# Programação Dinâmica: Múltiplas escolhas

Prof. Daniel Kikuti

Universidade Estadual de Maringá

# Corte de hastes (Cutting Rods)

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. **Introduction to Algorithms**, Third Edition. The MIT Press. Chapter 15.

# Corte de hastes

## O Problema

Uma empresa compra hastes longas, corta-as em pedaços menores e as revende.

Uma **instância do problema** contém:

- tamanho $n$ da haste;
- tabela com preços de venda.

**Objetivo:** Maximizar o lucro.

Assuma que:

- é possível usar qualquer número de cortes (de $0$ a $n-1$);
- não há um custo para se efetuar um corte.

# Exemplo

## Considere a instância:

Suponha $n = 4$ e a seguinte tabela de preços:

| length $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| price $p_i$ | 1 | 5 | 8 | 9 | 10 | 17 | 17 | 20 | 24 | 30 |

## Maneiras de se cortar a haste

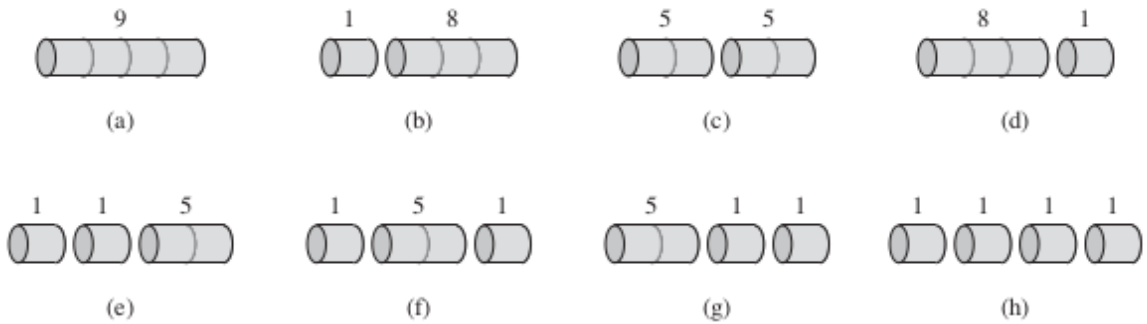



**Figure 15.2** The 8 possible ways of cutting up a rod of length 4. Above each piece is the value of that piece, according to the sample price chart of Figure 15.1. The optimal strategy is part (c)—cutting the rod into two pieces of length 2—which has total value 10.

# Definindo uma solução recursiva para o problema

## Uma abordagem inicial

Seja $r_n$ o máximo que podemos obter de lucro com uma haste de tamanho $n$, podemos definir o valor de $r_n$ em termos de receitas ótimas advindas de hastes mais curtas:

$$r_n = \max\left(p_n, r_1 + r_{n-1}, r_2 + r_{n-2}, \ldots, r_{n-1} + r_1\right).$$

# Definindo uma solução recursiva para o problema

## Uma abordagem inicial

Seja $r_n$ o máximo que podemos obter de lucro com uma haste de tamanho $n$, podemos definir o valor de $r_n$ em termos de receitas ótimas advindas de hastes mais curtas:

$$r_n = \max\left(p_n, r_1 + r_{n-1}, r_2 + r_{n-2}, \ldots, r_{n-1} + r_1\right).$$

## Outra alternativa (mais simples)

Fixar o corte da primeira peça (com tamanho $i$) e encontrar recursivamente a solução ótima para o restante da haste $(n - i)$.

$$r_n = \max_{1 \leq i \leq n} (p_i + r_{n-i}).$$

Consideramos que $r_0 = 0$.

# Algoritmo força bruta recursivo

```
CUT-ROD(p, n)
1 if n = 0 then
2 |   q ← 0
3 else
4 |   q ← −∞
5 |   for i ← 1 to n do
6 |   |   q ← max(q, p[i] + CUT-ROD(p, n − i))
7 return q
```

# Algoritmo força bruta recursivo

```
CUT-ROD(p, n)
1 if n = 0 then
2 |   q ← 0
3 else
4 |   q ← −∞
5 |   for i ← 1 to n do
6 |   |   q ← max(q, p[i] + CUT-ROD(p, n − i))
7 return q
```

