Corte de hastes

Hastes de aço são vendidas em pedaços de tamanho inteiro.
As usinas produzem hastes longas,
e os comerciantes cortam em pedaços para vender.

Suponha que o preço de uma haste de tamanho i
esteja tabelado como p_i.

Corte de hastes

Hastes de aço são vendidas em pedaços de tamanho inteiro.
As usinas produzem hastes longas,
e os comerciantes cortam em pedaços para vender.

Suponha que o preço de uma haste de tamanho i
esteja tabelado como p_i.

Problema: Dada uma haste de tamanho n e a tabela p de preços, qual a melhor forma de cortar a haste para maximizar o preço de venda total?

Corte de hastes

Hastes de aço são vendidas em pedaços de tamanho inteiro.
As usinas produzem hastes longas,
e os comerciantes cortam em pedaços para vender.

Suponha que o preço de uma haste de tamanho i
esteja tabelado como p_i.

Problema: Dada uma haste de tamanho n e a tabela p de preços, qual a melhor forma de cortar a haste para maximizar o preço de venda total?

Versão simplificada: qual o maior valor q_n
que se pode obter de uma haste de tamanho n?

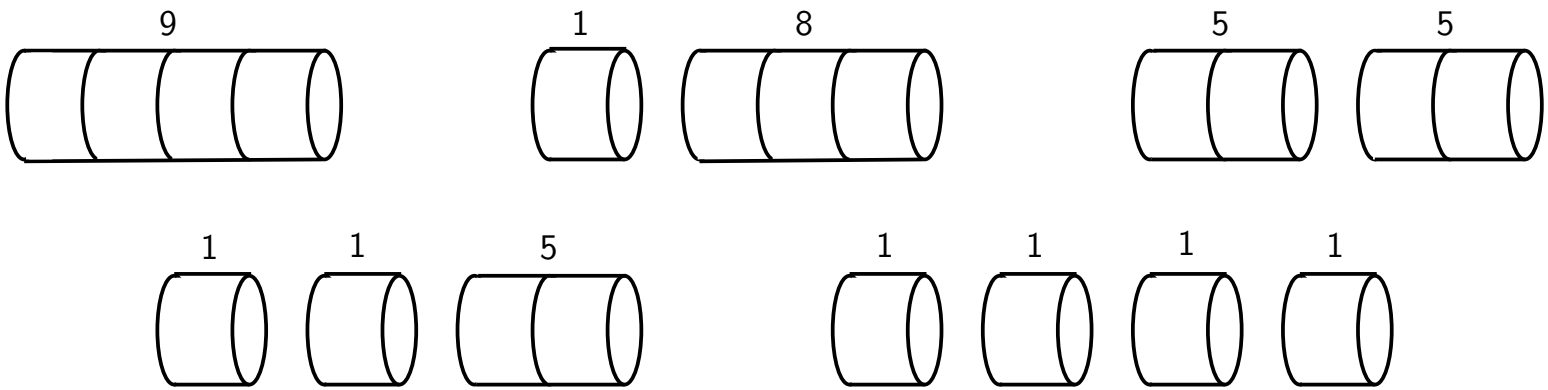
Exemplo

n	1	2	3	4	5	6	7	8	9
p_n	1	5	8	9	10	17	17	20	24

Exemplo

n	1	2	3	4	5	6	7	8	9
p_n	1	5	8	9	10	17	17	20	24

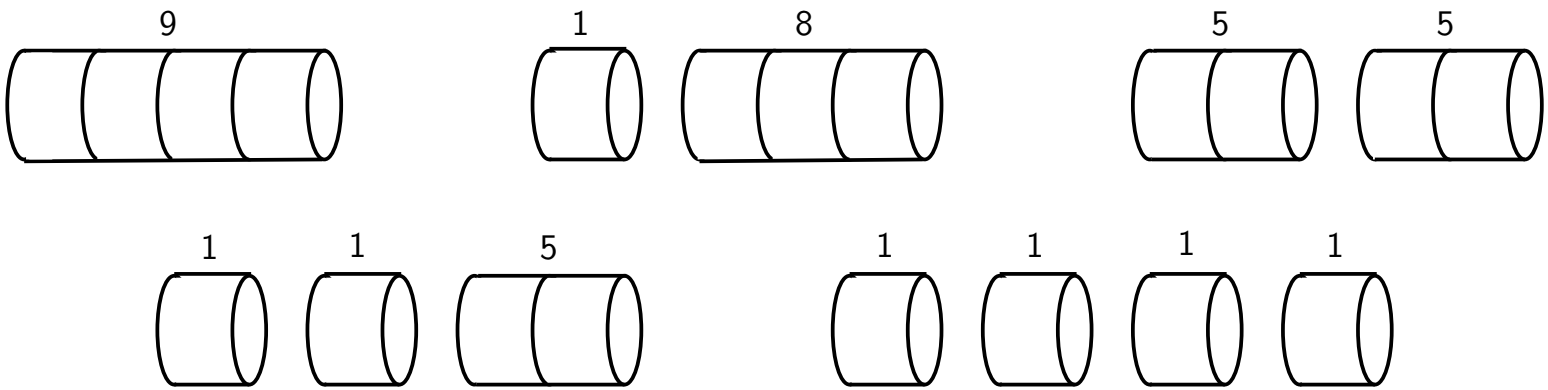
Possíveis cortes para $n = 4$:



Exemplo

n	1	2	3	4	5	6	7	8	9
p_n	1	5	8	9	10	17	17	20	24

Possíveis cortes para $n = 4$:



Melhor corte (de maior lucro): o terceiro, com valor 10.

