
*Universidade Federal do Piauí*
*Campus Senador Helvídio Nunes de Barros*

# Estudo Teórico e Experimental de Abordagens Recursivas e Dinâmicas Aplicadas ao Problema do Corte de Hastes

**Discentes:** Luciano Sousa Barbosa; Pedro Henrique Silva Rodrigues; Tiago Lima de Moura.

**Docente:** Raí Araújo de Miranda
**Disciplina:** Projeto e Análise de Algoritmos

# Sumário

1. Introdução
2. Referencial Teórico
3. Metodologia
4. Resultados e Discussão
5. Referências

# Introdução

- Análise de algoritmos é fundamental para avaliar desempenho computacional;
- Técnicas de projeto influenciam diretamente a eficiência dos algoritmos.

# Introdução

- Problema do Corte de Hastes:
  - Objetivo: maximizar o lucro a partir de uma tabela de preços;
  - Evidencia subestrutura ótima e sobreposição de subproblemas.

# Introdução

- Objetivos:
  - Comparar soluções recursivas  e com programação dinâmica;
  - Avaliar o desempenho computacional das abordagens;
  - Relacionar resultados empíricos com a análise teórica.

# Referencial Teórico - Abordagem Recursiva

- Técnica baseada em funções que chamam a si mesmas;
- Reduz progressivamente o tamanho da instância do problema;
- Não possui reaproveitamento automático de resultados intermediários;
- Estratégia:
  - Testar todas as possíveis divisões da haste de comprimento $n$;
  - Calcular recursivamente o lucro dos comprimentos restantes.

