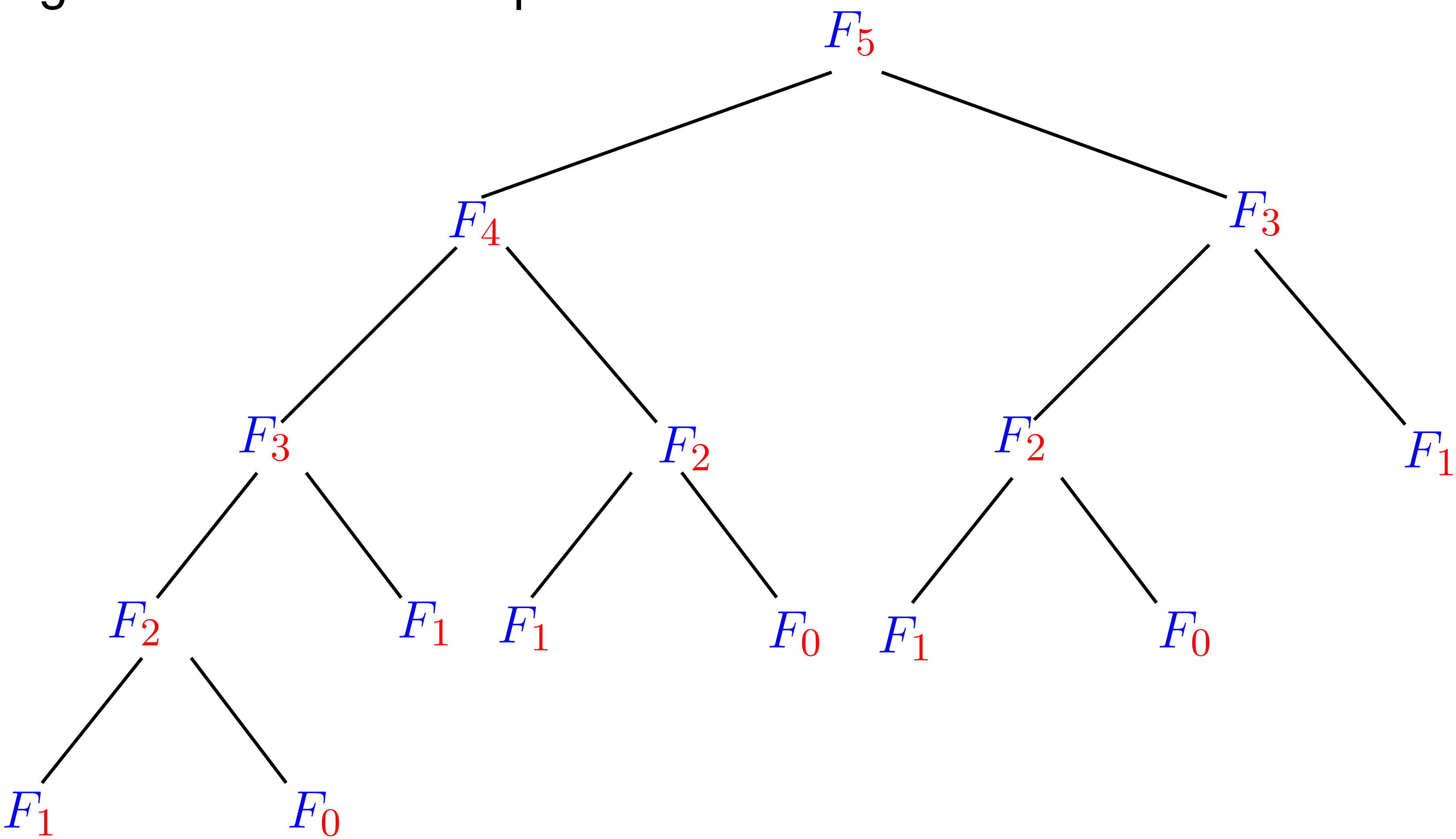
[CLRS 3.2-7] Prove que $F_{i+2} \geq \phi^i$ para $i = 2, 3, \ldots$

Veja o exercício [CLRS 4.5].

Consumo de tempo

Consumo de tempo é exponencial.

Algoritmo resolve subproblemas muitas vezes.



