

Sumário

- Elementos de Programação Dinâmica
- Problema da Mochila com e sem regras
- Subsequência Comum mais longa
- Multiplicações de Matrizes

Aulas 16 & 17



Algoritmos Greedy - Problema da Mochila Fracionária

Input:

- Mochila com capacidade V
- n itens com pesos w_1, \dots, w_n e valores v_1, \dots, v_n

Objetivo:

- Que itens devemos colocar na mochila de modo a transportarmos o maior valor possível, sabendo que podemos transportar uma quantidade fracionária de cada item?

$$\max \sum_{i=1}^n v_i \cdot x_i$$

$$\forall 1 \leq i \leq n. \quad 0 \leq x_i \leq 1$$

$$\sum_{i=1}^n x_i \cdot w_i \leq W$$

• $x_i \Rightarrow$ fração do produto i transportada

Algoritmos Greedy - Problema da Mochila Fracionária

Input:

- Mochila com capacidade V
- n items com pesos w_1, \dots, w_n e valores v_1, \dots, v_n

Escolha Greedy: Colocar na mochila a maior quantidade possível do item com maior valor por unidade de peso.

- Calcular para cada item i : v_i/w_i

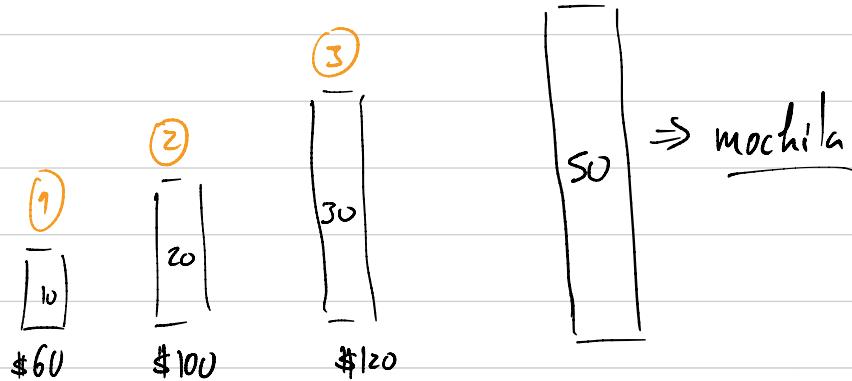
- Determinar o item k tal que:

$$v_k/w_k = \max \left\{ v_i/w_i \mid 1 \leq i \leq n \right\}$$

- Subproblema:

$$(x_k, V') = \begin{cases} (1, V - w_k) & \text{se } V > w_k \\ (w_k/v_k, 0) & \text{c.c.} \end{cases}$$

Algoritmos Greedy - Problema de Machila Fraccionaria



$$\frac{10}{60} = \frac{1}{6} \quad \frac{100}{240} = \frac{5}{12} \quad \frac{120}{360} = \frac{1}{3}$$

$$C = 60 + 100 + \frac{2}{3} \cdot 120$$

$$\textcircled{1} \quad x_1 = 1, \quad W^1 = 50 - 10 = 40$$

$$= 60 + 100 + 80$$

$$\textcircled{2} \quad x_2 = 1, \quad W^1 = 40 - 20 = 20$$

$$= 240$$

$$\textcircled{3} \quad x_3 = \frac{20}{30} = \frac{2}{3}, \quad V^1 = 0$$

Algoritmos Greedy - Problema da Mochila Fracionária

Fractional Knapsack (\vec{w}, \vec{v}, W)

let $\vec{x}[1..n]$ be a new array initialized to 0
let $W' = W$

```
for k=1 to n
    if  $\vec{w}[k] < W'$ 
         $\vec{x}[k] := 1$ 
         $W' := W' - \vec{w}[k]$ 
    else
         $\vec{x}[k] := W' / \vec{w}[k]$ 
    return  $\vec{x}$ 
```

return \vec{x}

- Admitimos que os arrays \vec{w} e \vec{v}
se encontram ordenados por ordem
decrescente de valor por unidade de
peso: $\underline{v_k/w_k}$

Algoritmos Greedy - Problema da Mochila Fracionária

Fractional Knapsack (\vec{v}, \vec{w}, w)

let $\vec{x}[1..n]$ be a new array initialized to 0
let $w' = w$

• Análise de Complexidade

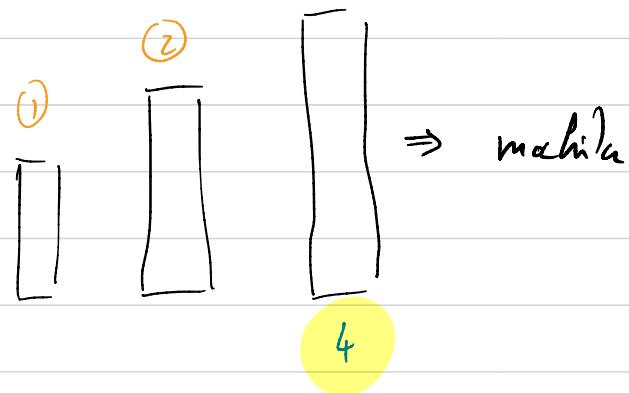
```
for k=1 to n
    if  $\vec{w}[k] < w'$ 
         $\vec{x}[k] := 1$ 
         $w' := w' - \vec{w}[k]$ 
    else
         $\vec{x}[k] := w' / \vec{w}[k]$ 
return  $\vec{x}$ 
```

return \vec{x}

$O(n)$

Problema da Mochila NÃO Fracionária

- Não é permitido transportar quantidades fracionárias



- Exemplo: Escolha greedy não conduz à solução óptima

$$\textcircled{1} \quad w_1 = 1 \quad v_1 = 3 \quad 3/1 = 3 > 4/4 = 1$$

$$\textcircled{2} \quad w_2 = 4 \quad v_2 = 4$$

Escolha greedy $\Rightarrow \textcircled{1}$ X

Problema da Mochila NÃO Fracionária

- Problema da mochila não fracionária
 - Com repetição: dispomos de quantidades ilimitadas de cada item
 - Sem repetição: dispomos de uma quantidade limitada de cada item
- Problema da mochila não fracionária com repetição

Input:

- $\vec{w} [1, \dots, n]$: vetor de pesos
- $\vec{v} [1, \dots, n]$: vetor de valores

Output:

- K : valor que conseguimos transportar na mochila.

Problema da Mochila NÃO Fracionária

- Problema da mochila não fracionária
 - Com repetição: dispomos de quantidades ilimitadas de cada item
 - Sem repetição: dispomos de uma quantidade limitada de cada item
- Problema da mochila não fracionária com repetição

Input:

- $\vec{w} [1, \dots, n]$: vetor de pesos
- $\vec{v} [1, \dots, n]$: vetor de valores

Output:

- k : valor que conseguimos transportar na mochila.