# Corte de Hastes Recursivo

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

```
int CorteDeHastes(int n, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

5

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

5

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

5

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

5

i = 1
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

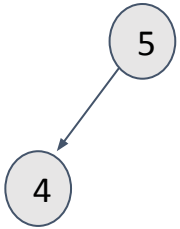

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



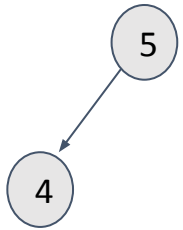

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



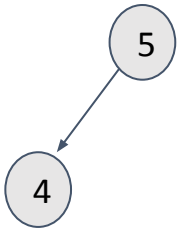

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



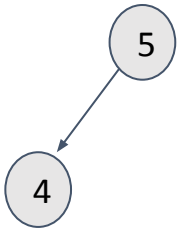

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



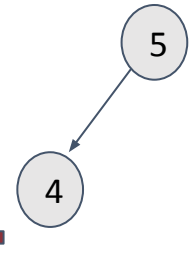

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



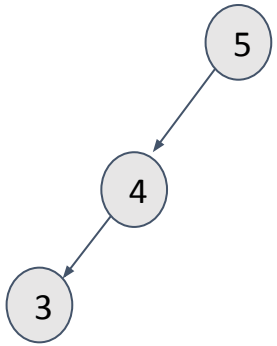

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



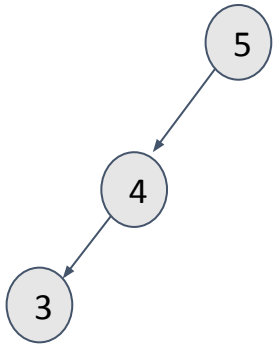

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



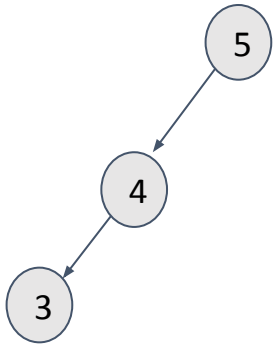

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



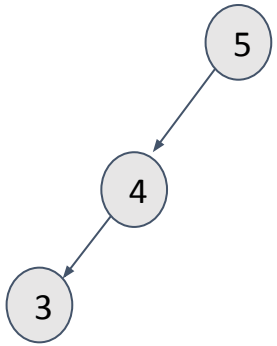

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



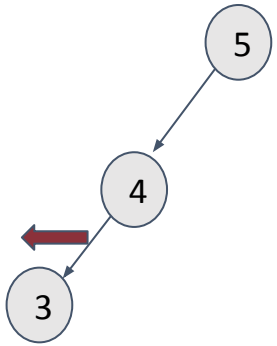

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



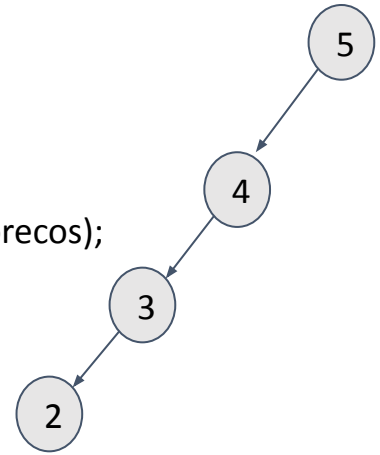

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



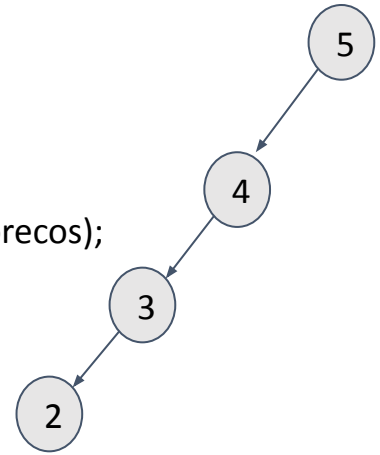

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



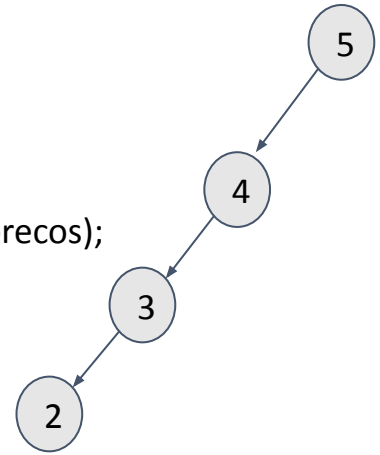

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



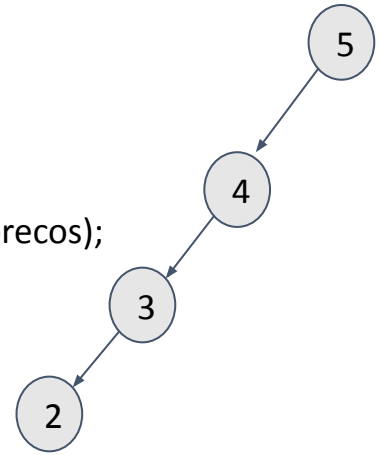

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



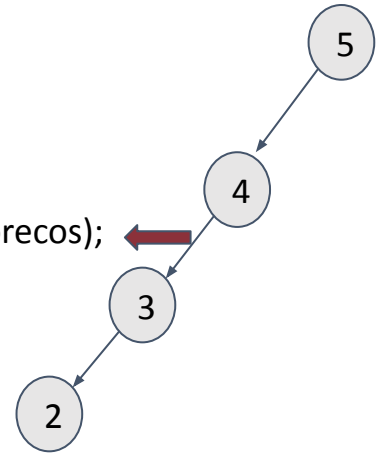

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



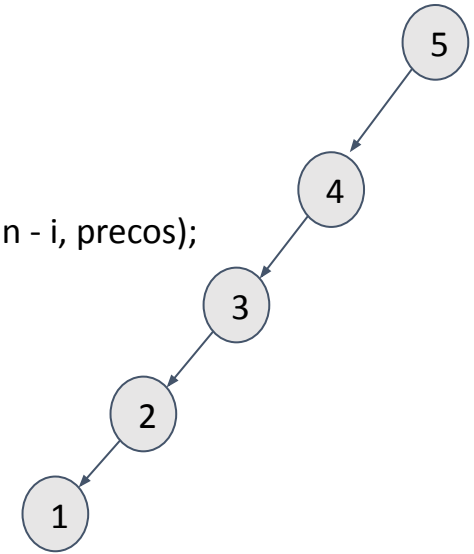

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 1, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



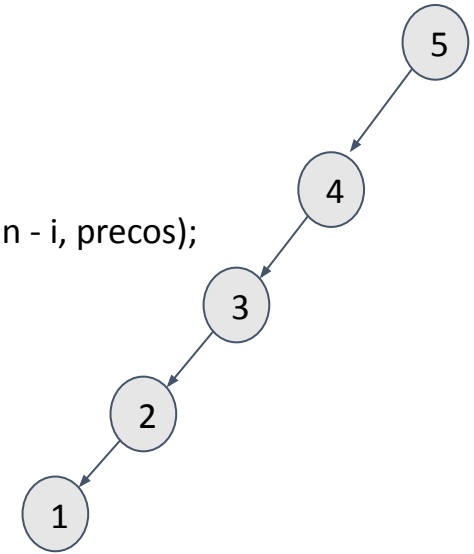

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 1, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



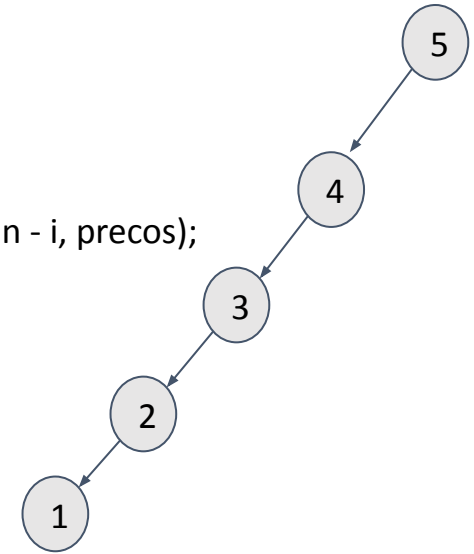

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 1, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



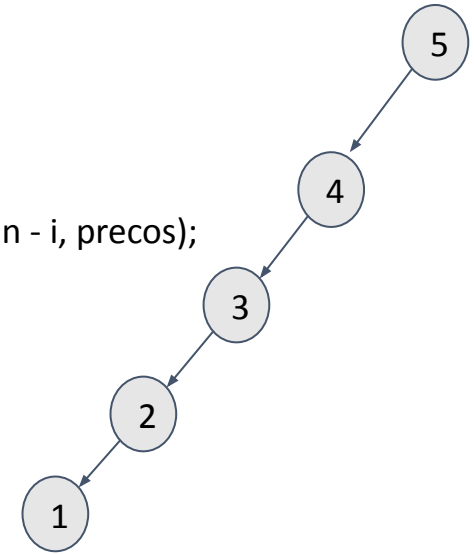

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 1, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



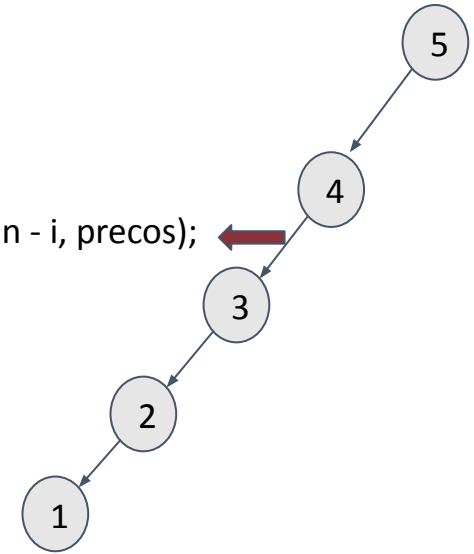

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 1, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



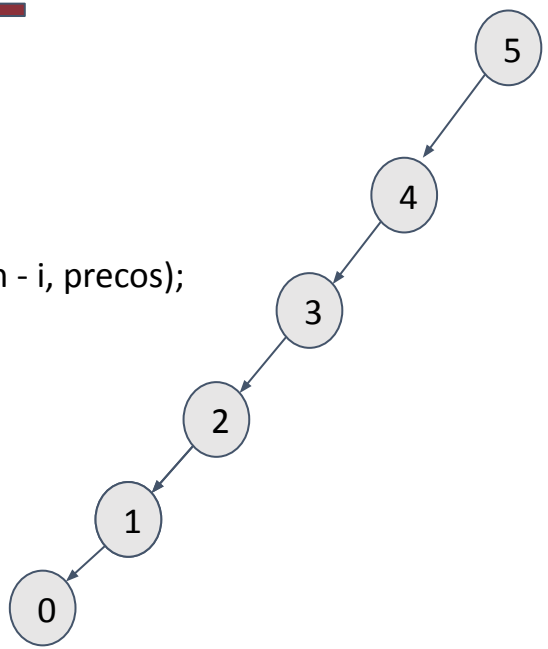

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 0, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



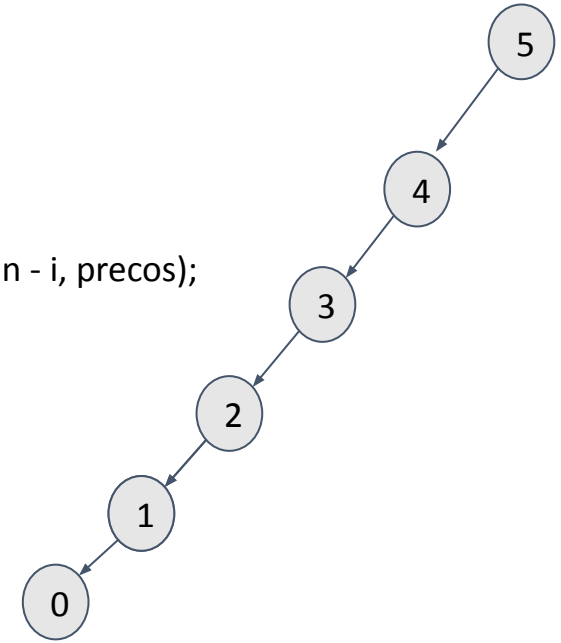

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 0, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



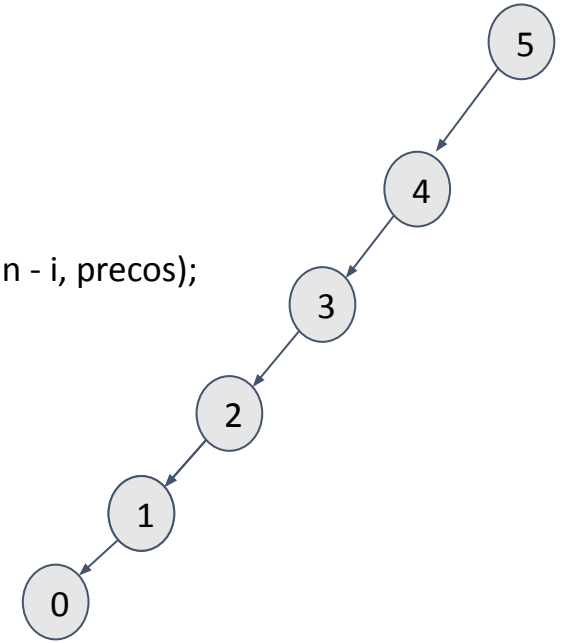

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 0, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



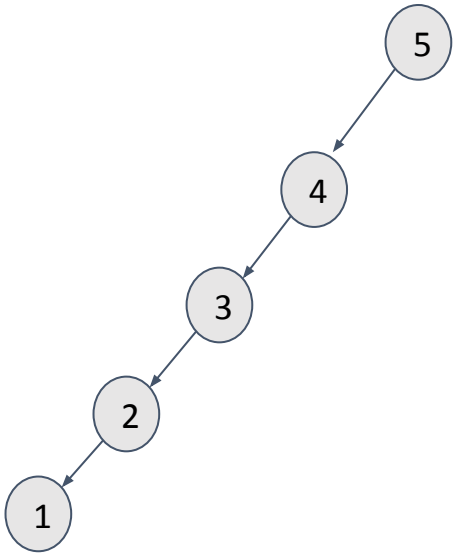

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = ??
lucro = ??
valor = ??

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 1, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 2+0 ;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



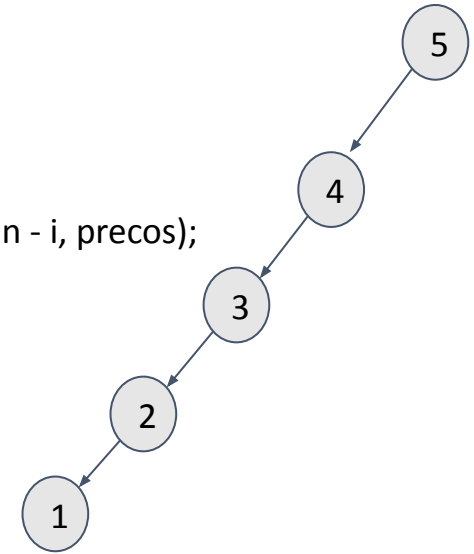

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = 2

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 1, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



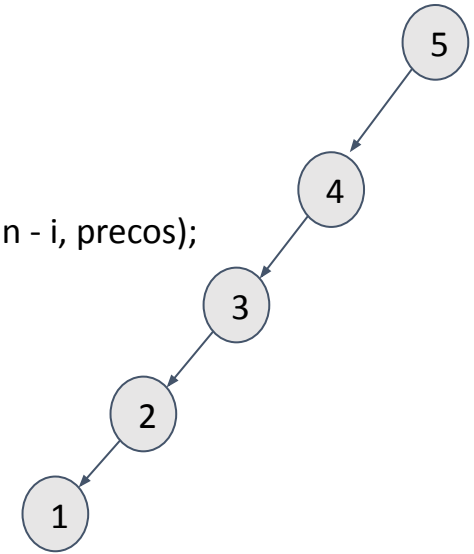

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = 2

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 1, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



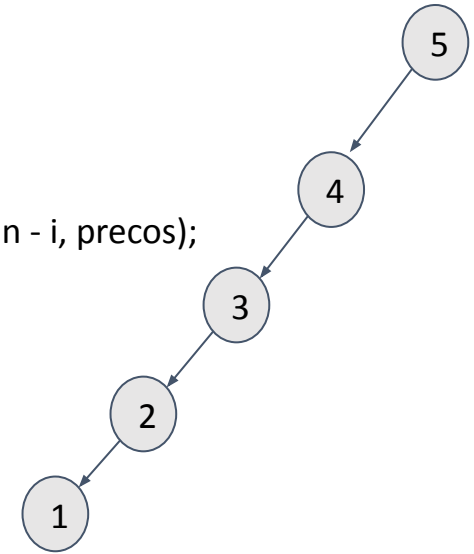

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = 2
valor = 2

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 1, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



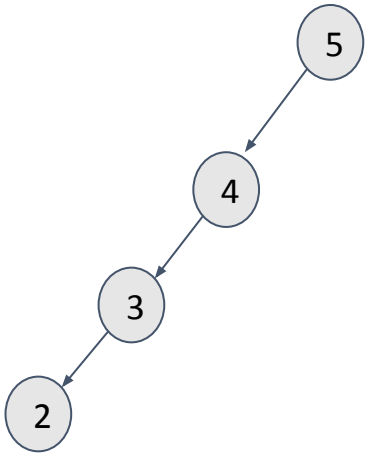

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = 2
valor = 2

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 2 + 2;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



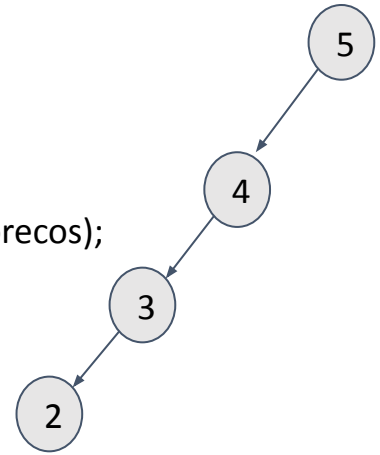

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = 4

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



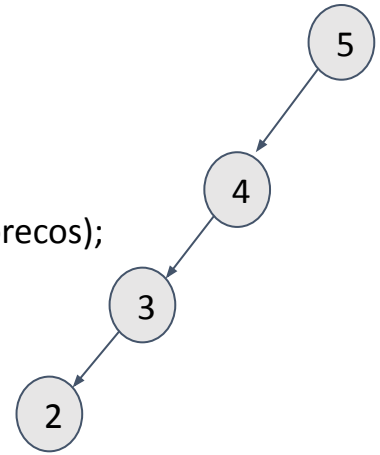

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = 4

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



