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

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

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

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 boolena 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	valor $= 840$
x	1	0	0	1	valor $= 940$
x	0	1	1	0	valor $= 1000$

Subestrutura ótima

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

Se $x[n] = 1$

então

Subestrutura ótima

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

Se $x[n] = 1$

então $x[1\,..\,n{-}1]$ é mochila boolena ótima
para $(w, v, n-1, W - w[n])$

senão

Subestrutura ótima

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

Se $x[n] = 1$

então $x[1\,..\,n{-}1]$ é mochila boolena ótima
para $(w, v, n-1, W - w[n])$

senão $x[1\,..\,n{-}1]$ é mochila boolena ótima
para $(w, v, n-1, W)$

Subestrutura ótima

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

Se $x[n] = 1$

então $x[1\,..\,n{-}1]$ é mochila boolena ótima
para $(w, v, n - 1, W - w[n])$

senão $x[1\,..\,n{-}1]$ é mochila boolena 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.

$$
\begin{aligned}
t[i, Y] &= \text{ valor de uma mochila booleana ótima para } (w, v, i, Y) \\
&= \text{ valor da expressão } x \cdot v \text{ sujeito às restrições}
\end{aligned}
$$

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

onde x é uma mochila booleana ótima

Simplificação

Problema:
encontrar o valor de uma mochila booleana ótima.

$$\begin{aligned} t[i, Y] &= \text{ valor de uma mochila booleana ótima para } (w, v, i, Y) \\ &= \text{ valor da expressão } x \cdot v \text{ sujeito às restrições} \end{aligned}$$

$$x \cdot w \le Y,$$

onde x é uma mochila booleana ótima

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

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)
se $n = 0$ **ou** $W = 0$
  então devolva 0
se $w[n] > W$
  então devolva REC-MOCHILA $(w, v, n-1, W)$
$a \leftarrow$ REC-MOCHILA $(w, v, n-1, W)$
$b \leftarrow$ REC-MOCHILA $(w, v, n-1, W-w[n]) + v[n]$
devolva $\max\{a, b\}$

Solução recursiva

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

REC-MOCHILA (w, v, n, W)
se $n = 0$ **ou** $W = 0$
  então devolva 0
se $w[n] > W$
  então devolva REC-MOCHILA $(w, v, n-1, W)$
$a \leftarrow$ REC-MOCHILA $(w, v, n-1, W)$
$b \leftarrow$ REC-MOCHILA $(w, v, n-1, W-w[n]) + v[n]$
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)
se $n = 0$ **ou** $W = 0$
  então devolva 0
se $w[n] > W$
  então devolva REC-MOCHILA $(w, v, n-1, W)$
$a \leftarrow$ REC-MOCHILA $(w, v, n-1, W)$
$b \leftarrow$ REC-MOCHILA $(w, v, n-1, W - w[n]) + v[n]$
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

$$(w, v, {\color{blue}i}, {\color{red}Y}),$$

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...

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

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

Programação dinâmica

	0	1	2	3	4	5	6	7	Y
0	0	0	0	0	0	0	0	0	
1	0								
2	0	$\star$	$\star$	$\star$	$\star$	$\star$			
3	0					??			
4	0								
5	0								
6	0								
7	0								
i									

Exemplo

$W = 5$ 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

$W = 5$ e $n = 4$

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

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

Exemplo

$W = 5$ 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

$W = 5$ e $n = 4$

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

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

Exemplo

$W = 5$ e $n = 4$

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

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

Exemplo

$W = 5$ 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

$W = 5$ 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

$W = 5$ 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

$W = 5$ 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

$W = 5$ 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

$W = 5$ 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

$W = 5$ 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)
para Y ← 0 até W faça
    t[0, Y] ← 0
    para i ← 1 até n faça
        a ← t[i−1, Y]
        se w[i] > Y
            então b ← 0
            senão b ← t[i−1, Y−w[i]] + v[i]
        t[i, Y] ← max{a, b}
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)
para Y ← 0 até W faça
    t[0, Y] ← 0
    para i ← 1 até n faça
        a ← t[i−1, Y]
        se w[i] > Y
            então b ← 0
            senão b ← t[i−1, Y−w[i]] + v[i]
        t[i, Y] ← max{a, b}
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 $w[1], \ldots, w[n]$ 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 $w[1], \ldots, w[n]$ 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)
Y ← W
para i ← n decrescendo até 1 faça
    se t[i, Y] = t[i−1, Y]
        então x[i] ← 0
        senão x[i] ← 1
                Y ← Y − w[i]
devolva x
```

Obtenção da mochila

```
MOCHILA (w, n, W, t)
Y ← W
para i ← n decrescendo até 1 faça
    se t[i, Y] = t[i−1, Y]
        então x[i] ← 0
        senão x[i] ← 1
              Y ← Y − w[i]
devolva x
```

Consumo de tempo é $\Theta(n)$.

Versão recursiva

MEMOIZED-MOCHILA-BOOLEANA (w, v, n, W)
para $i \leftarrow 0$ **até** n **faça**
  para $Y \leftarrow 0$ **até** W **faça**
    $t[i, Y] \leftarrow \infty$
devolva LOOKUP-MOC (w, v, n, W)

Versão recursiva

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

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

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

se $t[i, Y] < \infty$
    então devolva $t[i, Y]$
se $i = 0$ **ou** $Y = 0$ **então** $t[i, Y] \leftarrow 0$

Versão recursiva

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

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

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

se $t[i, Y] < \infty$
    então devolva $t[i, Y]$
se $i = 0$ **ou** $Y = 0$ **então** $t[i, Y] \leftarrow 0$
senão se $w[i] > Y$
        então $t[i, Y] \leftarrow$ LOOKUP-MOC $(w, v, i-1, Y)$
        senão $a \leftarrow$ LOOKUP-MOC $(w, v, i-1, Y)$
            $b \leftarrow$ LOOKUP-MOC $(w, v, i-1, Y-w[i]) + v[i]$
            $t[i, Y] \leftarrow \max\{a, b\}$
devolva $t[i, Y]$

Problema fracionário da mochila

Problema: Dados (w, v, n, W), encontrar uma mochila ótima.

Problema fracionário da mochila

Problema: Dados (w, v, n, W), encontrar uma mochila 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	valor $= 840$
x	1	0	0	1	valor $= 940$
x	0	1	1	0	valor $= 1000$
x	1	$1/3$	0	0	valor $= 1040$

Subestrutura ótima

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

Subestrutura ótima

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

Se $x[n] = \delta$

então $x[1\,..\,n{-}1]$ é mochila ótima para

$$(w, v, n-1, W - \delta\, w[n])$$

Subestrutura ótima

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

Se $x[n] = \delta$

então $x[1\,..\,n{-}1]$ é mochila ótima para

$$(w, v, n-1, W - \delta\, w[n])$$

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

Subestrutura ótima

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

Se $x[n] = \delta$

então $x[1\,..\,n{-}1]$ é mochila ótima para

$$(w, v, n-1, W - \delta\, w[n])$$

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

Então podemos fazer uma algoritmo de PD como no caso anterior.
Mas não dá para fazer algo mais simples, melhor?

Escolha gulosa

Suponha $w[i] \neq 0$ para todo i.

Escolha gulosa

Suponha $w[i] \neq 0$ para todo i.

Se $v[n]/w[n] \geq v[i]/w[i]$ para todo i

então EXISTE uma mochila ótima $x[1\,..\,n]$ tal que

$$x[n] = \min\left\{1, \frac{W}{w[n]}\right\}$$