Solução Recursiva:

$$k(w) = \max \left\{ k(w-w_k) + v_k \mid \begin{array}{l} 1 \leq k \leq n, \\ w_k \leq w \end{array} \right\}$$

Problema da Mochila Não Fracionária

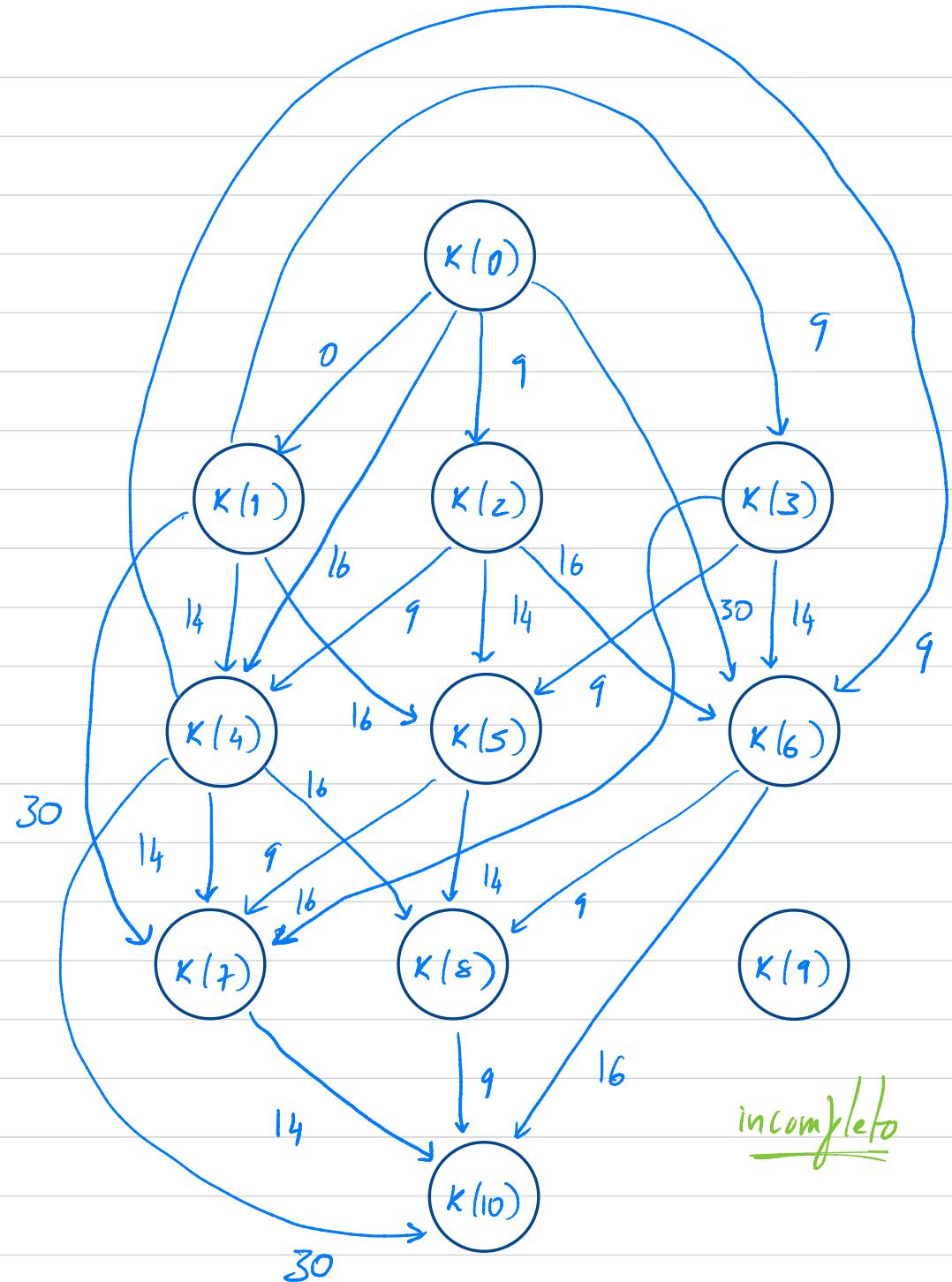
- Problema da Mochila com repetição

- $k(V)$: maior valor que conseguimos transportar numa mochila com capacidade V

$$k(W) = \max \left\{ k(W - v_k) + v_k \mid w_k \leq W \right\}$$

Exemplo:

Item	Peso	Valor	$\underline{\underline{W=10}}$
1	6	30 \$	
2	3	14 \$	
3	4	16 \$	
4	2	9 \$	



Problema da Mochila Não Fracionária

- Problema da Machila com reperição

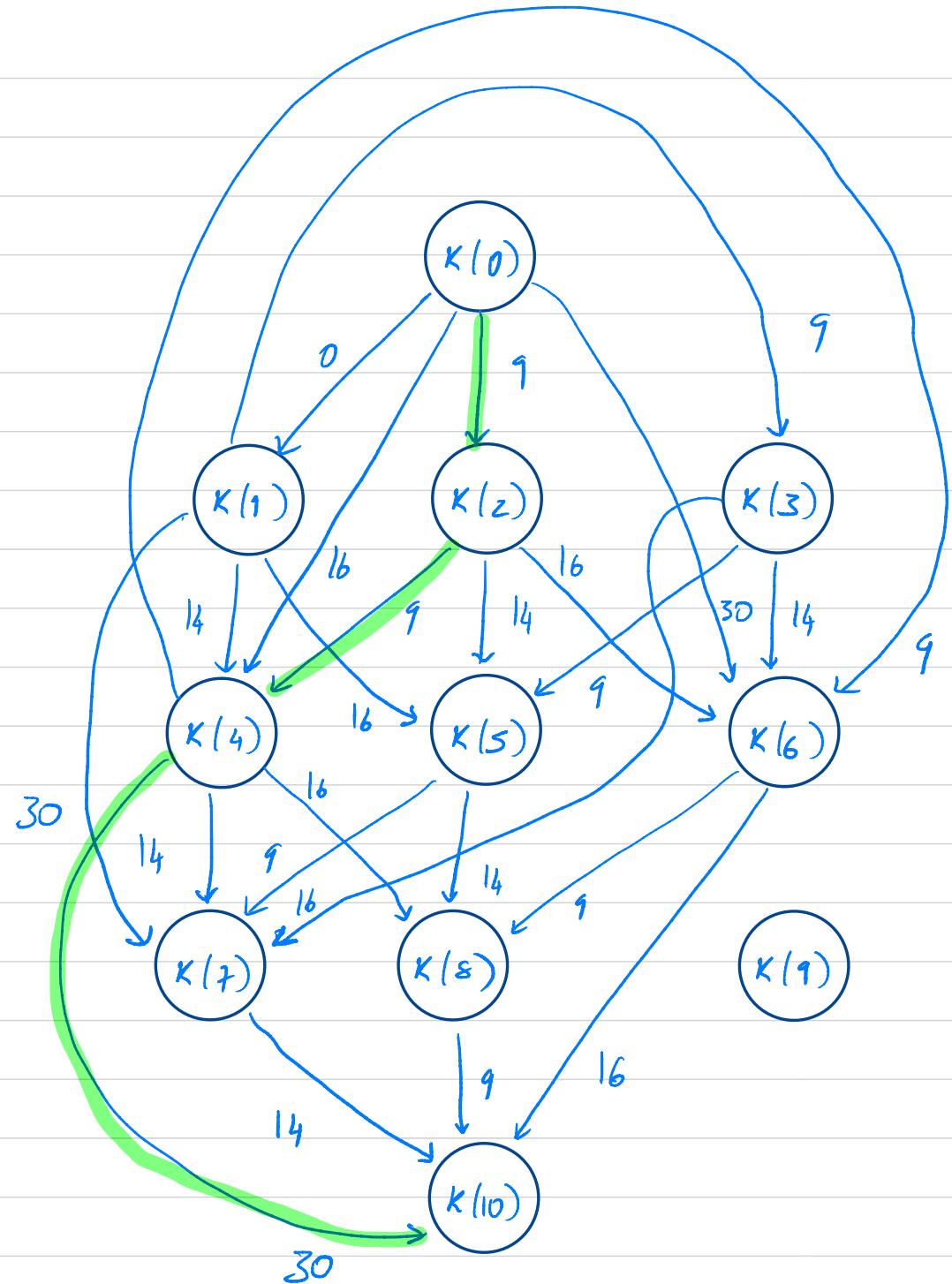
- $k(V)$: maior valor que conseguimos transportar numa mochila com capacidade V

$$K(w) = \max \left\{ K(w-w_k) + v_k \mid w_k \leq w \right\}$$

Exemplo :