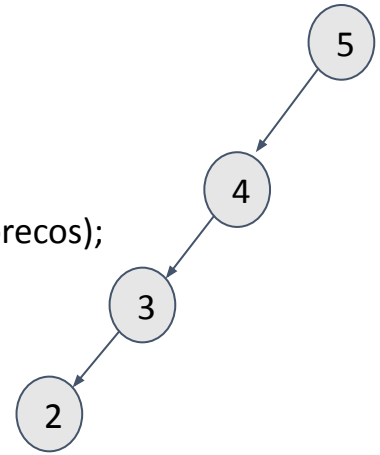

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = 4
valor = 4

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



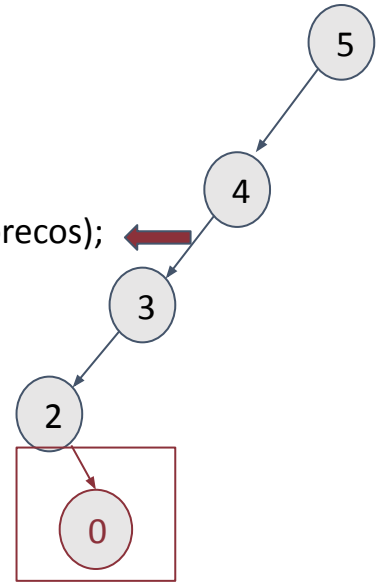

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 4
valor = 4

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



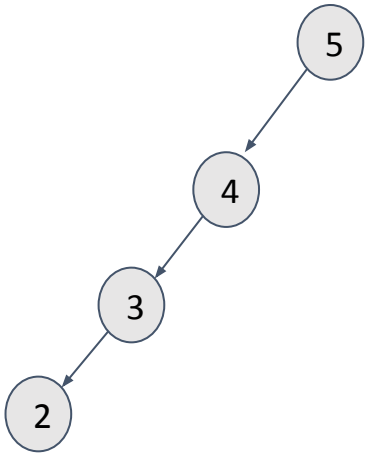

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 4
valor = 4

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 4 + 0;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



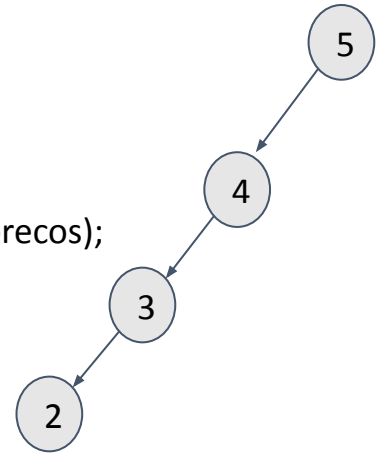

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 4
valor = 4

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



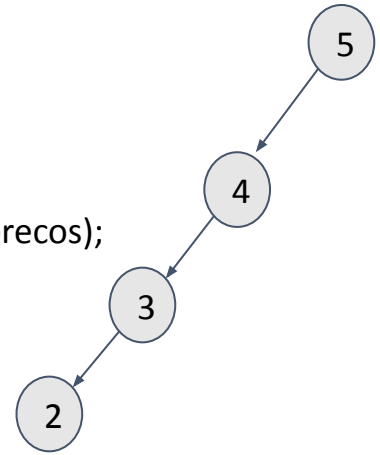

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 4
valor = 4

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | -1 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 2, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



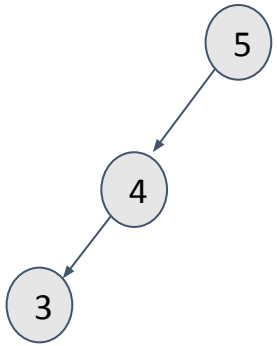

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 4
valor = 4

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 2 + 4;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



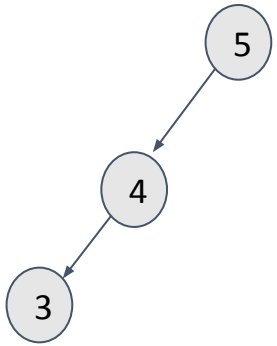

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = 6

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



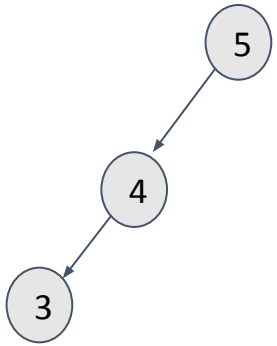

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = 6

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



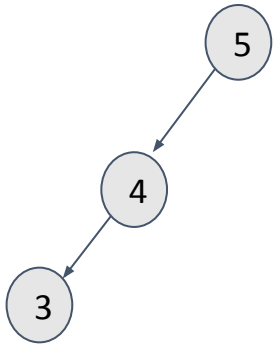

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = 6
valor = 6

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



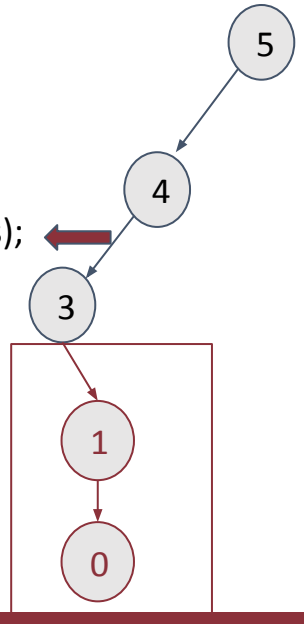

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 6
valor = 6

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



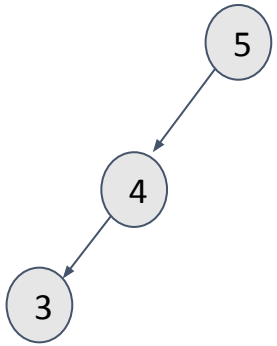

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 6
valor = 6

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 4 + 2;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



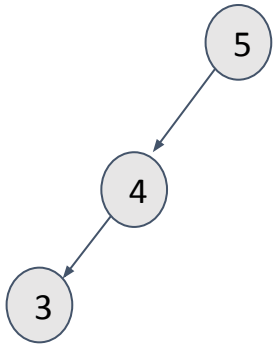

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 6
valor = 6

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



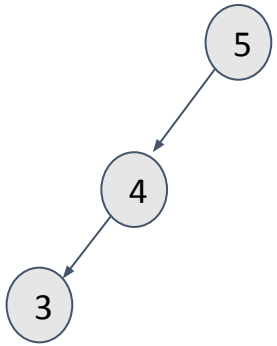

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 6
valor = 6

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



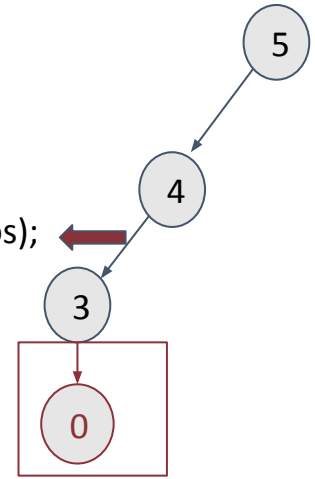

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 6
valor = 6

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



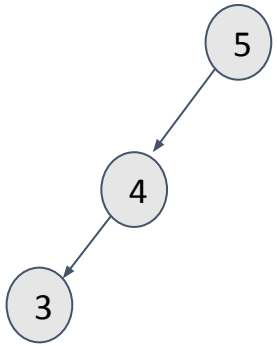

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 6
valor = 6

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 9 + 0;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



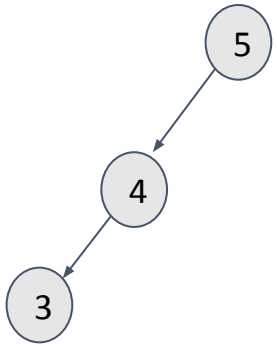

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 6
valor = 9

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



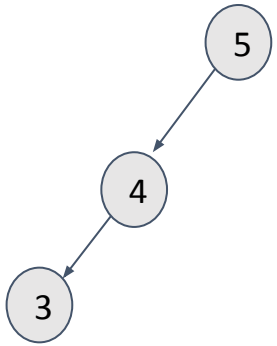

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 6
valor = 9

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



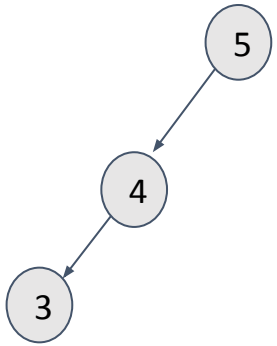

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 9
valor = 9

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | -1 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 3, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



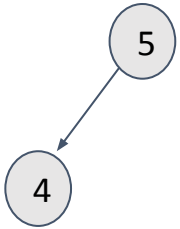

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 9
valor = 9

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 2 + 9;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



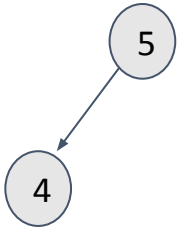

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = 11

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



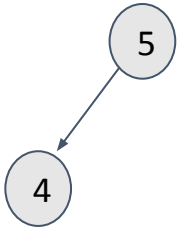

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = 11

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



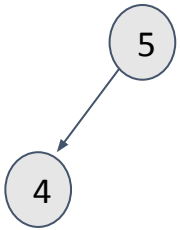

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = 11
valor = 11

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



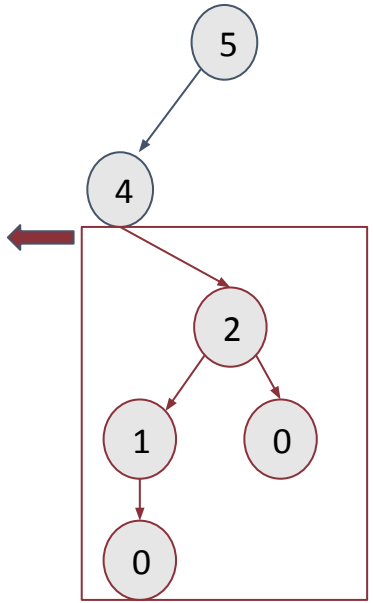

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 11
valor = 11

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



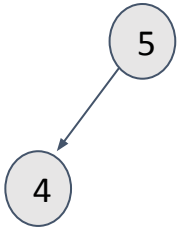

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 11
valor = 11

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 4 + 4;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



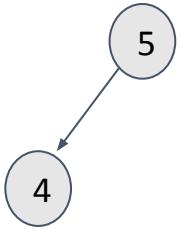

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 11
