

# Mais programação dinâmica

## KT 6.4

Aproveite para olhar todo o Cap 6 do KT,  
que é sobre programação dinâmica.

= “recursão-com-tabela”

= transformação inteligente de recursão em iteração

## Consulta Discente 2024.2

Olá pessoal!

O ApoioBCC vem, através dos professores, informar que adiamos a consulta discente em uma semana, tornando **dia 07/10/2024** o prazo de preenchimento final do formulário e pedir, mais uma vez, que vocês respondam-na.

Estamos com uma baixíssima adesão ao formulário, o que pode significar que está tudo bem com o andamento das aulas, então, se não estiver, não deixe de nos informar a respeito.

Lembrando que o formulário é 100% anônimo.

Link (até 07/10/2024):

<https://forms.gle/wvAdeMswH9MWARGF6>

Obrigado, mais uma vez!

Att. ApoioBCC

# Mochila

Dados dois vetores  $x[1..n]$  e  $w[1..n]$ , denotamos por  $x \cdot w$  o **produto escalar**

$$w[1]x[1] + w[2]x[2] + \cdots + w[n]x[n].$$

Suponha dado um número inteiro não-negativo  $W$  e vetores positivos  $w[1..n]$  e  $v[1..n]$ .

Uma **mochila** é qualquer vetor  $x[1..n]$  tal que

$$x \cdot w \leq W \quad \text{e} \quad 0 \leq x[i] \leq 1 \quad \text{para todo } i$$

O **valor** de uma mochila é o número  $x \cdot v$ .

Uma mochila é **ótima** se tem valor máximo.

# Problema booleano da mochila

Uma mochila  $x[1..n]$  tal que  $x[i] = 0$  ou  $x[i] = 1$  para todo  $i$  é dita **booleana**.

**Problema (Knapsack Problem):**

Dados  $(w, v, n, W)$ , encontrar uma **mochila booleana ótima**.

Exemplo:  $W = 50, n = 4$

	1	2	3	4
$w$	40	30	20	10
$v$	840	600	400	100
$x$	1	0	0	0
$x$	1	0	0	1
$x$	0	1	1	0

valor = 840

valor = 940

valor = 1000

## Subestrutura ótima

Suponha que  $x[1..n]$  é  
mochila booleana ótima para o problema  $(w, v, n, W)$ .

Se  $x[n] = 1$

então

## Subestrutura ótima

Suponha que  $x[1..n]$  é  
mochila booleana ótima para o problema  $(w, v, n, W)$ .

Se  $x[n] = 1$

então  $x[1..n-1]$  é mochila booleana ótima  
para  $(w, v, n-1, W - w[n])$

senão

## Subestrutura ótima

Suponha que  $x[1..n]$  é  
mochila booleana ótima para o problema  $(w, v, n, W)$ .

Se  $x[n] = 1$

então  $x[1..n-1]$  é mochila booleana ótima  
para  $(w, v, n-1, W - w[n])$

senão  $x[1..n-1]$  é mochila booleana ótima  
para  $(w, v, n-1, W)$

## Subestrutura ótima

Suponha que  $x[1..n]$  é  
mochila booleana ótima para o problema  $(w, v, n, W)$ .

Se  $x[n] = 1$

então  $x[1..n-1]$  é mochila booleana ótima  
para  $(w, v, n-1, W - w[n])$

senão  $x[1..n-1]$  é mochila booleana ótima  
para  $(w, v, n-1, W)$

**NOTA.** Não há nada de especial acerca do índice  $n$ .  
Uma afirmação semelhante vale para qualquer índice  $i$ .

# Simplificação

**Problema:**

encontrar o **valor** de uma mochila booleana ótima.

# Simplificação

**Problema:**

encontrar o **valor** de uma mochila booleana ótima.

$t[i, Y]$  = valor de uma mochila booleana ótima para  $(w, v, i, Y)$

# Simplificação

## Problema:

encontrar o **valor** de uma mochila booleana ótima.

$t[i, Y]$  = valor de uma mochila booleana ótima para  $(w, v, i, Y)$

= valor da expressão  $x \cdot v$  sujeito às restrições

$$x \cdot w \leq Y,$$

onde  $x$  é uma mochila booleana ótima

# Simplificação

## Problema:

encontrar o **valor** de uma mochila booleana ótima.