Item	Peso	Value
1	6	30 ₱
2	3	14 ₱
3	4	16 ₱
4	2	9 ₱

Caminho mais longo
em DAG



Problema da Mochila Não Fracionária

- Problema da Mochila com repetição - Implementação naïf

$$K(W) = \max \left\{ K(W-w_k) + v_k \mid w_k \leq W \right\}$$

knapSack (W, \vec{v} , \vec{w} , n)

let $K = 0$

for $i=1$ to n

if $\vec{w}[i] \leq W$

$K := \max [K, KnapSack (W - \vec{w}[i], \vec{v}, \vec{w}, n)]$

return K

Problema da Mochila Não Fracionária

- Problema da Mochila com repetição - Implementação naïf

$$K(W) = \max \left\{ K(W-w_k) + v_k \mid w_k \leq W \right\}$$

knapSack (W, \vec{v}, \vec{w}, n)

let $k = 0$

for $i=1$ to n

if $\vec{w}[i] \leq W$

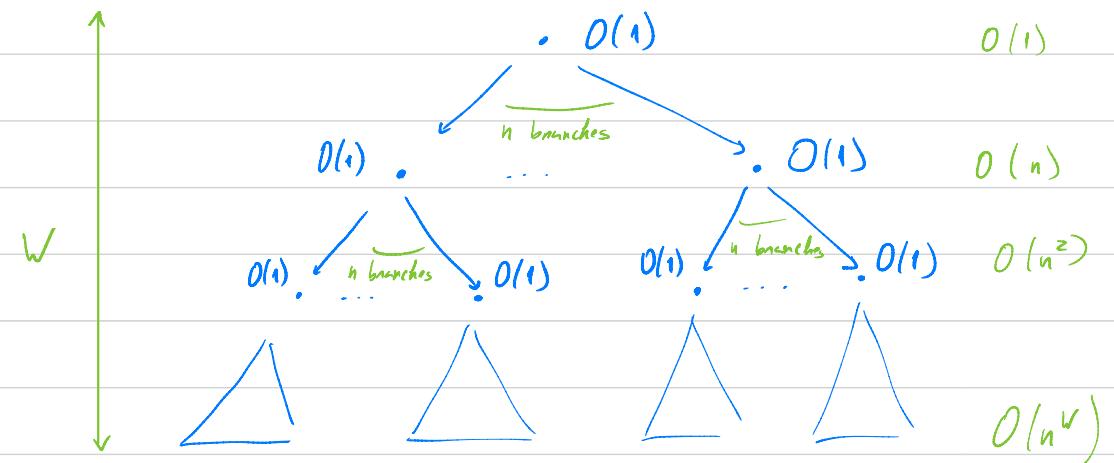
$K := \max [k, \text{knapSack} (W - \vec{w}[i], \vec{v}, \vec{w}, n)]$

return K

$$\underline{T(W)} = O(n^W)$$

Análise de Complexidade

$$T(W) = \begin{cases} n \times T(W-1) & \text{se } W > 0 \\ O(1) & \text{se } W = 0 \end{cases}$$



Problema da Mochila Não Fracionária

- Problema da Mochila com repetição - Programação Dinâmica

$$k(w) = \max \left\{ k(w-w_k) + v_k \mid w_k \leq w \right\}$$

Knapsack (w, \vec{v}, \vec{w}, n)

let $\vec{k}[0..w]$ be a new vector initialized to $\underline{0}$

$$\vec{k}[0] := 0$$

for $w=1$ to w

for $i:=1$ to n

if ($\vec{w}[i] \leq w$)

$$\vec{k}[w] = \max \left(\vec{k}[w], \vec{k}[w - \vec{w}[i]] + \vec{v}[i] \right)$$

Problema da Mochila Não Fracionária

- Problema da Mochila sem repetição - Programação Dinâmica

$$K(W) = \max \left\{ K(W-w_k) + v_k \mid w_k \leq W \right\}$$

Knapsack (W, \vec{v}, \vec{w}, n)

let $\vec{k}[0..W]$ be a new vector

$$\vec{k}[0] := 0$$

for $w=1$ to W

for $i:=1$ to n

if ($\vec{w}[i] \leq w$)

$$\vec{k}[w] = \max \left(\vec{k}[w], \vec{k}[w - \vec{w}[i]] + \vec{v}[i] \right)$$

Análise de complexidade

$$\underline{\mathcal{O}(n \cdot W)}$$

Problema da Machila Não Fracionária

Programação Dinâmica versus Memoization

Knapsack ($\underline{w}, \vec{v}, \vec{w}, n$)

let $\vec{k}[0..W]$ be a new vector

$$\vec{k}[0] := 0$$

for $w=1$ to \underline{w}

for $i:=1$ to n

$$\text{if } (\vec{w}[i] \leq w)$$

$$\vec{k}[w] = \max(\vec{k}[w], \vec{k}[w - \vec{w}[i]] + \vec{v}[i])$$

Programação Dinâmica

- Construção Incremental
da tabela de soluções

Knapsack' ($\underline{w}, \vec{k}, \vec{v}, \vec{w}, n$)

if ($\vec{k}[\underline{w}] \neq \infty$) return $\vec{k}[\underline{w}]$

$$\text{let } R := 0$$

for $i=1$ to n

$$\text{if } \vec{w}[i] \leq \underline{w}$$

$$R := \max(R, \text{Knapsack}'(\underline{w} - \vec{w}[i], \vec{k}, \vec{v}, \vec{w}, n) + \vec{v}[i])$$

$$\vec{k}[\underline{w}] := R$$

return R

Memoization

- A tabela é preenchida
à medida q vamos
precisando dos valores
respectivos.

Problema da Mochila Não Fracionária Sem Repetição

- Podemos usar cada elemento exactamente uma vez

$$\underbrace{k(\underline{w}, j)}_{=} = \boxed{\dots}$$

↳ Valor máximo q podemos transportar
numa mochila c/ capacidade \underline{w}
usando unicamente elementos do
conjunto $\underline{\{1, \dots, n\}}$.