valor = 8

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



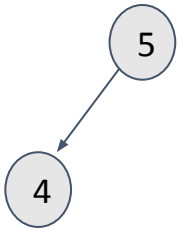

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 11
valor = 8

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



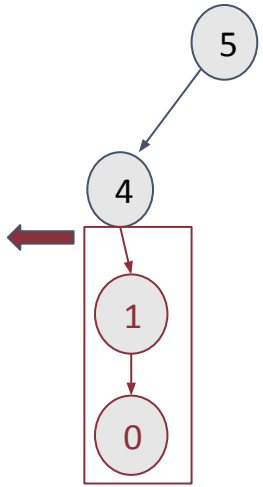

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 11
valor = 8

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



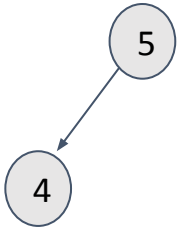

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 11
valor = 8

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 9 + 2;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



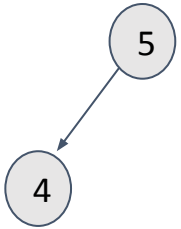

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 11
valor = 11

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



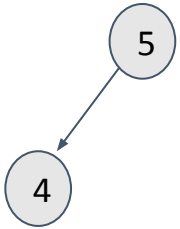

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 11
valor = 11

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



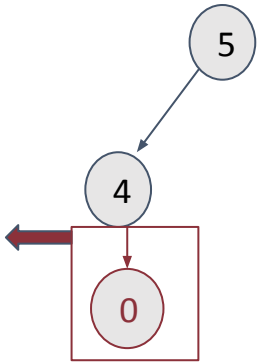

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 4
lucro = 11
valor = 11

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



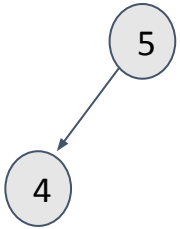

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 4
lucro = 11
valor = 11

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 8 + 0;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



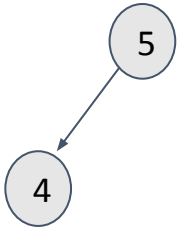

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 4
lucro = 11
valor = 8

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```



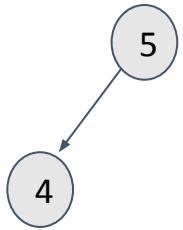

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 4
lucro = 11
valor = 8

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | -1 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 4, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```




| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 4
lucro = 11
valor = 8

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 2 + 11;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

5

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = 13

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

5

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 1
lucro = min
valor = 13

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

5

i = 1
lucro = 13
valor = 13

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

5

i = 2
lucro = 13
valor = 13

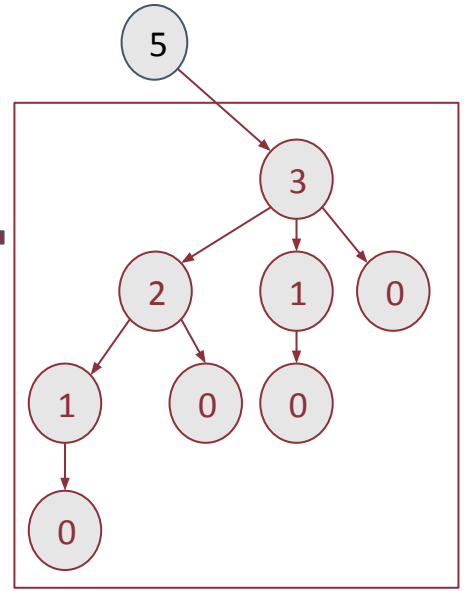
| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 13
valor = 13



# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 4 + 9;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

5

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 2
lucro = 13
valor = 13

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

5

i = 2
lucro = 13
valor = 13

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

5

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 13
valor = 13

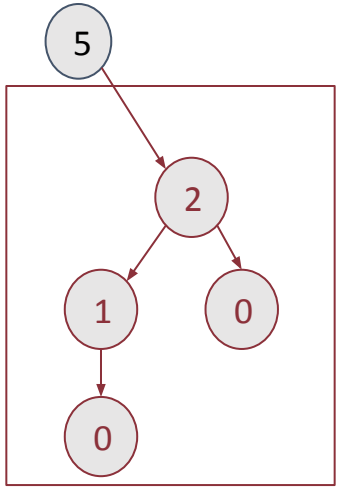
| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 13
valor = 13



# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 9 + 4;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

5

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 3
lucro = 13
valor = 13

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

5

i = 3
lucro = 13
valor = 13

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

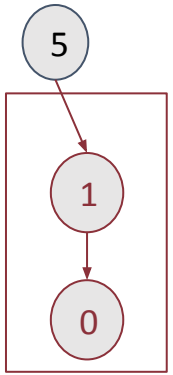
5

i = 4
lucro = 13
valor = 13

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```




| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 4
lucro = 13
valor = 13

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 8 + 2;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

5

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 4
lucro = 13
valor = 10

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

5

i = 4
lucro = 13
valor = 10

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

5

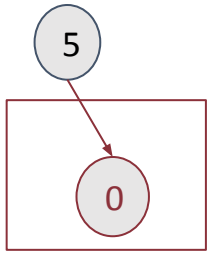
| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 5
lucro = 13
valor = 10

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```




| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 5
lucro = 13
valor = 10

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = 10 + 0;
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

5

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

i = 5
lucro = 13
valor = 10

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

5

i = 5
lucro = 13
valor = 10

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | -1 |

# Corte de Hastes Recursivo