Resolve subproblemas muitas vezes

```
FIBO-REC(5)
  FIBO-REC(4)
    FIBO-REC(3)
      FIBO-REC(2)
        FIBO-REC(1)
        FIBO-REC(0)
      FIBO-REC(1)
    FIBO-REC(2)
      FIBO-REC(1)
      FIBO-REC(0)
  FIBO-REC(3)
    FIBO-REC(2)
      FIBO-REC(1)
      FIBO-REC(0)
    FIBO-REC(1)
```

FIBO-REC(5) = 5

Resolve subproblemas muitas vezes

```
FIBO-REC(8)
  FIBO-REC(7)
    FIBO-REC(6)
      FIBO-REC(5)
        FIBO-REC(4)
          FIBO-REC(3)
            FIBO-REC(2)
              FIBO-REC(1)
              FIBO-REC(0)
            FIBO-REC(1)
          FIBO-REC(2)
            FIBO-REC(1)
            FIBO-REC(0)
        FIBO-REC(3)
          FIBO-REC(2)
            FIBO-REC(1)
            FIBO-REC(0)
          FIBO-REC(1)
      FIBO-REC(4)
        FIBO-REC(3)
          FIBO-REC(2)
            FIBO-REC(1)
            FIBO-REC(0)
```

```
        FIBO-REC(1)
      FIBO-REC(2)
        FIBO-REC(1)
        FIBO-REC(0)
  FIBO-REC(5)
    FIBO-REC(4)
      FIBO-REC(3)
        FIBO-REC(2)
          FIBO-REC(1)
          FIBO-REC(0)
        FIBO-REC(1)
      FIBO-REC(2)
        FIBO-REC(1)
        FIBO-REC(0)
    FIBO-REC(3)
      FIBO-REC(2)
        FIBO-REC(1)
        FIBO-REC(0)
      FIBO-REC(1)
FIBO-REC(6)
  FIBO-REC(5)
    FIBO-REC(4)
      FIBO-REC(3)
```

```
      FIBO-REC(2)
        FIBO-REC(1)
        FIBO-REC(0)
      FIBO-REC(1)
    FIBO-REC(2)
      FIBO-REC(1)
      FIBO-REC(0)
  FIBO-REC(3)
    FIBO-REC(2)
      FIBO-REC(1)
      FIBO-REC(0)
    FIBO-REC(1)
FIBO-REC(4)
  FIBO-REC(3)
    FIBO-REC(2)
      FIBO-REC(1)
      FIBO-REC(0)
    FIBO-REC(1)
  FIBO-REC(2)
    FIBO-REC(1)
    FIBO-REC(0)
```

Algoritmo de programação dinâmica

FIBO (n)
$f[0] \leftarrow 0$
$f[1] \leftarrow 1$
para $i \leftarrow 2$ **até** n **faça**
    $f[i] \leftarrow f[i-1] + f[i-2]$
devolva $f[n]$

Note a tabela $f[0 \,.\,.\, n{-}1]$.

f					$\star$	$\star$	$??$			

Consumo de tempo é $\Theta(n)$.

Algoritmo de programação dinâmica

Versão com economia de espaço.

FIBO (n)
se $n = 0$ **então devolva** 0
f_ant $\leftarrow 0$
f_atual $\leftarrow 1$
para $i \leftarrow 2$ **até** n **faça**
    f_prox $\leftarrow$ f_atual $+$ f_ant
    f_ant $\leftarrow$ f_atual
    f_atual $\leftarrow$ f_prox
devolva f_atual

Versão recursiva eficiente

MEMOIZED-FIBO (f, n)

para $i \leftarrow 0$ **até** n **faça**
    $f[i] \leftarrow -1$
devolva LOOKUP-FIBO (f, n)

LOOKUP-FIBO (f, n)