$kmpschNoRep(\underline{w}, \vec{v}, \vec{w}, n)$
let $\vec{k}[\underline{j}]$ be a $\underline{w} \times n$ array

```
for  $w=0$  to  $\underline{w}$ 
   $\vec{k}[w, 0] := 0$ 
for  $i=0$  to  $n$ 
   $\vec{k}[0, i] := 0$ 
```

$\left. \begin{array}{c} \\ \\ \end{array} \right\}$ inicialização

Problema da Mochila Não Fracionária Sem Repetição

- Podemos usar cada elemento exactamente uma vez

$$k(\underline{w}, j) = \begin{cases} \max \left(k(\underline{w}, j-1), k(\underline{w} - w_j, j-1) + v_j \right) & \text{se } \underline{w} \geq w_j \\ k(\underline{w} - w_j, j-1) & \text{c.c.} \end{cases}$$

$kmp\text{suchNoRep}(\underline{w}, \vec{v}, \vec{w}, n)$
let $\vec{k}[\underline{j}[]]$ be a $\underline{w} \times n$ array

```
for  $w=0$  to  $\underline{w}$ 
   $\vec{k}[w, 0] := 0$ 
for  $i=0$  to  $n$ 
   $\vec{k}[0, i] := 0$ 
```

inicialização

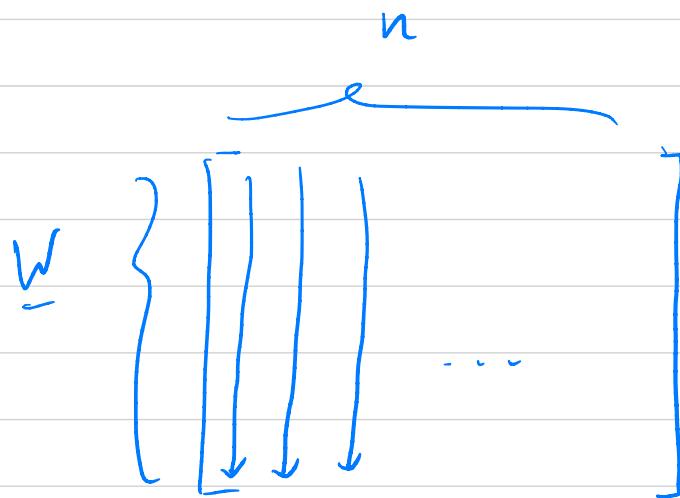
Problema da Mochila Não Fracionária Sem Repetição

- Podemos usar cada elemento exatamente uma vez

keys such No Rep (W, \vec{v}, \vec{w}, n)

let $\vec{k}[\cdot][\cdot]$ be a $W \times n$ array

```
for  $w=0$  to  $W$ 
   $\vec{k}[w, 0] := 0$ 
for  $i=0$  to  $n$ 
   $\vec{k}[0, i] := 0$ 
```



```
for  $i=1$  to  $n$ 
  for  $w=1$  to  $V$ 
     $\vec{k}[w, i] = \max \left( \vec{k}[w, i-1], \vec{k}[w - \vec{w}[i], i-1] + \vec{v}[i] \right)$ 
```

Ocultar $\vec{k}[W, n]$

Problema da Mochila Não Fracionária Sem Repetição

- Podemos usar cada elemento exatamente uma vez

$$k(\underline{w}, j) = \max \left(k(\underline{w}, j-1), k(\underline{w} - w_j, j-1) + v_j \right)$$

knapsackNoRep (\underline{w} , \vec{v} , \vec{w} , n)

let $\vec{k}[\underline{j}]$ be a $\underline{w} \times n$ array

Análise de Complexidade

$$\Theta(n \cdot V)$$

for $w=0$ to \underline{w}
 $\vec{k}[w, 0] := 0$

for $i=0$ to n
 $\vec{k}[0, i] := 0$

for $i=1$ to n

for $w=1$ to V

$$\vec{k}[w, i] = \max \left(\vec{k}[w, i-1], \vec{k}[w - \vec{w}[i], i-1] + \vec{v}[i] \right)$$

return $\vec{k}[V, n]$

Sumário Aula Anterior

- Problema da Mochila com Repetição

Input: $\vec{w} = \langle w_1, \dots, w_n \rangle$: vetor de pesos

$\vec{v} = \langle v_1, \dots, v_n \rangle$: vetor de valores

W : capacidade da mochila

$$\underline{K}(W) = \max \left\{ K(W - w_k) + v_k \mid w_k \leq W \right\}$$

↳ valor máximo q' eu consigo transportar numa mochila
com capacidade \underline{K}

- Problema da Mochila sem Repetição

$$K(W, j) = \max(K(W, j-1), K(W - v_j, j-1))$$

Sumário Aula Anterior

- Problema da Mochila com Repetição

Input: $\vec{w} = \langle w_1, \dots, w_n \rangle$: vetor de pesos

$\vec{v} = \langle v_1, \dots, v_n \rangle$: vetor de valores

W : capacidade da mochila

$$\underline{K}(W) = \max \left\{ K(W - w_k) + v_k \mid w_k \leq W \right\}$$

↳ valor máximo q' eu consigo transportar numa mochila
com capacidade \underline{K}

- Problema da Mochila sem Repetição

$$K(W, j) = \max(K(W, j-1), K(W - v_j, j-1))$$

Maior Subsequência Comum

Definição [Subsequência & Subsequência]

- Uma sequência $\vec{z} = \langle z_1, \dots, z_n \rangle$ é uma **subsequência** de uma outra sequência $\vec{x} = \langle x_1, \dots, x_m \rangle$ se existir uma sequência de índices $\langle i_1, \dots, i_n \rangle$ tal que:

$$\langle x_{i_1}, \dots, x_{i_n} \rangle = \langle z_1, \dots, z_n \rangle$$

- Dadas duas sequências $\vec{x} = \langle x_1, \dots, x_n \rangle$ e $\vec{y} = \langle y_1, \dots, y_m \rangle$, dizemos que $\vec{z} = \langle z_1, \dots, z_k \rangle$ é uma **subsequência comum** de \vec{x} e \vec{y} se \vec{z} é uma subsequência de \vec{x} e de \vec{y} .

Exemplo:

- $\vec{x} = \langle A, B, C, B, D, A, B \rangle$
- $\vec{y} = \langle B, D, C, A, B, A \rangle$

Maior Subsequência Comum

Definição [Subsequência & Subsequência]

- Uma sequência $\vec{z} = \langle z_1, \dots, z_n \rangle$ é uma **subsequência** de uma outra sequência $\vec{x} = \langle x_1, \dots, x_m \rangle$ se existir uma sequência de índices $\langle i_1, \dots, i_n \rangle$ tal que:

$$\langle x_{i_1}, \dots, x_{i_n} \rangle = \langle z_1, \dots, z_n \rangle$$

- Dadas duas sequências $\vec{x} = \langle x_1, \dots, x_n \rangle$ e $\vec{y} = \langle y_1, \dots, y_m \rangle$, dizemos que $\vec{z} = \langle z_1, \dots, z_k \rangle$ é uma **subsequência comum** de \vec{x} e \vec{y} se \vec{z} é uma subsequência de \vec{x} e de \vec{y} .