```
int CorteDeHastes(n = 5, int precos[]) {
   if (n == 0) {
      return 0;
   }
   int lucro = INT_MIN;
   for (int i = 1; i <= n; i++) {
      int valor = precos[i] + CorteDeHastes(n - i, precos);
      if (valor > lucro)
         lucro = valor;
   }
   return lucro;
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

5

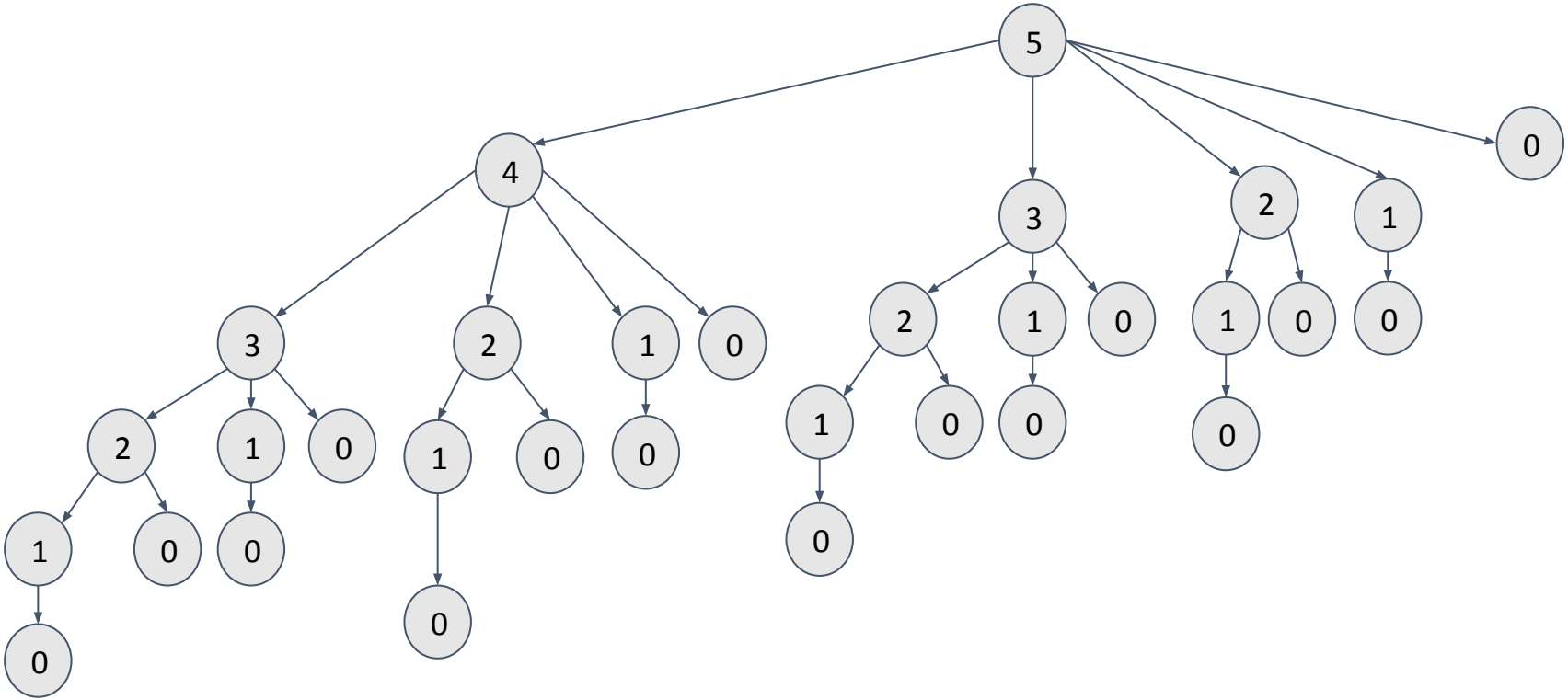
i = 5
lucro = 13
valor = 10

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 11 | 13 |

# Corte de Hastes Recursivo

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| P | 2 | 4 | 9 | 8 | 10 |
| So | 2 | 4 | 9 | 11 | 13 |

# Corte de Hastes Recursivo



# Referencial Teórico - Programação Dinâmica

- Técnica de projeto de algoritmos para problemas de otimização;
- Baseada no Princípio da Otimalidade de Bellman;
- Armazena e reaproveita resultados intermediários;
- Estratégia:
  - Decomposição sistemática em subproblemas menores;
  - Cálculo do lucro ótimo para cada comprimento.