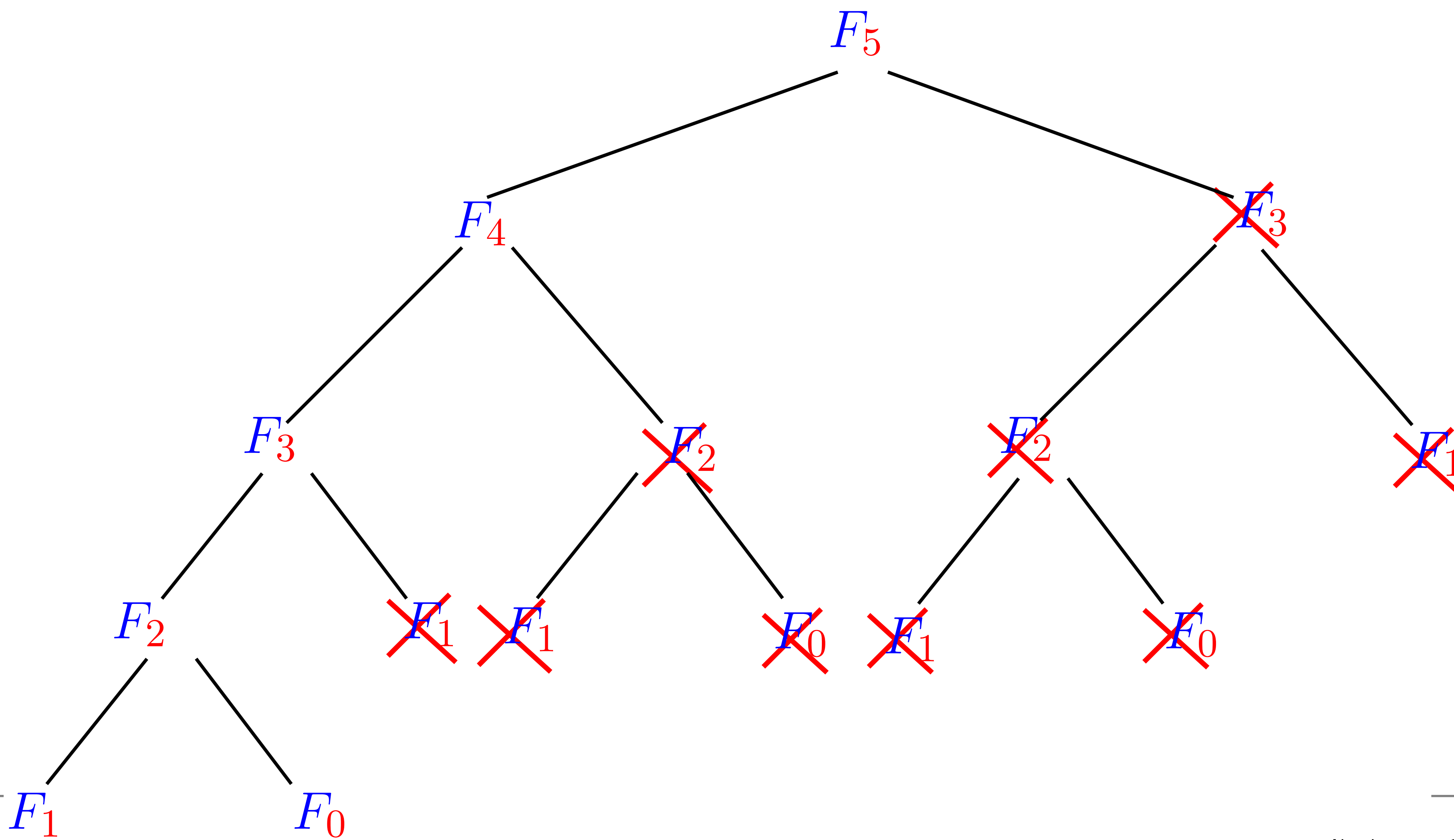
se $f[n] \geq 0$
    então devolva $f[n]$
se $n \leq 1$
    então $f[n] \leftarrow n$
    senão $f[n] \leftarrow$ LOOKUP-FIBO$(f, n-1)$
        $+$ LOOKUP-FIBO$(f, n-2)$
devolva $f[n]$

Não recalcula valores de f.

Consumo de tempo

Consumo de tempo é $\Theta(n)$.

Algoritmo resolve cada subproblemas apenas uma vezes.



Multiplicação iterada de matrizes

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

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

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

para $i \leftarrow 1$ **até** p **faça**
    para $j \leftarrow 1$ **até** d **faça**
        $AB[i,j] \leftarrow 0$
        para $k \leftarrow 1$ **até** q **faça**
            $AB[i,j] \leftarrow AB[i,j] + A[i,k] \cdot B[k,j]$

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

Multiplicação iterada

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

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

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

Exemplo: $A_1 \cdot A_2 \cdot A_3$

$$
\begin{array}{ccccccc}
10 & A_1 & 100 & A_2 & 5 & A_3 & 50
\end{array}
$$

$((A_1\ A_2)\ A_3)$	7500	multiplicações escalares
$(A_1\ (A_2\ A_3))$	75000	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: $(A_i \cdots A_k)\,(A_{k+1} \cdots A_j)$

Decomposição sugere um algoritmo recursivo.

Algoritmo recursivo

Recebe $p[i-1\,..\,j]$ e devolve o número mínimo de multiplicações escalares para calcular $A_i \cdots A_j$.

REC-MAT-CHAIN (p, i, j)
  se $i = j$
      então devolva 0
  $m \leftarrow \infty$
  para $k \leftarrow i$ **até** $j - 1$ **faça**
      $q_1 \leftarrow$ REC-MAT-CHAIN (p, i, k)
      $q_2 \leftarrow$ REC-MAT-CHAIN $(p, k+1, j)$
      $q \leftarrow q_1 + p[i-1]p[k]p[j] + q_2$
      se $q < m$
          então $m \leftarrow q$
  devolva m

Consumo de tempo?