Exemplo:

$$\begin{aligned}\bullet \vec{x} &= \langle A, B, C, B, D, A, B \rangle \\ \bullet \vec{y} &= \langle B, D, C, A, B, A \rangle\end{aligned}$$

$$\bullet \vec{z} = \langle B, C, A, B \rangle$$

$$\text{E ainda: } \vec{z}'' = \langle B, D, A, B \rangle$$

$$\begin{aligned}\bullet \vec{x} &= \langle A, B, C, B, D, A, B \rangle \\ \bullet \vec{y} &= \langle B, D, C, A, B, A \rangle\end{aligned}$$

$$\bullet \vec{z}' = \langle B, C, B, A \rangle$$

Maior Subsequência Comum

Solução Recursiva

Input: $\vec{X}[1, \dots, n]$

$\vec{Y}[1, \dots, m]$

Output: maior subsequência comum entre

\vec{X} e \vec{Y} : $\vec{Z}[1, \dots, k]$

$C(i, j)$: Tamanho da maior subsequência comum entre $\vec{X}[1, \dots, i]$ e $\vec{Y}[1, \dots, j]$

$$C(i, j) = \left\{ \begin{array}{l} \text{(definição)} \\ \text{(caso base)} \\ \text{(caso recursivo)} \end{array} \right.$$

Maior Subsequência Comum

Solução Recursiva

Input: $\vec{x}[1, \dots, n]$

$\vec{y}[1, \dots, m]$

Output: maior subsequência comum entre

\vec{x} e \vec{y} : $\vec{z}[1, \dots, k]$

$c(i, j)$: Tamanho da maior subsequência comum entre $\vec{x}[1, \dots, i]$ e $\vec{y}[1, \dots, j]$

$$c(i, j) = \begin{cases} 0 & \text{se } i=0 \vee j=0 \\ c(i-1, j-1) + 1 & \text{se } \vec{x}[i] = \vec{y}[j] \\ \max(c(i-1, j), c(i, j-1)) & \text{c.c.} \end{cases}$$

Maior Subsequência Comum

Solução Recursiva

$$C(i, j) = \begin{cases} 0 & \text{se } i=0 \vee j=0 \\ C(i-1, j-1) + 1 & \text{se } \vec{x}[i] = \vec{y}[j] \\ \max(C(i-1, j), C(i, j-1)) & \text{c.c.} \end{cases}$$

$LCS(\vec{x}, \vec{y})$

let $n = \vec{x}.size()$

let $m = \vec{y}.size()$

let $C[i][j]$ be an $(n+1) \times (m+1)$ matrix

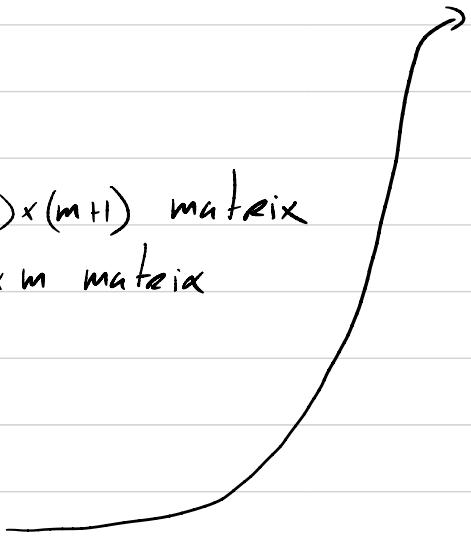
let $B[i][j]$ be an $n \times m$ matrix

for $i=0$ to n

$C[i, 0] := 0$

for $j=0$ to n

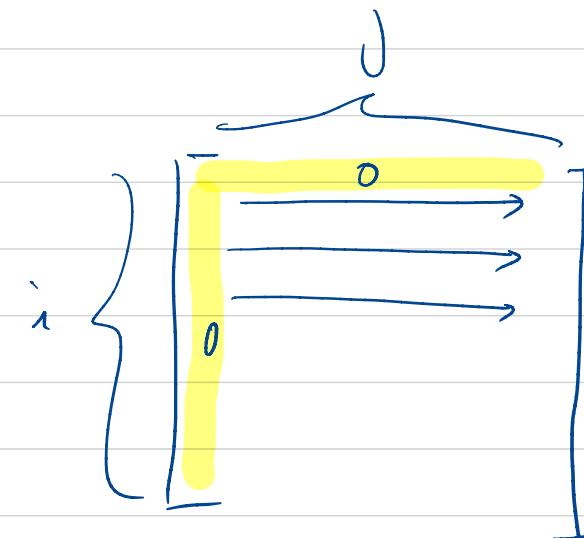
$C[0, j] := 0$



Maior Subsequência Comum

Solução Recursiva

$$C(i, j) = \begin{cases} 0 & \text{se } i=0 \vee j=0 \\ C(i-1, j-1) + 1 & \text{se } \vec{x}[i] = \vec{y}[j] \\ \max(C(i-1, j), C(i, j-1)) & \text{c.c.} \end{cases}$$



$LCS(\vec{x}, \vec{y})$

let $n = \vec{x}.size()$

let $m = \vec{y}.size()$

let $C[J][J]$ be an $(n+1) \times (m+1)$ matrix

let $B[J][J]$ be an $n \times m$ matrix

for $i=0$ to n

$C[i, 0] := 0$

for $j=0$ to n

$C[0, j] := 0$

```

for i=1 to n
  for j=1 to m
    if ( $\vec{x}[i] = \vec{y}[j]$ )
      | |  $C[i, j] := C[i-1, j-1] + 1$ ;  $B[i, j] := "R"$ 
    else if ( $C[i-1, j] \geq C[i, j-1]$ )
      | |  $C[i, j] := C[i-1, j]$ ;  $B[i, j] := "U"$ 
    else
      | |  $C[i, j] := C[i, j-1]$ ;  $B[i, j] := "L"$ 
return (C, B)
  
```

Maior Subsequência Comum

Solução Recursiva

$$C(i, j) = \begin{cases} 0 & \text{se } i=0 \vee j=0 \\ C(i-1, j-1) + 1 & \text{se } \vec{x}[i] = \vec{y}[j] \\ \max(C(i-1, j), C(i, j-1)) & \text{c.c.} \end{cases}$$

Análise de Complexidade

$LCS(\vec{x}, \vec{y})$

let $n = \vec{x}.size()$

let $m = \vec{y}.size()$

let $C[J][J]$ be an $(n+1) \times (m+1)$ matrix

let $B[J][J]$ be an $n \times m$ matrix

for $i=0$ to n

$C[i, 0] := 0$

for $j=0$ to n

$C[0, j] := 0$

```
for i=1 to n
  for j=1 to m
    if ( $\vec{x}[i] = \vec{y}[j]$ )
      | |  $C[i, j] := C[i-1, j-1] + 1$ ;  $B[i, j] := "R"$ 
    else if ( $C[i-1, j] \geq C[i, j-1]$ )
      | |  $C[i, j] := C[i-1, j]$ ;  $B[i, j] := "U"$ 
    else
      | |  $C[i, j] := C[i, j-1]$ ;  $B[i, j] := "L"$ 
return (C, B)
```

Maior Subsequência Comum

Solução Recursiva

$$C(i, j) = \begin{cases} 0 & \text{se } i=0 \vee j=0 \\ C(i-1, j-1) + 1 & \text{se } \vec{x}[i] = \vec{y}[j] \\ \max(C(i-1, j), C(i, j-1)) & \text{c.c.} \end{cases}$$

