Programação Dinâmica - Parte 2

Projeto e Análise de Algoritmos II

Antonio Luiz Basile

Faculdade de Computação e Informática
Universidade Presbiteriana Mackenzie

August 20, 2018

Exemplo 2: Problema do corte da haste de aço

Definition

O **problema do corte da haste** é descrito como segue. Dada uma haste de tamanho n e uma tabela de preços p_i para $i = 1, 2, \ldots, n$, determine a receita máxima r_n obtida pelo corte de uma haste de aço e pela subsequente venda de suas partes.

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

Figure: Exemplo de tabela de preços para hastes

Corte da Haste: entendendo o problema

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

$$
\begin{aligned}
r_1 &= 1 \quad \text{from solution } 1 = 1 \quad \text{(no cuts)}\,, \\
r_2 &= 5 \quad \text{from solution } 2 = 2 \quad \text{(no cuts)}\,, \\
r_3 &= 8 \quad \text{from solution } 3 = 3 \quad \text{(no cuts)}\,, \\
r_4 &= 10 \quad \text{from solution } 4 = 2 + 2\,, \\
r_5 &= 13 \quad \text{from solution } 5 = 2 + 3\,, \\
r_6 &= 17 \quad \text{from solution } 6 = 6 \quad \text{(no cuts)}\,, \\
r_7 &= 18 \quad \text{from solution } 7 = 1 + 6 \text{ or } 7 = 2 + 2 + 3\,, \\
r_8 &= 22 \quad \text{from solution } 8 = 2 + 6\,, \\
r_9 &= 25 \quad \text{from solution } 9 = 3 + 6\,, \\
r_{10} &= 30 \quad \text{from solution } 10 = 10 \quad \text{(no cuts)}\,.
\end{aligned}
$$

Corte da Haste: entendendo o problema

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

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

Problema do corte da Haste: resolvendo

Tarefa: Escreva um programa em C para resolver o problema do corte da haste. Não precisa se preocupar com eficiência neste momento, apenas em resolver o problema.

Haste: solução direta recursiva

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

Figure: Pseudo-código para a solução ingênua

Problema da Haste: solução direta recursiva

Uma implementação recursiva direta e ingênua para resolver o problema do corte das hastes de aço é espetacularmente ineficiente.

```
int haste (int n, int p[])
{
  if (n==0) return 0;
  int i, max = -1;
  for (i = 0; i < n; i++){
    int temp = p[n-i-1] + haste (i, p);
    if (temp > max) max = temp;
  }
  return max;
}
```

Corte da Haste: recálculo

Árvore de recursão para uma haste de tamanho 4.