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

n = 5

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | | | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

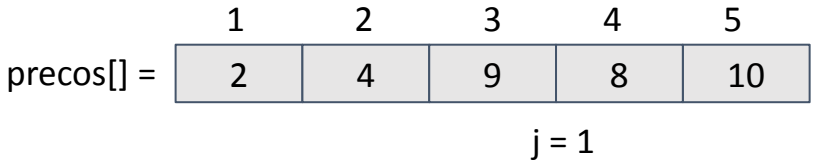
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

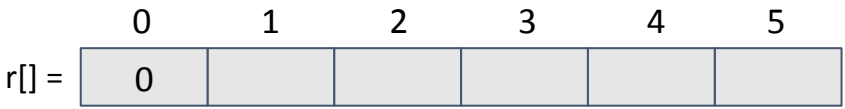
| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | | | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

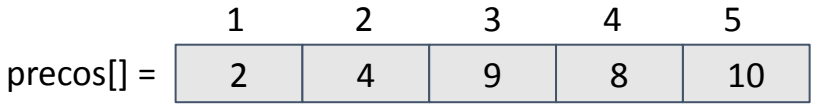


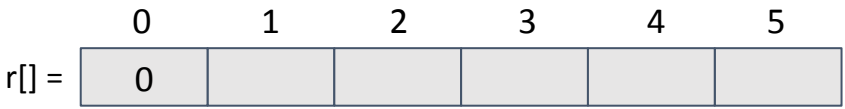
# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```



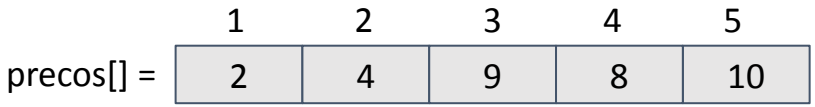

j = 1
lucro = INT_MIN



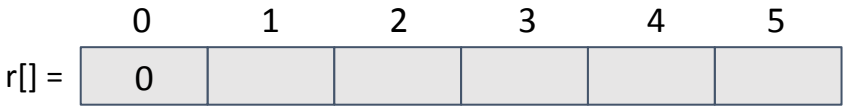
# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```



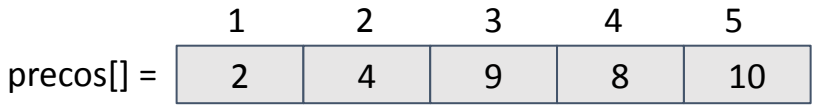

j = 1
lucro = INT_MIN
i = 1



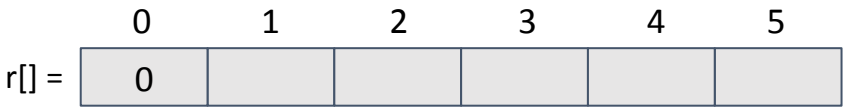
# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[1] + r[1 - 1];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```




j = 1
lucro = INT_MIN
i = 1
valor = 2 + 0 = 2



# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

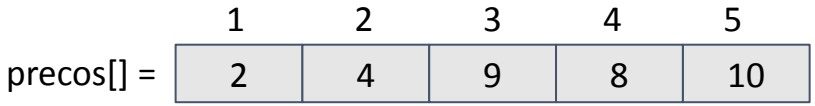
j = 1
lucro = INT_MIN
i = 1
valor = 2 + 0 = 2

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | | | | | |

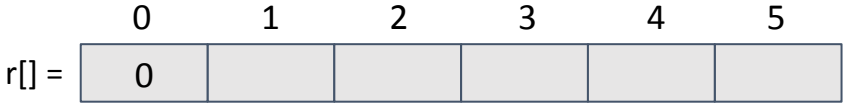
# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```



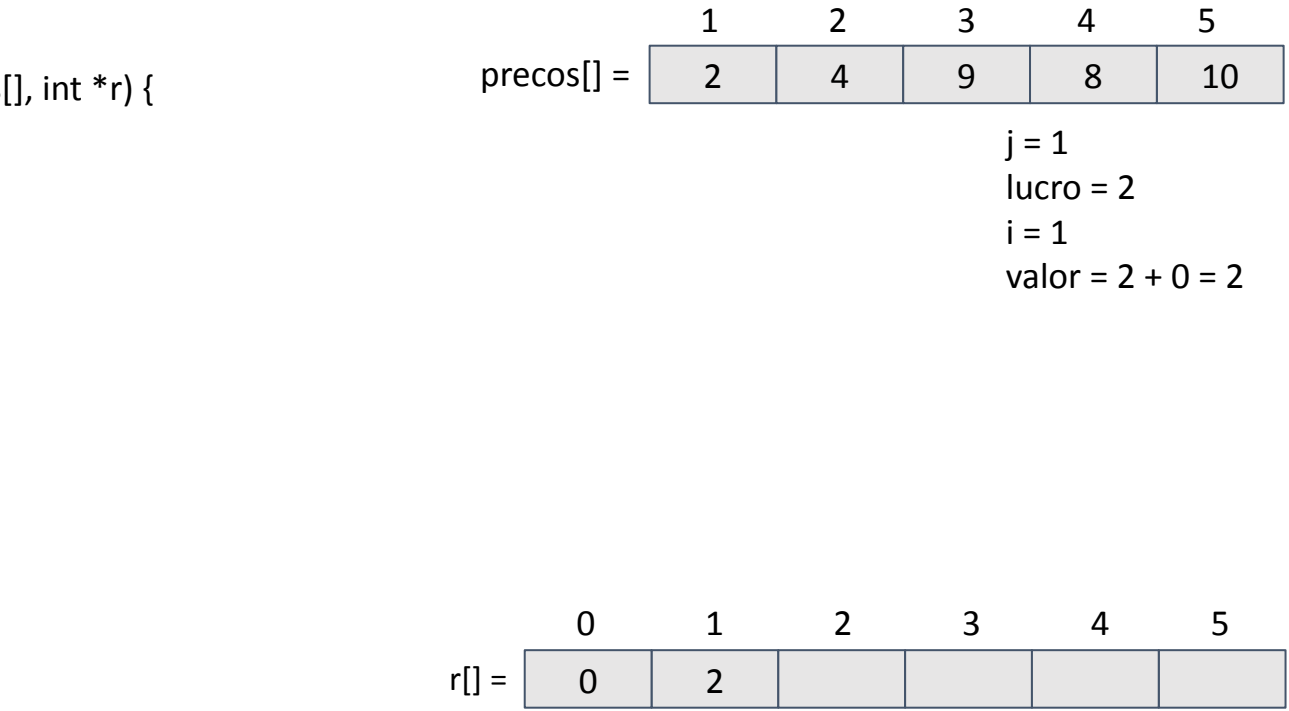

j = 1
lucro = 2
i = 1
valor = 2 + 0 = 2



# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```



# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 2
lucro = 2
i = 1
valor =

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 2
lucro = INT_MIN
i = 1
valor =

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 2
lucro = INT_MIN
i = 1
valor =

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[1] + r[2 - 1];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 2
lucro = INT_MIN
i = 1
valor = 2 + 2 = 4

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

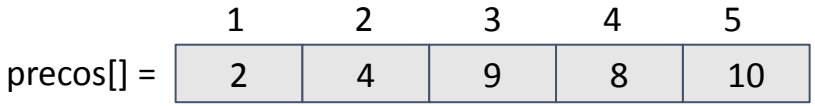
j = 2
lucro = INT_MIN
i = 1
valor = 4

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | | | | |

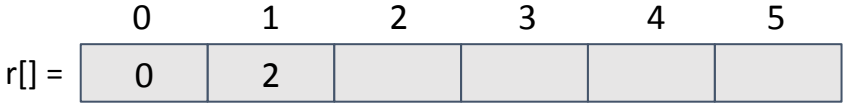
# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```




j = 2
lucro = 4
i = 1
valor = 4



# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[2] + r[2 - 2];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 2
lucro = 4
i = 2
valor = 4 + 0 = 4

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 2
lucro = 4
i = 2
valor = 4

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 2
lucro = 4
i = 2
valor = 4

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = 4
i = 2
valor = 4

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = INT_MIN
i = 2
valor = 4

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = INT_MIN
i = 1
valor =

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[1] + r[3 - 1];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = INT_MIN
i = 1
valor = 2 + 4 = 6

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = INT_MIN
i = 1
valor = 2 + 4 = 6

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = 6
i = 1
valor = 2 + 4 = 6

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = 6
i = 2
valor = 2 + 4 = 6

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[2] + r[3 - 2];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = 6
i = 2
valor = 4 + 2 = 6

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = 6
i = 2
valor = 4 + 2 = 6

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = 6
i = 3
valor = 4 + 2 = 6

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[3] + r[3 - 3];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = 6
i = 3
valor = 9 + 0 = 9

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = 6
i = 3
valor = 9 + 0 = 9

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = 9
i = 3
valor = 9 + 0 = 9

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 3
lucro = 9
i = 3
valor = 9 + 0 = 9

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 9
i = 3
valor = 9 + 0 = 9

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = INT_MIN
i = 3
valor = 9 + 0 = 9

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = INT_MIN
i = 1
valor = 9 + 0 = 9