Análise de Complexidade

$\Theta(n \cdot m)$

$LCS(\vec{x}, \vec{y})$

let $n = \vec{x}.size()$

let $m = \vec{y}.size()$

let $C[i][j]$ be an $(n+1) \times (m+1)$ matrix

let $B[i][j]$ be an $n \times m$ matrix

for $i=0$ to n

$C[i, 0] := 0$

for $j=0$ to n

$C[0, j] := 0$

```
for i = 1 to n
    for j = 1 to m
        if ( $\vec{x}[i] == \vec{y}[j]$ )
            | |  $C[i, j] := C[i-1, j-1] + 1$ ;  $B[i, j] := "R"$ 
        | | else if ( $C[i-1, j] \geq C[i, j-1]$ )
            | | |  $C[i, j] := C[i-1, j]$ ;  $B[i, j] := "U"$ 
        | | else
            | | |  $C[i, j] := C[i, j-1]$ ;  $B[i, j] := "L"$ 
return (C, B)
```

Maior Subsequência Comum

	O	B	D	C	A	B	A
O	0	0	0	0	0	0	0
A	0						
B	0						
C	0						
B	0						
D	0		1				
A	0			1	1	1	1
B	0						

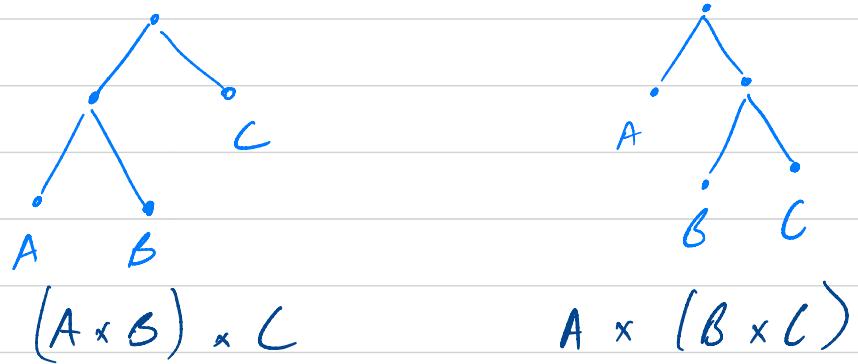
Maior Subsequência Comum

	O	B	D	C	A	B	A
O	0	0	0	0	0	0	0
A	0	0↑	0↑	0↑	1↖	1↖	1↑
B	0	1↖	1↖	1↖	1↑	2↑	2↖
C	0	1↑	1↑	2↑	2↖	2↑	2↑
B	0	1↖	1↑	2↑	2↑	3↑	3↖
D	0	1↑	2↑	2↑	2↑	3↑	3↑
A	0	1↑	2↑	2↑	3↖	3↑	4↑
B	0	1↖	2↑	2↑	3↑	4↖	4↑

Multiplicação de Matrizes

- $A \times B \times C$

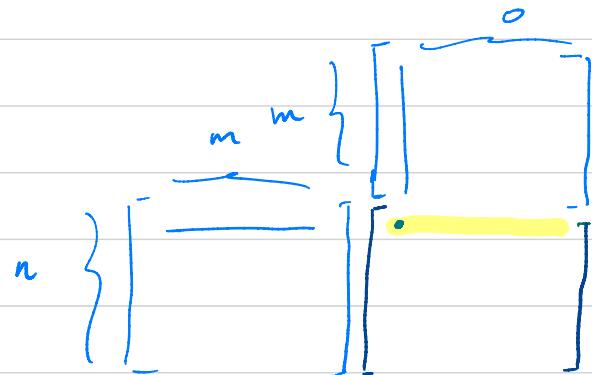
$$\downarrow \quad \downarrow \quad \downarrow \\ a \times b_1 \quad b_1 \times b_2 \quad b_2 \times c$$



- Queremos escolher a configuração que minimize o nº de multiplicações escalares.

Multiplicação de Matrizes

- $(n \times m) \times (m \times o) : n \times m \times o$



Custo da matriz:

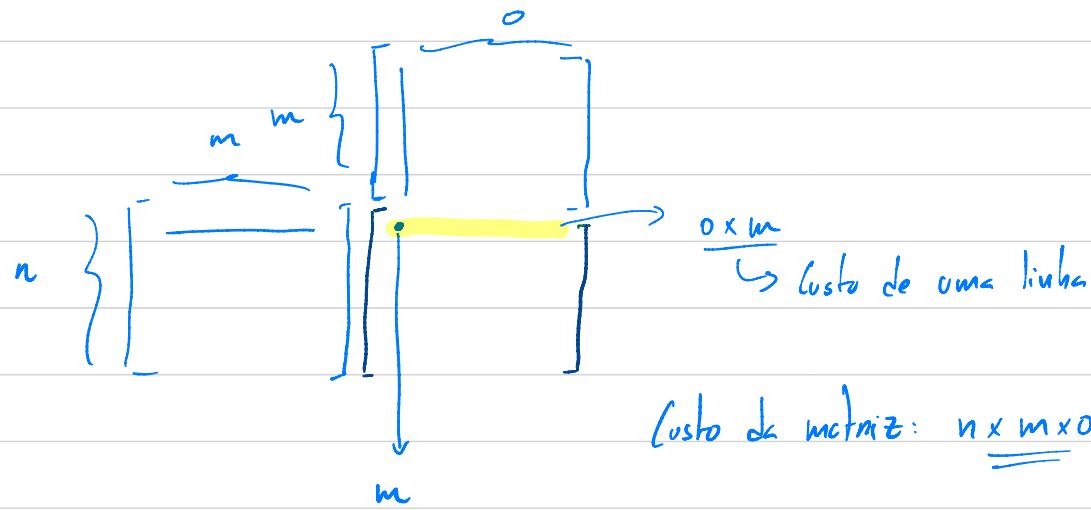
$$\bullet A \times B \times C$$
$$\downarrow \quad \downarrow \quad \downarrow$$
$$a \times b_1 \quad b_1 \times b_2 \quad b_2 \times c$$

$$\textcircled{1} \quad (A \times B) \times C \quad] \text{ Custo:}$$

$$\textcircled{2} \quad A \times (B \times C) \quad] \text{ Custo:}$$

Multiplicação de Matrizes

- $(n \times m) \times (m \times o) : n \times m \times o$



- $A \times B \times C$
 $\downarrow \quad \downarrow \quad \downarrow$
 $a \times b_1 \quad b_1 \times b_2 \quad b_2 \times c$

① $(A \times B) \times C$] Custo: $a \times b_1 \times b_2 + a \times b_2 \times c$
 $\underbrace{a \times b_2} \quad \underbrace{b_2 \times c}$

\neq

② $A \times (B \times C)$] Custo: $b_1 \times b_2 \times c + a \times b_1 \times c$
 $\underbrace{a \times b_1} \quad \underbrace{b_1 \times c}$

Multiplicação de Matrizes

$\vec{A} = \langle A_1, \dots, A_n \rangle \Rightarrow$ Queremos calcular $A_1 \times \dots \times A_n$

$C[i,j]$: menor custo da multiplicação $A_i \times \dots \times A_j$

$$C[i,j] = \left\{ \begin{array}{l} \end{array} \right.$$

$$\bullet A_1 \times \cdots \times A_k \times A_{k+1} \times \cdots \times A_j$$



$$m_1 \times m_k \qquad m_k \times m_j$$

Multiplicação de Matrizes

$\vec{A} = \langle A_1, \dots, A_n \rangle \Rightarrow$ Queremos calcular $A_1 \times \dots \times A_n$