$t[i, Y]$  = valor de uma mochila booleana ótima para  $(w, v, i, Y)$   
= valor da expressão  $x \cdot v$  sujeito às restrições

$$x \cdot w \leq Y,$$

onde  $x$  é uma mochila booleana ótima

Possíveis valores de  $Y$ :  $0, 1, 2, \dots, W$

# Recorrência

$t[i, Y]$  = valor da expressão  $x \cdot v$  sujeito à restrição

$$x \cdot w \leq Y$$

# Recorrência

$t[i, Y]$  = valor da expressão  $x \cdot v$  sujeito à restrição

$$x \cdot w \leq Y$$

$t[0, Y] = 0$  para todo  $Y$

$t[i, 0] = 0$  para todo  $i$

# Recorrência

$t[i, Y]$  = valor da expressão  $x \cdot v$  sujeito à restrição

$$x \cdot w \leq Y$$

$t[0, Y] = 0$  para todo  $Y$

$t[i, 0] = 0$  para todo  $i$

$t[i, Y] = t[i-1, Y]$  se  $w[i] > Y$

# Recorrência

$t[i, Y]$  = valor da expressão  $x \cdot v$  sujeito à restrição

$$x \cdot w \leq Y$$

$t[0, Y] = 0$  para todo  $Y$

$t[i, 0] = 0$  para todo  $i$

$t[i, Y] = t[i-1, Y]$  se  $w[i] > Y$

$t[i, Y] = \max \{ t[i-1, Y], t[i-1, Y-w[i]] + v[i] \}$  se  $w[i] \leq Y$

## Solução recursiva

Devolve o valor de uma mochila ótima para  $(w, v, n, W)$ .

```
REC-MOCHILA ( $w, v, n, W$ )
1  se  $n = 0$  ou  $W = 0$ 
2    então devolva 0
3  se  $w[n] > W$ 
4    então devolva REC-MOCHILA ( $w, v, n-1, W$ )
5   $a \leftarrow$  REC-MOCHILA ( $w, v, n-1, W$ )
6   $b \leftarrow$  REC-MOCHILA ( $w, v, n-1, W-w[n]$ ) +  $v[n]$ 
7  devolva  $\max\{a, b\}$ 
```

## Solução recursiva

Devolve o valor de uma mochila ótima para  $(w, v, n, W)$ .

```
REC-MOCHILA ( $w, v, n, W$ )
1  se  $n = 0$  ou  $W = 0$ 
2    então devolva 0
3  se  $w[n] > W$ 
4    então devolva REC-MOCHILA ( $w, v, n-1, W$ )
5   $a \leftarrow$  REC-MOCHILA ( $w, v, n-1, W$ )
6   $b \leftarrow$  REC-MOCHILA ( $w, v, n-1, W-w[n]$ ) +  $v[n]$ 
7  devolva  $\max\{a, b\}$ 
```

Consumo de tempo no pior caso é  $\Omega(2^n)$ .

## Solução recursiva

Devolve o valor de uma mochila ótima para  $(w, v, n, W)$ .

```
REC-MOCHILA ( $w, v, n, W$ )
1  se  $n = 0$  ou  $W = 0$ 
2    então devolva 0
3  se  $w[n] > W$ 
4    então devolva REC-MOCHILA ( $w, v, n-1, W$ )
5   $a \leftarrow$  REC-MOCHILA ( $w, v, n-1, W$ )
6   $b \leftarrow$  REC-MOCHILA ( $w, v, n-1, W - w[n]$ ) +  $v[n]$ 
7  devolva  $\max\{a, b\}$ 
```

Consumo de tempo no pior caso é  $\Omega(2^n)$ .

Por que demora tanto?

O mesmo subproblema é resolvido muitas vezes.

# Programação dinâmica

Cada subproblema, valor de uma mochila ótima para

$$(w, v, i, Y),$$

é resolvido **uma só** vez.

Em que ordem calcular os componentes da tabela  $t$ ?

# Programação dinâmica

Cada subproblema, valor de uma mochila ótima para

$$(w, v, i, Y),$$

é resolvido **uma só** vez.

Em que ordem calcular os componentes da tabela  $t$ ?

Olhe a recorrência e pense...

$$t[i, Y] = t[i-1, Y] \text{ se } w[i] > Y$$

$$t[i, Y] = \max \{ t[i-1, Y], t[i-1, Y-w[i]] + v[i] \} \text{ se } w[i] \leq Y$$

# Programação dinâmica

	0	1	2	3	4	5	6	7	Y
0	0	0	0	0	0	0	0	0	
1	0								
2	0	*	*	*	*	*			
3	0					??			
4	0								
5	0								
6	0								
7	0								

*i*

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
$w$	4	2	1	3
$v$	500	400	300	450

	0	1	2	3	4	5	$Y$
0	0	0	0	0	0	0	
1	0						
2	0						
3	0						
4	0						
$i$							

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
$w$	4	2	1	3
$v$	500	400	300	450

	0	1	2	3	4	5	$Y$
0	0	0	0	0	0	0	
1	0	0					
2	0						
3	0						
4	0						
$i$							

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
$w$	4	2	1	3
$v$	500	400	300	450

	0	1	2	3	4	5	$Y$
0	0	0	0	0	0	0	
1	0	0	0				
2	0						
3	0						
4	0						
$i$							

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
$w$	4	2	1	3
$v$	500	400	300	450