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[1] + r[4 - 1];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = INT_MIN
i = 1
valor = 2 + 9 = 11

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = INT_MIN
i = 1
valor = 2 + 9 = 11

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 11
i = 1
valor = 2 + 9 = 11

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 11
i = 2
valor = 2 + 9 = 11

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[2] + r[4 - 2];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 11
i = 2
valor = 4 + 4 = 8

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 11
i = 2
valor = 4 + 4 = 8

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 11
i = 3
valor = 4 + 4 = 8

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[3] + r[4 - 3];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 11
i = 3
valor = 9 + 2 = 11

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 11
i = 3
valor = 9 + 2 = 11

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 11
i = 4
valor = 9 + 2 = 11

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[4] + r[4 - 4];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 11
i = 4
valor = 8 + 0 = 8

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 11
i = 4
valor = 8 + 0 = 8

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 4
lucro = 11
i = 4
valor = 8 + 0 = 8

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 11
i = 4
valor = 8 + 0 = 8

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = INT_MIN
i = 4
valor = 8 + 0 = 8

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = INT_MIN
i = 1
valor = 8 + 0 = 8

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[1] + r[5 - 1];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = INT_MIN
i = 1
valor = 2+11 = 13

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

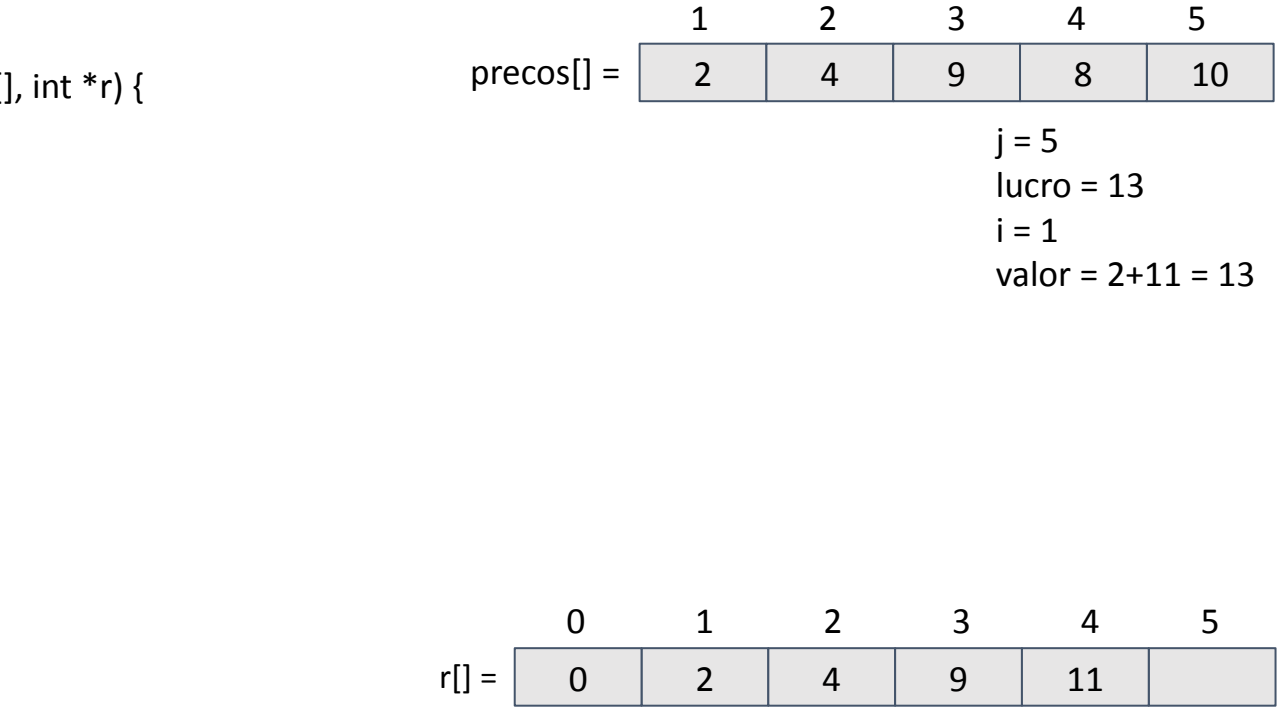
j = 5
lucro = INT_MIN
i = 1
valor = 2+11 = 13

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```



# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 2
valor = 2+11 = 13

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[2] + r[5 - 2];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 2
valor = 4+9 = 13

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 2
valor = 4+9 = 13

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 3
valor = 4+9 = 13

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[3] + r[5 - 3];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```



| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 3
valor = 9 + 4 = 13

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 3
valor = 9 + 4 = 13

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 4
valor = 9 + 4 = 13

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[4] + r[5 - 4];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 4
valor = 8+2 = 10

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 4
valor = 8+2 = 10

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 5
valor = 8+2 = 10

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[5] + r[5 - 5];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 5
valor = 10+0 = 10

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 5
valor = 10+0 = 10

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 5
lucro = 13
i = 5
valor = 10+0 = 10

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | 13 |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[n];
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 6
lucro = 13
i = 5
valor = 10+0 = 10