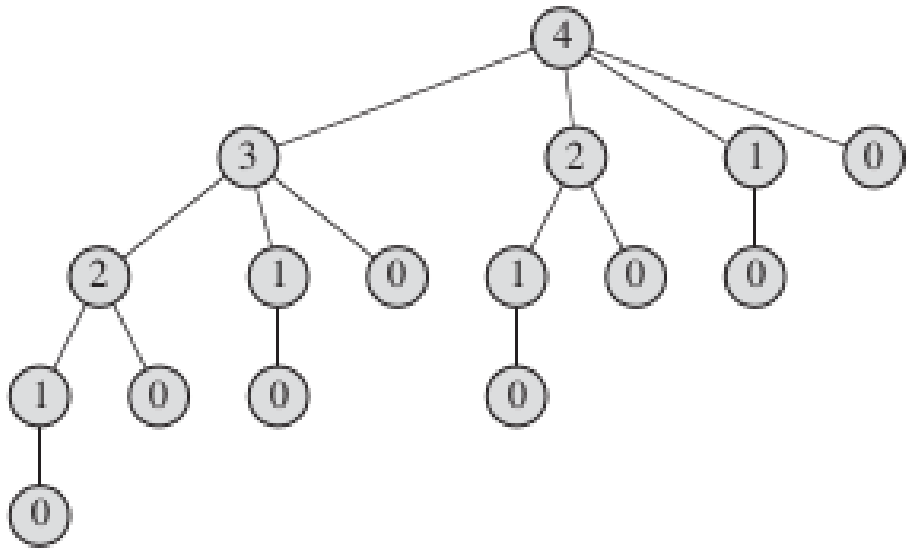
## Complexidade?

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

Exercício 15.1-1 do Cormen pede para você demonstrar isto.

# Sobreposição de problemas

A Figura a seguir ilustra sobreposição de problemas.



# Exercícios

1. Apresente um algoritmo memoizado.
2. Apresente um algoritmo iterativo (*bottom-up*).
3. Escreva um algoritmo para recuperar quais os cortes fazem parte da solução ótima.

# Multiplicação de cadeia de matrizes

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. **Introduction to Algorithms**, Third Edition. The MIT Press. Chapter 15.

# Multiplicação de cadeia de matrizes

## O Problema

Dada uma cadeia $\langle A_1, A_2, \ldots, A_n \rangle$ de $n$ matrizes, onde para cada $i = 1, \ldots, n$ a matriz $A_i$ tem dimensão $p_{i-1} \times p_i$, expresse de modo totalmente parentizado o produto $A_1, A_2, \ldots, A_n$, de maneira a minimizar o número de multiplicações escalares.

Uma cadeia é totalmente parentizada se:

- é uma única matriz;
- é um produto de duas cadeias de matrizes totalmente parentizadas entre parênteses.

## Lembrete sobre multiplicação de matrizes

- Associativa: $(A \cdot B) \cdot C = A \cdot (B \cdot C)$;
- Não comutativa: $A \cdot B \not\Rightarrow B \cdot A$;
- $A_{p \times q} \cdot B_{q \times r} = C_{p \times r}$.

# Número de multiplicações escalares e parentização

## Exemplo

No quadro.

## Número de parentizações distintas

Para a cadeia $\langle A_1, A_2, A_3, A_4 \rangle$, podemos expressar o produto de 5 maneiras distintas:

- $(A_1(A_2(A_3A_4)))$
- $(A_1((A_2A_3)A_4))$
- $((A_1A_2)(A_3A_4))$
- $((A_1(A_2A_3))A_4)$
- $(((A_1A_2)A_3)A_4)$

# Número de parentizações distintas

Seja $P(n)$ o número de parentizações para uma cadeia de $n$ matrizes:

$$P(n) = \begin{cases} 1 & \text{se } n = 1, \\ \sum_{k=1}^{n-1} P(k) \cdot P(n-k) & \text{se } n \geq 2. \end{cases}$$

## Exercício

Mostrar por substituição que $P(n) = \Omega(2^n)$.

# Subestrutura ótima

## Notação

- Usaremos a notação $A_{i..j}$, $i \leq j$, para denotar $A_i A_{i+j} \cdots A_j$.
- Se o problema é não trivial ($i < j$), então a parentização de $A_i A_{i+1} \cdots A_j$ deve dividir o produto entre $A_k$ e $A_{k+1}$ para algum inteiro $i \leq k < j$, ou seja:

$$(A_i \cdots A_k) \cdot (A_{k+1} \cdots A_j).$$

- O custo total desta parentização será a soma dos custos para:
  - computar a matriz $A_{i..k}$,
  - computar a matriz $A_{k+1..j}$,
  - multiplicá-las uma pela outra.

## Importante

A parentização da subcadeia $A_{i..k}$ dentro da parentização ótima de $A_{i..j}$ deve ser ótima.

# Definição recursiva

- Seja $m(i, j)$ o número mínimo de multiplicações escalares necessárias para computar $A_{i..j}$,
- $p = \langle p_0, p_1, \ldots, p_n \rangle$ as dimensões das matrizes. Ex: $p = \langle 10, 100, 5, 50 \rangle$ expressa $A_{10\times100} \cdot B_{100\times5} \cdot C_{5\times50}$.

$$m(i,j) = \begin{cases} 0 & \text{se } i = j, \\ \min_{i \le k < j}\{m(i,k) + m(k+1,j) + p_{i-1}p_k p_j\} & \text{se } i < j. \end{cases}$$