	0	1	2	3	4	5	$Y$
0	0	0	0	0	0	0	
1	0	0	0	0			
2	0						
3	0						
4	0						
$i$							

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
w	4	2	1	3
v	500	400	300	450

	0	1	2	3	4	5	Y
0	0	0	0	0	0	0	
1	0	0	0	0	500		
2	0						
3	0						
4	0						
i							

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
$w$	4	2	1	3
$v$	500	400	300	450

	0	1	2	3	4	5	$Y$
0	0	0	0	0	0	0	
1	0	0	0	0	500	500	
2	0						
3	0						
4	0						
$i$							

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
w	4	2	1	3
v	500	400	300	450

	0	1	2	3	4	5	Y
0	0	0	0	0	0	0	
1	0	0	0	0	500	500	
2	0	0					
3	0						
4	0						
i							

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
$w$	4	2	1	3
$v$	500	400	300	450

	0	1	2	3	4	5	$Y$
0	0	0	0	0	0	0	
1	0	0	0	0	500	500	
2	0	0	400				
3	0						
4	0						
$i$							

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
w	4	2	1	3
v	500	400	300	450

	0	1	2	3	4	5	Y
0	0	0	0	0	0	0	
1	0	0	0	0	500	500	
2	0	0	400	400			
3	0						
4	0						
i							

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
$w$	4	2	1	3
$v$	500	400	300	450

	0	1	2	3	4	5	$Y$
0	0	0	0	0	0	0	
1	0	0	0	0	500	500	
2	0	0	400	400	500		
3	0						
4	0						
$i$							

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
$w$	4	2	1	3
$v$	500	400	300	450

	0	1	2	3	4	5	$Y$
0	0	0	0	0	0	0	
1	0	0	0	0	500	500	
2	0	0	400	400	500	500	
3	0						
4	0						
$i$							

# Exemplo

$$W = 5 \text{ e } n = 4$$

	1	2	3	4
$w$	4	2	1	3
$v$	500	400	300	450

	0	1	2	3	4	5	$Y$
0	0	0	0	0	0	0	
1	0	0	0	0	500	500	
2	0	0	400	400	500	500	
3	0	300	400	700	700	800	
4	0	300	400	700	750	850	
$i$							

# Algoritmo de programação dinâmica

Devolve o valor de uma mochila booleana ótima para  $(w, v, n, W)$ .

**MOCHILA-BOOLEANA**  $(w, v, n, W)$

```
1  para  $Y \leftarrow 0$  até  $W$  faça
2     $t[0, Y] \leftarrow 0$ 
3    para  $i \leftarrow 1$  até  $n$  faça
4       $a \leftarrow t[i-1, Y]$ 
5      se  $w[i] > Y$ 
6        então  $b \leftarrow 0$ 
7        senão  $b \leftarrow t[i-1, Y-w[i]] + v[i]$ 
8       $t[i, Y] \leftarrow \max\{a, b\}$ 
9  devolva  $t[n, W]$ 
```

# Algoritmo de programação dinâmica

Devolve o valor de uma mochila booleana ótima para  $(w, v, n, W)$ .

**MOCHILA-BOOLEANA**  $(w, v, n, W)$

```
1  para  $Y \leftarrow 0$  até  $W$  faça
2     $t[0, Y] \leftarrow 0$ 
3    para  $i \leftarrow 1$  até  $n$  faça
4       $a \leftarrow t[i-1, Y]$ 
5      se  $w[i] > Y$ 
6        então  $b \leftarrow 0$ 
7        senão  $b \leftarrow t[i-1, Y-w[i]] + v[i]$ 
8       $t[i, Y] \leftarrow \max\{a, b\}$ 
9  devolva  $t[n, W]$ 
```

Consumo de tempo é  $\Theta(nW)$ .

## Conclusão

O consumo de tempo do algoritmo  
MOCHILA-BOOLEANA é  $\Theta(nW)$ .

## Conclusão

O consumo de tempo do algoritmo  
MOCHILA-BOOLEANA é  $\Theta(nW)$ .

NOTA:

O consumo  $\Theta(n2^{\lg W})$  é exponencial!

# Conclusão

O consumo de tempo do algoritmo  
MOCHILA-BOOLEANA é  $\Theta(nW)$ .

## NOTA:

O consumo  $\Theta(n2^{\lg W})$  é exponencial!

Explicação: o “tamanho” de  $W$  é  $\lg W$  e não  $W$   
(tente multiplicar  $w[1], \dots, w[n]$  e  $W$  por 1000)

# Conclusão

O consumo de tempo do algoritmo  
MOCHILA-BOOLEANA é  $\Theta(nW)$ .

## NOTA:

O consumo  $\Theta(n2^{\lg W})$  é exponencial!