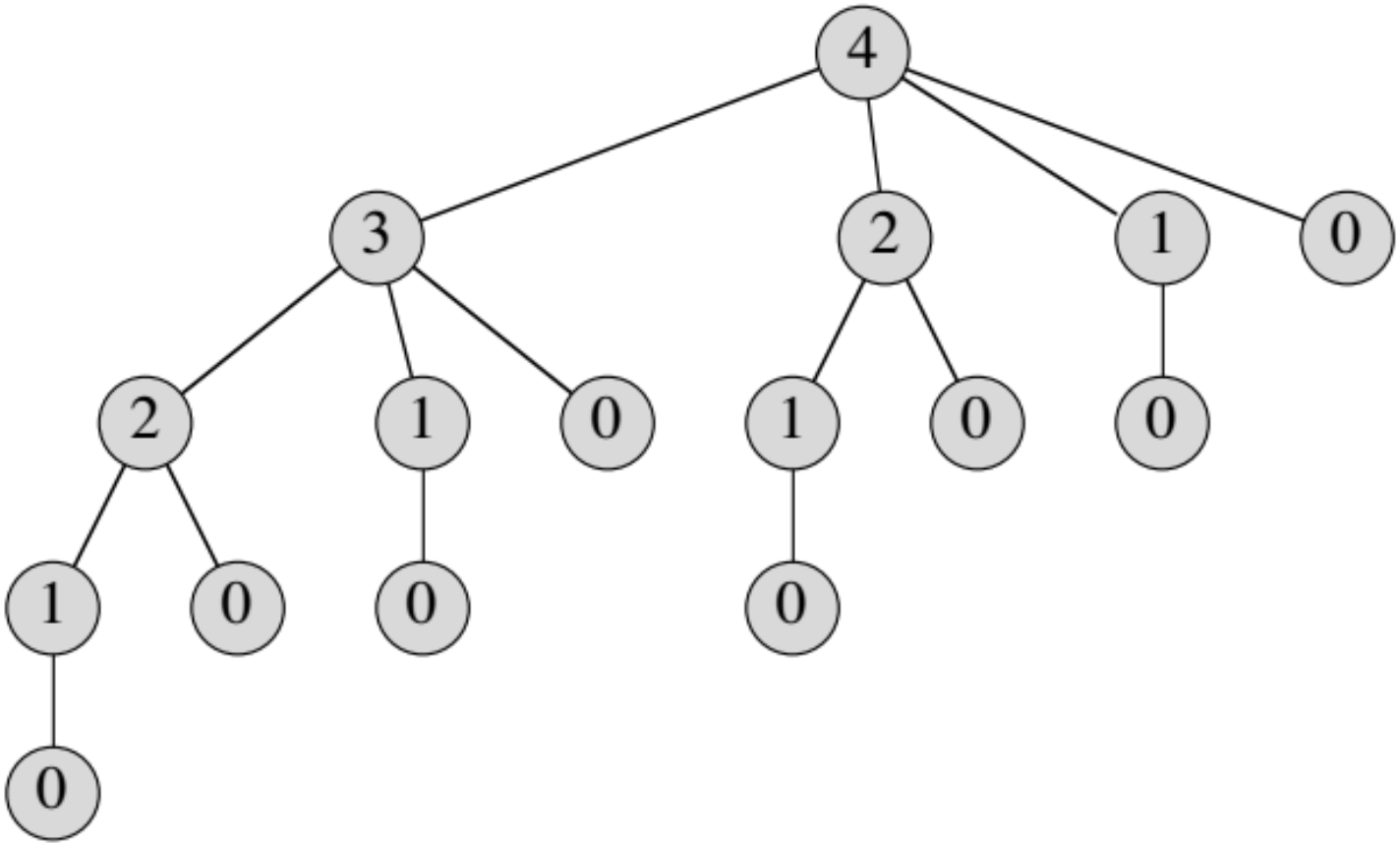

Figure: Árvore de recurão (CLR)

Problema do corte da Haste: melhorando a solução ingênua

Tarefa: Escreva um programa em C para resolver o problema do corte da haste usando programação dinâmica. Agora o importante é melhorar a eficiência e não apenas resolver o problema. Dica: use um meio de armazenar o que já foi calculado.

Corte da Haste: tamanho do problema

De quantas maneiras podemos cortar uma haste de tamanho n?

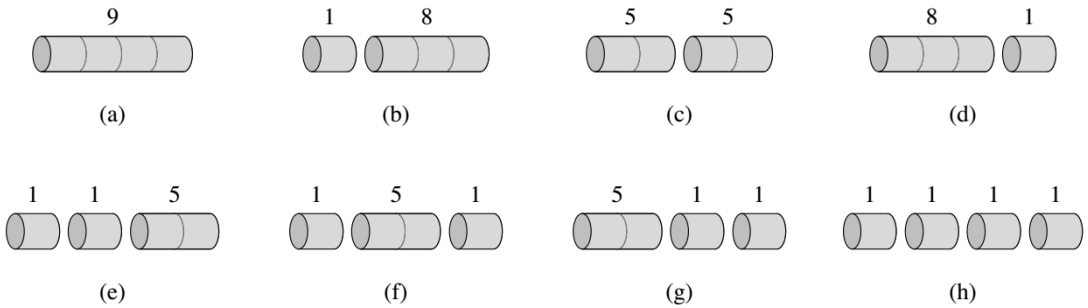

Figure: Corte da haste de tamanho 4

Corte da Haste: tamanho do problema

De quantas maneiras podemos cortar uma haste de tamanho n?

Proof Details: a rod of length n can have exactly n-1 possible cut positions – choose $0 \leq k \leq n$-1 actual cuts. We can choose the k cuts (without repetition) anywhere we want, so that for each such k the number of different choices is

$$\binom{n-1}{k}$$

When we sum up over all possibilities ($k = 0$ to $k = n$-1):

$$\sum_{k=0}^{n-1}\binom{n-1}{k} = \sum_{k=0}^{n-1}\frac{(n-1)!}{k!(n-1-k)!} = (1+1)^{n-1} = 2^{n-1}.$$

Figure: (Giampiero Pecelli)

Para uma haste de tamanho n há 2^{n-1} modos.

Problema da Haste: programação dinâmica

Outra dica: antes de ver a solução do problema, use o pseudo-código abaixo como modelo.

```
BOTTOM-UP-CUT-ROD(p, n)
1  let r[0..n] be a new array
2  r[0] = 0
3  for j = 1 to n
4       q = -∞
5       for i = 1 to j
6            q = max(q, p[i] + r[j - i])
7       r[j] = q
8  return r[n]
```

Figure: Pseudo-código para a solução com programação-dinâmica

Problema da Haste: programação dinâmica

```
int hasteDin (int n, int p[])
{
  int r[n+1], i, j;
  r[0] = 0;
  for (j = 1; j <= n; j++){
    int max = -1;
    for (i = 1; i <= j; i++){
      int temp = p[i-1] + r[j-i];
      if (temp > max) max = temp;
    }
    r[j] = max;
  }
  return r[n];
}
```