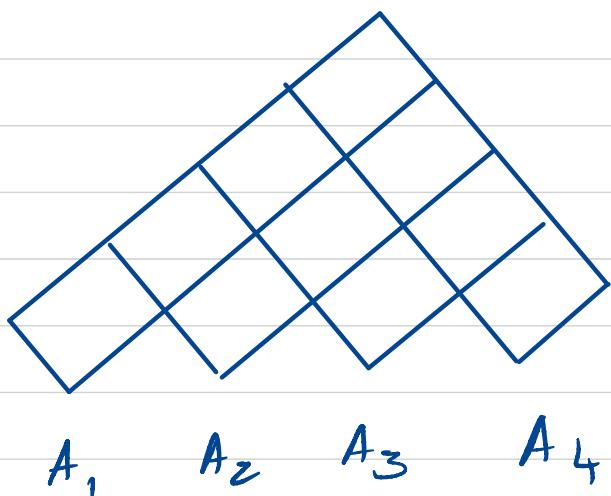
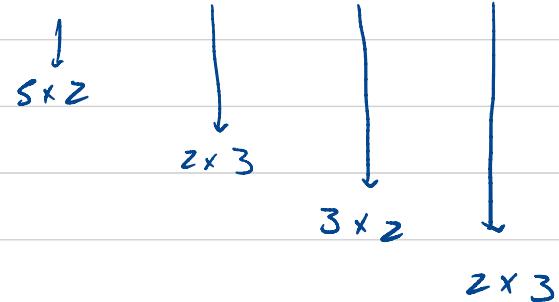
$C[i, j]$: menor custo da multiplicação $A_i \times \dots \times A_j$

$$\bullet \underbrace{A_i \times \dots \times A_k}_{m_{i-1} \times m_k} \times \underbrace{A_{k+1} \times \dots \times A_j}_{m_k \times m_j}$$

$$C[i, j] = \begin{cases} \min_{i \leq k < j} \{ C[i, k] + C[k+1, j] + m_{i-1} \times m_k \times m_j \} & \text{se } j > i \\ 0 & \text{se } i = j \\ \perp & \text{c.c.} \end{cases}$$

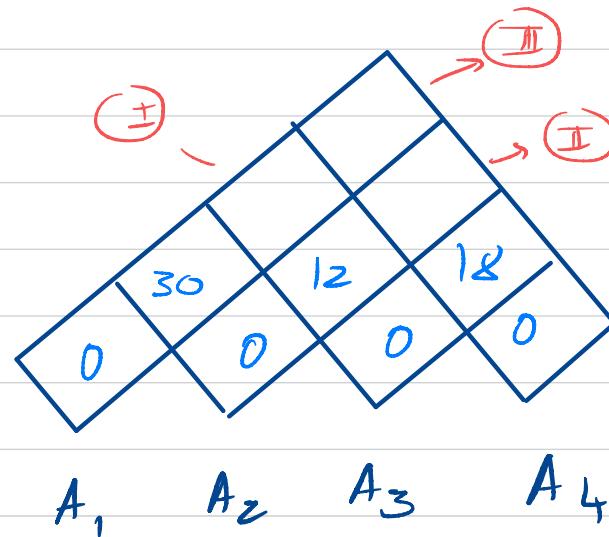
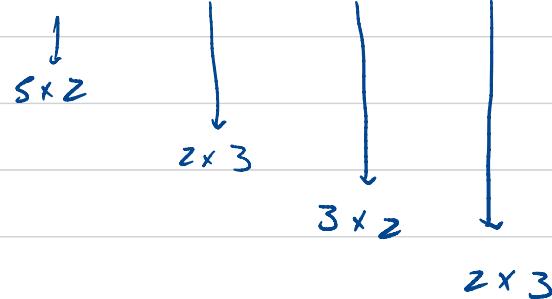
Multiplicação de Matrizes

- $A_1 \times A_2 \times A_3 \times A_4$



Multiplicação de Matrizes

- $A_1 \times A_2 \times A_3 \times A_4$



- $C[1,2] = 5 \times 2 \times 3 = 30$
- $C[2,3] = 2 \times 3 \times 2 = 12$
- $C[3,4] = 3 \times 2 \times 3 = 18$

① $C[1,3] \Rightarrow A_1 \times (A_2 \times A_3)$

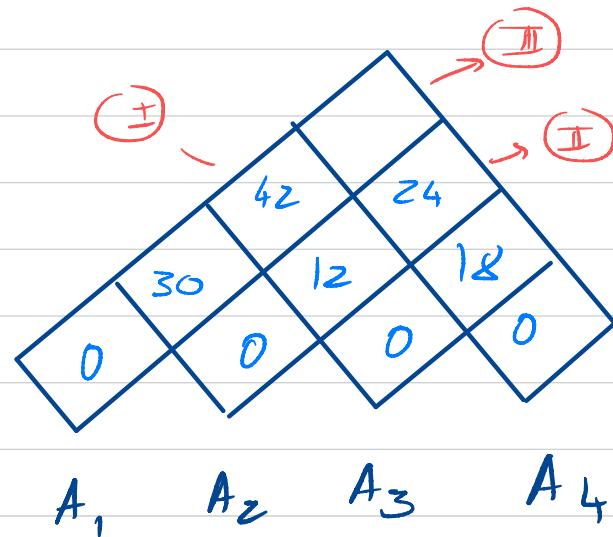
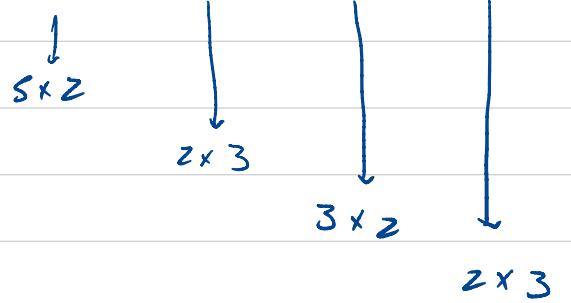
- $C[1,2] + C[3,2] + 5 \times 3 \times 2 = 30 + 0 + 30 = 60$
- $C[1,1] + C[2,3] + 5 \times 2 \times 3 = 0 + 12 + 30 = 42$

② $C[2,4] \Rightarrow (A_2 \times A_3) \times A_4$

- $C[2,3] + C[4,3] + 2 \times 2 \times 3 = 12 + 0 + 12 = 24$
- $C[2,2] + C[3,4] + 2 \times 3 \times 3 = 0 + 18 + 18 = 36$