Solução recursiva

Corta-se um primeiro pedaço de tamanho i e o pedaço restante, de tamanho $n - i$, do melhor jeito possível. O valor desse corte é

$$p_i + q_{n-i}.$$

Solução recursiva

Corta-se um primeiro pedaço de tamanho i e o pedaço restante, de tamanho $n - i$, do melhor jeito possível. O valor desse corte é

$$p_i + q_{n-i}.$$

A questão é escolher o melhor i; o que maximiza a expressão acima:

$$q_n = \max_{1 \leq i \leq n} \{p_i + q_{n-i}\}.$$

$$q_0 = 0.$$

Primeiro código

```
CORTA-HASTE (p, n)
1  se n = 0
2      então devolva 0
3  q ← −∞
4  para i ← 1 até n
5      q ← max{q, p[i] + CORTA-HASTE(p, n − i)}
6  devolva q
```

Primeiro código

```
CORTA-HASTE (p, n)
1  se n = 0
2      então devolva 0
3  q ← −∞
4  para i ← 1 até n
5      q ← max{q, p[i] + CORTA-HASTE(p, n − i)}
6  devolva q
```

Consumo de tempo:

$$T(n) \;=\; 1 + \sum_{i=0}^{n-1} T(i)$$

Primeiro código

```
CORTA-HASTE (p, n)
1  se n = 0
2      então devolva 0
3  q ← −∞
4  para i ← 1 até n
5      q ← max{q, p[i] + CORTA-HASTE(p, n − i)}
6  devolva q
```

Consumo de tempo:

$$T(n) \;=\; 1 + \sum_{i=0}^{n-1} T(i) \;=\; 2^n.$$

Com memoização

Note que r funciona como variável global.

CORTA-HASTE-MEMOIZADO (p, n)
1 $r[0] \leftarrow 0$
2 **para** $i \leftarrow 1$ **até** n
3 $\quad r[i] \leftarrow -\infty$
4 **devolva** CORTA-HASTE-MEMOIZADO-REC (p, n, r)

Com memoização

Note que r funciona como variável global.

```
CORTA-HASTE-MEMOIZADO (p, n)
1  r[0] ← 0
2  para i ← 1 até n
3      r[i] ← −∞
4  devolva CORTA-HASTE-MEMOIZADO-REC (p, n, r)

CORTA-HASTE-MEMOIZADO-REC (p, n, r)
1  se r[n] ≥ 0
2      devolva r[n]
3  senão q ← −∞
4      para i ← 1 até n
5          q ← max{q, p[i] + Corta-Haste-Memoizado-Rec(p, n − i, r)}
6      r[n] ← q
7      devolva q
```

Bottom up

```
CORTA-HASTE-BOTTOM-UP (p, n)
1  r[0] ← 0
2  para j ← 1 até n
3      q ← −∞
4      para i ← 1 até j
5          q ← max{q, p[i] + r[j − i]}
6      r[j] ← q
7  devolva q
```

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0				

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1			

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	2		

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5		

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5	6	

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5	6	

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5	8	

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5	8	9

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5	8	10

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5	8	10

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5	8	10

Recuperando o melhor corte

```
CORTA-HASTE-BOTTOM-UP-COMPLETO (p, n)
1  r[0] ← 0
2  para j ← 1 até n
3      q ← −∞
4      para i ← 1 até j
5          se q < p[i] + r[j − i]
6              q ← p[i] + r[j − i]
7              d[j] ← i
8      r[j] ← q
9  devolva q e d
```

Exemplo

n	1	2	3	4
p_n	1	5	8	9

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1			
d_n		1			

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	2		
d_n		1	1		

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5		
d_n		1	2		

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5	6	
d_n		1	2	1	

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5	8	
d_n		1	2	3	

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5	8	9
d_n		1	2	3	1

Exemplo

n	1	2	3	4
p_n	1	5	8	9

n	0	1	2	3	4
r_n	0	1	5	8	10
d_n		1	2	3	2

Listando os cortes

(usando concatenação de listas, estilo python)

```
LISTA-CORTES(d, n)
1  se n = 0 ou d[n] = n
2      devolva [ ]    ▷ lista vazia
3  senão
4      devolva [ d[n] ] .LISTA-CORTES(d, n − d[n])
```

Listando os cortes

(usando concatenação de listas, estilo python)

```
LISTA-CORTES(d, n)
1  se n = 0 ou d[n] = n
2      devolva [ ]    ▷ lista vazia
3  senão
4      devolva [ d[n] ] .LISTA-CORTES(d, n − d[n])
```

Consumo de tempo: $O(n)$