Consumo de tempo

A plataforma utilizada nos experimentos é um PC rodando Linux Debian ?.? com um processador Pentium II de 233 MHz e 128MB de memória RAM .

O programa foi compilados com o gcc versão ?? e opção de compilação “-O2”.

n	3	6	10	20	25
tempo	$0.0s$	$0.0s$	$0.01s$	$201s$	$567m$

Consumo de tempo

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

$$
\begin{aligned}
T(1) &= 0 \\
T(n) &= \sum_{h=1}^{n-1} (T(h) + T(n-h) + 1) \\
&= 2\sum_{h=2}^{n-1} T(h) \ + \ (n-1) \\
&= 2(T(2) + \cdots + T(n{-}1)) \ + \ (n-1) \ \text{ para } n \geq 2
\end{aligned}
$$

Consumo de tempo

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

$$
\begin{aligned}
T(1) &= 0 \\
T(n) &= \sum_{h=1}^{n-1} (T(h) + T(n-h) + 1) \\
&= 2 \sum_{h=2}^{n-1} T(h) \; + \; (n-1) \\
&= 2(T(2) + \cdots + T(n-1)) \; + \; (n-1) \quad \text{para } n \geq 2
\end{aligned}
$$

Fácil verificar: $T(n) \geq 2^{n-2}$ para $n \geq 2$

Recorrência

n	1	2	3	4	5	6	7	8
$T(n)$	0	1	4	13	40	121	364	1093
2^{n-2}	0.5	1	2	8	16	32	64	128

Prova: Para $n = 2$, $T(2) = 1 = 2^{2-2}$.
Para $n \geq 3$,

$$\begin{aligned} T(n) &= 2(T(2) + \cdots + T(n-1)) + n - 1 \\ &\overset{\text{hi}}{\geq} 2(2^0 + \cdots + 2^{n-3}) + n - 1 \\ &> 2^0 + \cdots + 2^{n-3} + n - 1 \\ &= 2^{n-2} - 1 + n - 1 \\ &> 2^{n-2} \quad \text{(pois } n \geq 3\text{).} \end{aligned}$$

Conclusão

O consumo de tempo do algoritmo
REC-MAT-CHAIN é $\Omega(2^n)$.

Fórmula fechada

Temos que

$$\begin{aligned} T(n) &= 2(T(2) + \cdots + T(n-2) + T(n-1)) + n - 1 \\ T(n-1) &= 2(T(2) + \cdots + T(n-2)) \qquad\qquad\qquad + n - 2 \end{aligned}$$

Logo,

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

ou seja

$$T(n) = 3T(n-1) + 1 \; .$$

Portanto,

$$T(n) = \frac{3^{n-1} - 1}{2} \; .$$

Nova conclusão

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

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

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

Programação dinâmica

$m[i,j] =$ 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:

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

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	j
1	0								
2		0	$\star$	$\star$	$\star$	??			
3			0			$\star$			
4				0		$\star$			
5					0	$\star$			
6						0			
7							0		
8								0	

i

Simulação

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

Simulação

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

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

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

$i \backslash j$	1	2	3	4	5	6
1	0	2000				
2		0	??			
3			0			
4				0		
5					0	
6						0

Simulação

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

	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

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

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

Simulação

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

	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

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

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

Simulação

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

	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

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

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

Simulação

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

	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

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

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

Simulação

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

	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

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

	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

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

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

Simulação

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

	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

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

	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

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

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

Simulação

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

	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

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

	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

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

	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

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

	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

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

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

Simulação

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

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

j

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

Simulação

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

	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

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

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

j

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

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

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

	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

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

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

j

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

Simulação

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

	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

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

	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

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

	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

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

	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

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

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

Simulação

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

	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

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

	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

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

	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

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

	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	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	
i							

Simulação

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

	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

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

	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

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

	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

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

	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

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

$i \backslash j$	1	2	3	4	5	6
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

Simulação

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

	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

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

	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

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

	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

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

	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

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

	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

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

i \ j	1	2	3	4	5	6
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

Algoritmo de programação dinâmica

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

MATRIX-CHAIN-ORDER (p, n)

```
para i ← 1 até n faça
      m[i, i] ← 0
para l ← 2 até n faça
      para i ← 1 até n − l + 1 faça
            j ← i + l − 1
            m[i, j] ← ∞
            para k ← i até j − 1 faça
                  q ← m[i, k] + p[i − 1]p[k]p[j] + m[k+1, j]
                  se q < m[i, j]
                        então m[i, j] ← q
devolva m[1, n]
```

Correção e consumo de tempo

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

Correção e consumo de tempo

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

Consumo de tempo: ???