Explicação: o “tamanho” de  $W$  é  $\lg W$  e não  $W$   
(tente multiplicar  $w[1], \dots, w[n]$  e  $W$  por 1000)

Se  $W$  é  $\Omega(2^n)$  o consumo de tempo é  $\Omega(n2^n)$ ,  
mais lento que o algoritmo **força bruta**!

## Obtenção da mochila

MOCHILA ( $w, n, W, t$ )

```
1   $Y \leftarrow W$ 
2  para  $i \leftarrow n$  decrescendo até 1 faça
3      se  $t[i, Y] = t[i-1, Y]$ 
4          então  $x[i] \leftarrow 0$ 
5          senão  $x[i] \leftarrow 1$ 
6               $Y \leftarrow Y - w[i]$ 
7  devolva  $x$ 
```

## Obtenção da mochila

```
MOCHILA ( $w, n, W, t$ )
1   $Y \leftarrow W$ 
2  para  $i \leftarrow n$  decrescendo até 1 faça
3      se  $t[i, Y] = t[i-1, Y]$ 
4          então  $x[i] \leftarrow 0$ 
5          senão  $x[i] \leftarrow 1$ 
6               $Y \leftarrow Y - w[i]$ 
7  devolva  $x$ 
```

Consumo de tempo é  $\Theta(n)$ .

## Versão recursiva

MEMOIZED-MOCHILA-BOOLEANA ( $w, v, n, W$ )

1 para  $i \leftarrow 0$  até  $n$  faça

2     para  $Y \leftarrow 0$  até  $W$  faça

3          $t[i, Y] \leftarrow \infty$

4 devolva LOOKUP-MOC ( $w, v, n, W$ )

## Versão recursiva

MEMOIZED-MOCHILA-BOOLEANA ( $w, v, n, W$ )

- 1 para  $i \leftarrow 0$  até  $n$  faça
- 2     para  $Y \leftarrow 0$  até  $W$  faça
- 3          $t[i, Y] \leftarrow \infty$
- 4 devolva LOOKUP-MOC ( $w, v, n, W$ )

LOOKUP-MOC ( $w, v, i, Y$ )

- 1 se  $t[i, Y] < \infty$
- 2     então devolva  $t[i, Y]$
- 3 se  $i = 0$  ou  $Y = 0$  então  $t[i, Y] \leftarrow 0$

## Versão recursiva

MEMOIZED-MOCHILA-BOOLEANA ( $w, v, n, W$ )

- 1 para  $i \leftarrow 0$  até  $n$  faça
- 2     para  $Y \leftarrow 0$  até  $W$  faça
- 3          $t[i, Y] \leftarrow \infty$
- 4 devolva LOOKUP-MOC ( $w, v, n, W$ )

LOOKUP-MOC ( $w, v, i, Y$ )

- 1 se  $t[i, Y] < \infty$
- 2     então devolva  $t[i, Y]$
- 3 se  $i = 0$  ou  $Y = 0$  então  $t[i, Y] \leftarrow 0$
- 4 senão se  $w[i] > Y$
- 5     então  $t[i, Y] \leftarrow$  LOOKUP-MOC ( $w, v, i-1, Y$ )
- 6     senão  $a \leftarrow$  LOOKUP-MOC ( $w, v, i-1, Y$ )
- 7          $b \leftarrow$  LOOKUP-MOC ( $w, v, i-1, Y - w[i]$ ) +  $v[i]$
- 8          $t[i, Y] \leftarrow \max\{a, b\}$
- 9 devolva  $t[i, Y]$

# Problema dos caramelos

Problema K em:

<https://maratona.sbc.org.br/primfase24/contest-pf2024/maratona.pdf>

Primeiramente temos que mastigar um pouco esse problema...

# Problema dos caramelos

Problema K em:

<https://maratona.sbc.org.br/primfase24/contest-pf2024/maratona.pdf>

Primeiramente temos que mastigar um pouco esse problema...

Suponha que temos **uma partição das sacolas de caramelos que deixa exatamente metade dos caramelos para cada um** deles.  
(Se isso não existir, não há nada a fazer.)

Será que, neste caso, **existe uma ordem** que deixe cada um dos dois com exatamente um dos conjuntos dessa partição?

# Problema dos caramelos

Problema K em:

<https://maratona.sbc.org.br/primfase24/contest-pf2024/maratona.pdf>

Primeiramente temos que mastigar um pouco esse problema...

Suponha que temos **uma partição das sacolas de caramelos que deixa exatamente metade dos caramelos para cada um** deles.  
(Se isso não existir, não há nada a fazer.)

Será que, neste caso, **existe uma ordem** que deixe cada um dos dois com exatamente um dos conjuntos dessa partição?

Agora... será que conseguimos decidir se **existe uma tal partição?**  
Como podemos usar o **problema da mochila** para isso?