Multiplicação de Matrizes

- $A_1 \times A_2 \times A_3 \times A_4$



- $C[1,2] = 5 \times 2 \times 3 = 30$

- $C[2,3] = 2 \times 3 \times 2 = 12$

- $C[3,4] = 3 \times 2 \times 3 = 18$

III

$$(A_1 \times (A_2 \times A_3)) \times A_4$$

- $42 + 30 + 12 = 72$

① $C[1,3] \Rightarrow A_1 \times (A_2 \times A_3)$

42

$$A_1 \times ((A_2 \times A_3) \times A_4)$$

- $42 + 30 + 12 = 72$

② $C[2,4] \Rightarrow (A_2 \times A_3) \times A_4$

24

$$(A_1 \times A_2) \times (A_3 \times A_4)$$

- $30 + 18 + 5 \times 3 \times 3 = \underline{\underline{93}}$

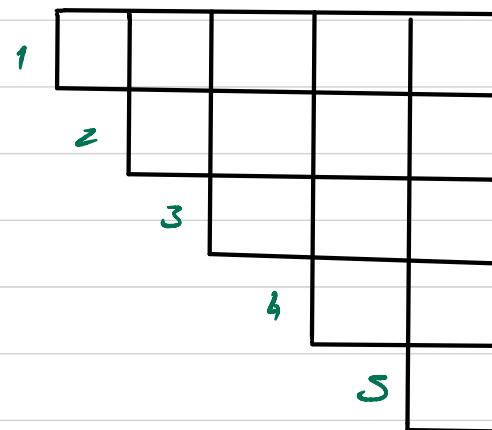
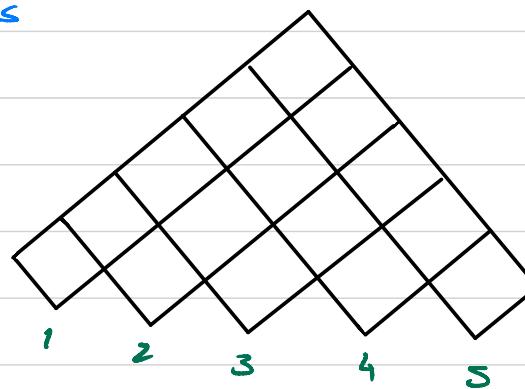
Multiplicação de Matrizes

$$C[i, j] = \begin{cases} \min_{i \leq k < j} \{ C[i, k] + C[k+1, j] + m_{i-1} \times m_k \times m_j \} & \text{se } j > i \\ 0 & \text{se } i = j \end{cases}$$

Multiplicação de Matrizes

$$C[i, j] = \begin{cases} \min_{i \leq k < j} \{ C[i, k] + C[k+1, j] + m_{i-1} \times m_k \times m_j \} & \text{se } j > i \\ 0 & \text{se } i = j \end{cases}$$

Multiplication cost (\vec{m}, n) // $\vec{m} [0, \dots, n]$ vector das dimensões



Multiplicação de Matrizes

$$C[i, j] = \begin{cases} \min_{i \leq k < j} \{ C[i, k] + C[k+1, j] + m_{i-1} \times m_k \times m_j \} & \text{se } j > i \\ 0 & \text{se } i = j \end{cases}$$

Multiplication cost (\vec{m}, n) // $\vec{m} [0, \dots, n]$ vector das dimensões

let $C[1, \dots, n; 1, \dots, n]$ be a new $n \times n$ matrix

for $i=1$ to n

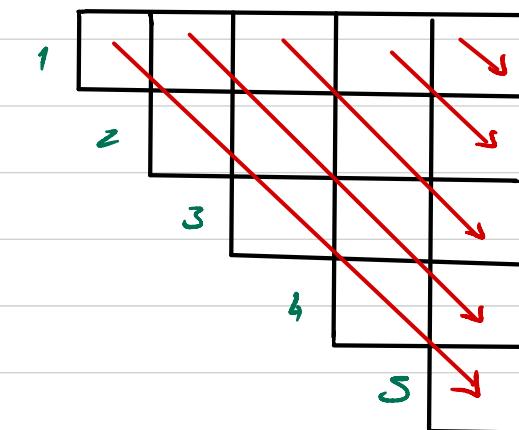
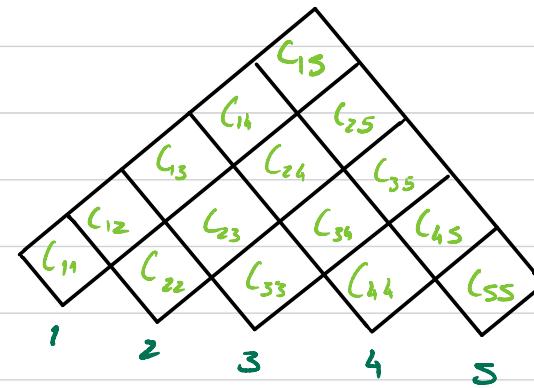
$C[i, i] := 0$

for $s=1$ to $n-1$

for $i=1$ to $n-s$

$$C[i, i+s] := \min \left\{ C[i, k] + C[k, i+s] + \vec{m}[i-1] \times \vec{m}[k] \times \vec{m}[i+s] \mid i \leq k \leq i+s \right\}$$

return C



Multiplicação de Matrizes

$$C[i, j] = \begin{cases} \min_{i \leq k < j} \{ C[i, k] + C[k+1, j] + m_{i-1} \times m_k \times m_j \} & \text{se } j > i \\ 0 & \text{se } i = j \end{cases}$$

Multiplication cost (\vec{m}, n) // $\vec{m} [0, \dots, n]$ vector das dimensões

let C be a new $n \times n$ matrix

for $i=1$ to n

$C[i, i] := 0$

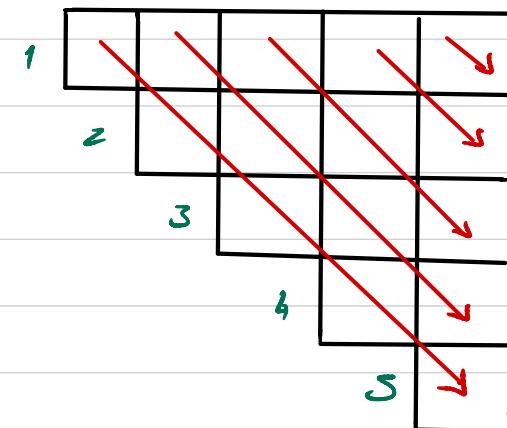
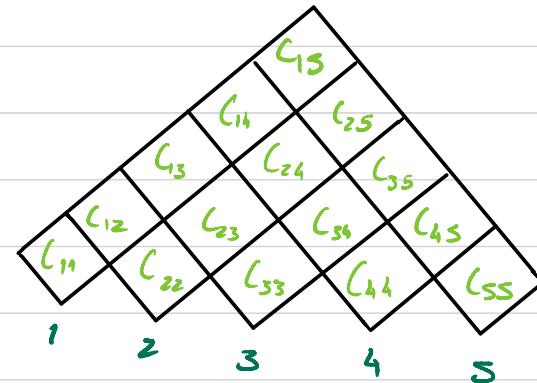
for $s=1$ to $n-1$

for $i=1$ to $n-s$

$C[i, i+s] := \min \{ C[i, k] + C[k, i+s] + \vec{m}[i-1] \times \vec{m}[k] \times \vec{m}[i+s] \mid i \leq k \leq i+s \}$

return C

Análise de Complexidade: $\Theta(n^3)$



Subseqüência Contígua de Soma Máxima

$$A = \langle 5, 15, -30, 10, -5, 40, 10 \rangle$$

- $S[i] \Rightarrow$

Subsequência Contígua de Soma Máxima

$$A = \langle 5, 15, -30, 10, -5, 40, 10 \rangle$$

- $S[i]$ ⇒ o valor da soma da subsequência contígua de soma máxima a terminar na posição i .

$$S[i] = \begin{cases} S[i-1] + A[i] & \text{se } S[i-1] \geq 0 \\ A[i] & \text{se } S[i-1] < 0 \\ 0 & \text{se } i=0 \end{cases}$$

$$\langle 5, 15, -30, 10, -5, 40, 10 \rangle$$

$$\begin{matrix} \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ S & 20 & -10 & 10 & 5 & 45 & 55 \end{matrix}$$