Correção e consumo de tempo

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

Consumo de tempo: $\mathrm{O}(n^3)$ (três loops encaixados)

Correção e consumo de tempo

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

Consumo de tempo: $\mathrm{O}(n^3)$ (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 das subcadeias $A_i \cdots A_j$ de comprimento l

Consumo de tempo: $\mathrm{O}(n^3)$ (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	$1, \ldots, n-1$	$(n-1) \cdot 1$
3	$1, \ldots, n-2$	$(n-2) \cdot 2$
4	$1, \ldots, n-3$	$(n-3) \cdot 3$
$\ldots$	$\ldots$	$\ldots$
$n-1$	$1, 2$	$2 \cdot (n-2)$
n	1	$1 \cdot (n-1)$
	total	$\sum_{h=1}^{n-1} h(n-h)$

Consumo de tempo

Para $n \geq 6$, $\sum_{h=1}^{n-1} h(n-h) \; =$

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

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)

para $i \leftarrow 1$ **até** n **faça**

  para $j \leftarrow 1$ **até** n **faça**

    $m[i, j] \leftarrow \infty$

devolva LOOKUP-CHAIN $(p, 1, n)$

Versão recursiva eficiente

LOOKUP-CHAIN (p, i, j)

se $m[i, j] < \infty$
    então devolva $m[i, j]$
se $i = j$
    então $m[i, j] \leftarrow 0$
    senão para $k \leftarrow i$ **até** $j - 1$ **faça**
        $q_1 \leftarrow$ LOOKUP-CHAIN (p, i, k)
        $q_2 \leftarrow$ LOOKUP-CHAIN $(p, k{+}1, j)$
        $q \leftarrow q_1 + p[i{-}1]p[k]p[j] + q_2$
        se $q < m[i, j]$
            então $m[i, j] \leftarrow q$
devolva $m[1, n]$

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.

Exercícios

Exercício 19.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 19.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 19.C [CLRS 15.3-2]
Desenhe a árvore de recursão para o algoritmo MERGE-SORT aplicado a um vetor de 16 elementos. Por que a técnica de programação dinâmica não é capaz de acelerar o algoritmo?

Exercício 19.D [CLRS 15.3-5 expandido]
Considere o seguinte algoritmo para determinar a ordem de multiplicação de uma cadeia de matrizes $A_1, A_2, \ldots, A_n$ de dimensões $p_0, p_1, \ldots, p_n$: primeiro, escolha k que minimize p_k; depois, determine recursivamente as ordens de multiplicação de $A_1, \ldots, A_k$ e $A_{k+1}, \ldots, A_n$. 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?

Mais exercícios

Exercício 19.E
Prove que o número de execuções da linha 9 em MATRIX-CHAIN-ORDER é $\mathrm{O}(n^3)$.

Exercício 19.F [Subset-sum. CLRS 16.2-2 simplificado]
Escreva um algoritmo de programação dinâmica para o seguinte problema: dados números inteiros não-negativos $w_1, \ldots, w_n$ e W, encontrar um subconjunto K de $\{1, \ldots, n\}$ que satisfaça $\sum_{k \in K} w_k \leq W$ e maximize $\sum_{k \in K} w_k$. (Imagine que $w_1, \ldots, w_n$ são os tamanhos de arquivos digitais que você deseja armazenar em um disquete de capacidade W.)

Exercício 19.G [Mochila 0-1. CLRS 16.2-2]
O problema da mochila 0-1 consiste no seguinte: dados números inteiros não-negativos $v_1, \ldots, v_n, w_1, \ldots, w_n$ e W, queremos encontrar um subconjunto K de $\{1, \ldots, n\}$ que

$$\text{satisfaça } \sum_{k \in K} w_k \leq W \text{ e maximize } \sum_{k \in K} v_k.$$

(Imagine que w_i é o *peso* e v_i é o *valor* do objeto i.) Resolva o problema usando programação dinâmica.

Mais um exercício

Exercício 19.H [Partição equilibrada]
Seja S o conjunto das raízes raízes quadradas dos números $1, 2, \ldots, 500$. Escreva e teste um programa que determine uma partição (A, B) de S tal que a soma dos números em A seja tão próxima quanto possível da soma dos números em B. Seu algoritmo resolve o problema? ou só dá uma solução "aproximada"?
Uma vez calculados A e B, seu programa deve imprimir a diferença entre a soma de A e a soma de B e depois imprimir a lista dos quadrados dos números em um dos conjuntos.