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | 13 |

# Corte de Hastes Dinâmico

```
int CorteDeHastesDP(int n, int precos[], int *r) {
   r[0] = 0;

   for (int j = 1; j <= n; j++) {
      int lucro = INT_MIN;
      for (int i = 1; i <= j; i++) {
         int valor = precos[i] + r[j - i];
         if (valor > lucro)
            lucro = valor;
      }
      r[j] = lucro;
   }
   return r[5] = 13;
}
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| precos[] = | 2 | 4 | 9 | 8 | 10 |

j = 6
lucro = 13
i = 5
valor = 10+0 = 10

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | 13 |

# Corte de Hastes Dinâmico

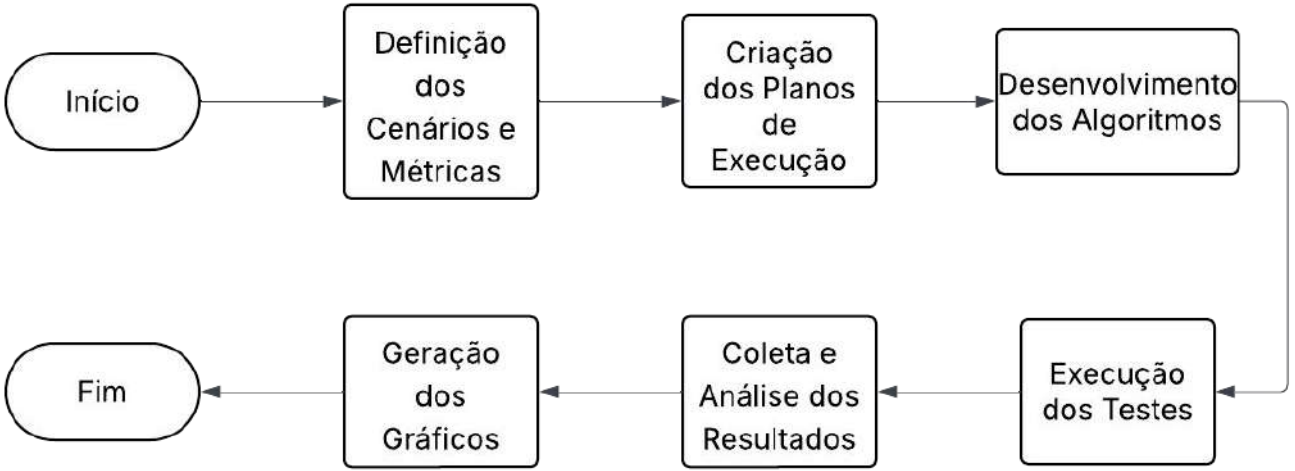
| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 2 | 4 | 9 | 8 | 10 |

- Logo, o **lucro máximo** para tamanho 5 é **13**.

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| r[] = | 0 | 2 | 4 | 9 | 11 | 13 |

# Metodologia

**Figura 1:** Fluxograma Metodológico.




**Fonte:** Autor Próprio, 2026.

# Metodologia

**Tabela 1:** Ambiente de Execução.

| | |
|---|---|
| Processador: | Intel Core i5-12450H 12º Gen (2.00 GHz) (8 núcleos, 12 threads, 12 MB cache) |
| RAM: | 16,0 GB @ 3200 MHz (utilizável: 15,7 GB) |
| SO: | Windows 11 Home Single Language (Executado no Ubuntu 24.04.3 LTS via WSL2) |
| Linguagens: | C e Python |

**Fonte:** Autor Próprio, 2026.

# Resultados e Discussão

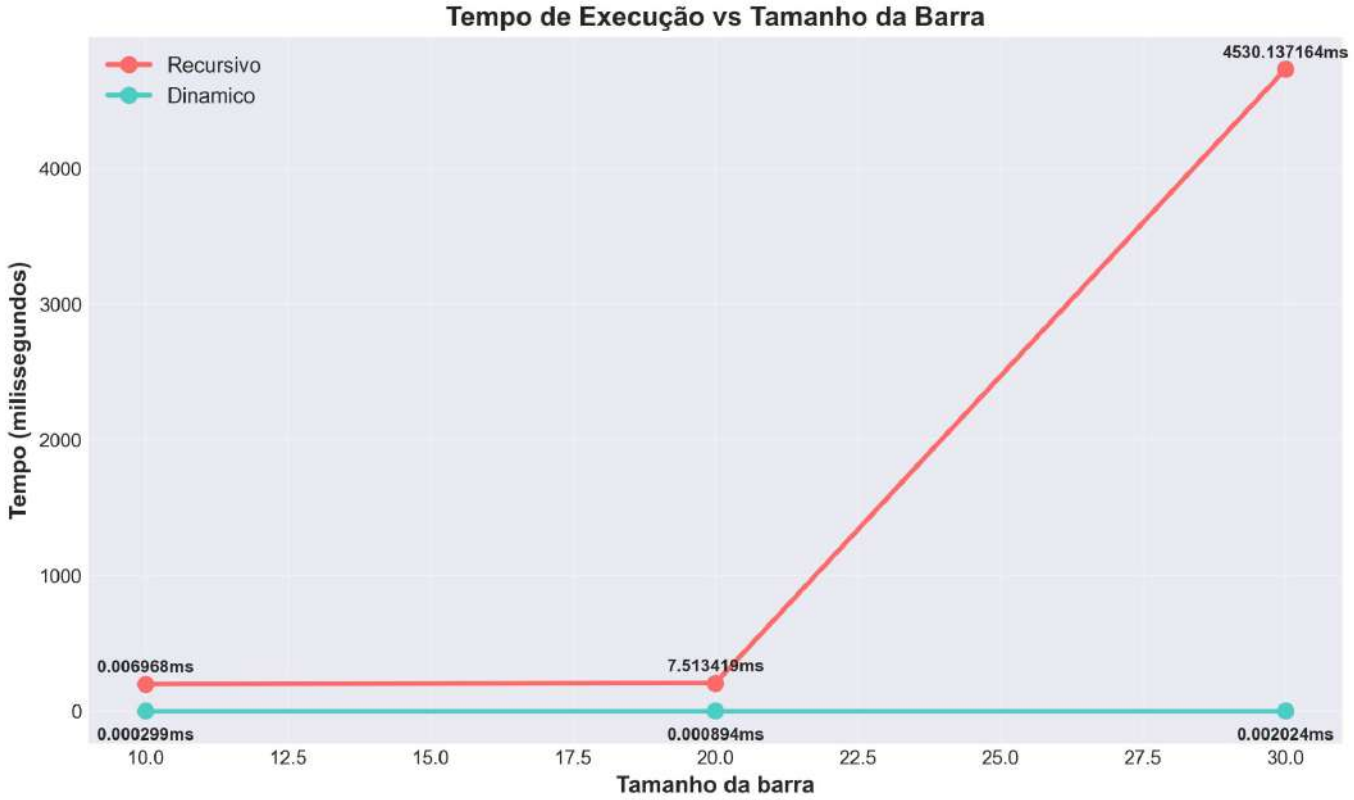



**Gráfico 1:** Tempo Médio de Execução.

**Fonte:** Autor Próprio, 2026.

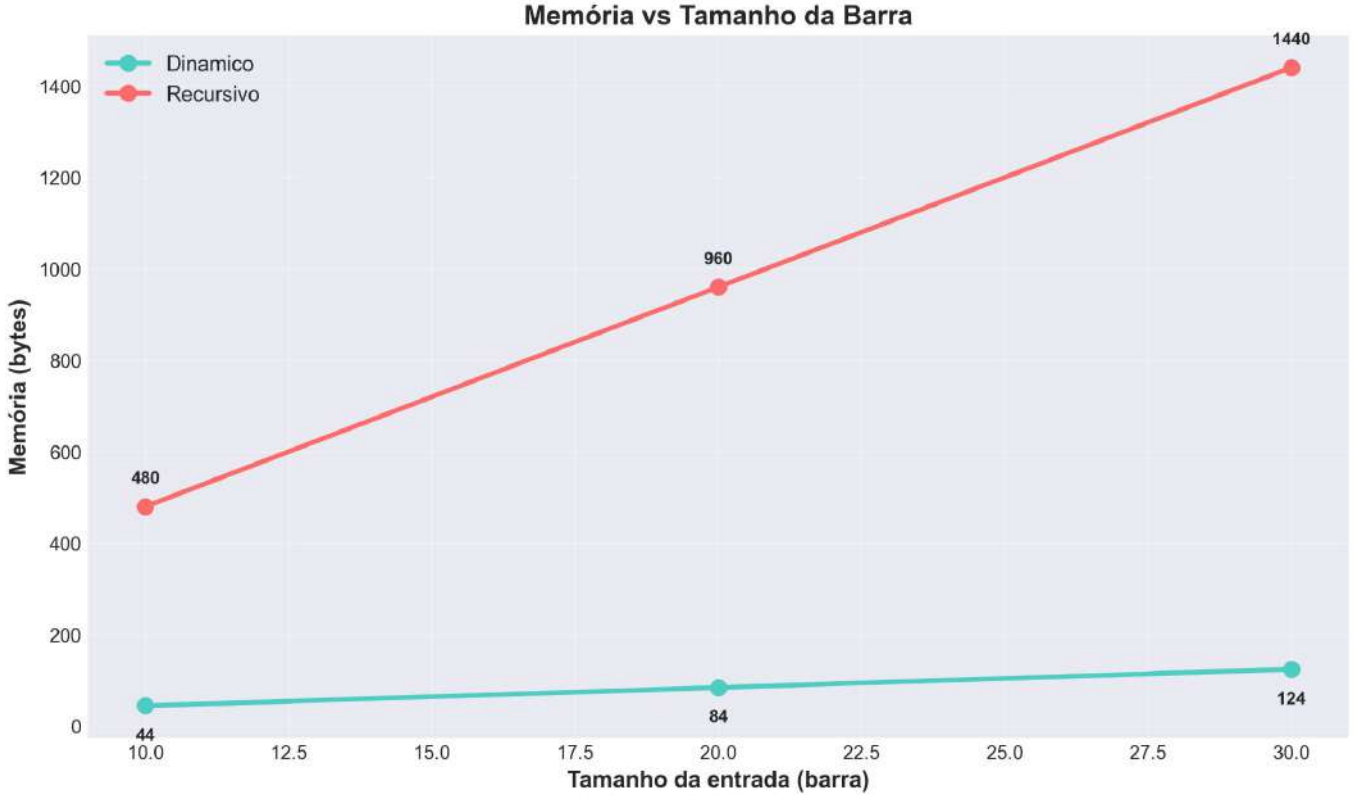



**Gráfico 2:** Consumo Estimado de Memória.

**Fonte:** Autor Próprio, 2026.

**Tabela 2:** Complexidade Assintótica dos Algoritmos.

| Abordagem | Tempo | Espaço |
|---|---|---|
| Recursiva | $O(2^n)$ | O(n) |
| Dinâmica | $O(n^2)$ | O(n) |

**Fonte:** Autor Próprio, 2026.

# Conclusão

- Programação dinâmica demonstrou superioridade empírica sobre a recursão;
- Diferença de desempenho cresce rapidamente com o aumento da entrada;
- Resultados confirmam a teoria da sobreposição de subproblemas.

# Referências


Aho, A. V., & Hopcroft, J. E. (1974). The design and analysis of computer algorithms. Pearson Education India.

Bellman, R., & Lee, E. (1984). History and development of dynamic programming. IEEE Control Systems Magazine, 4(4), 24-28.

Bellman, R. E., & Dreyfus, S. E. (2015). Applied dynamic programming. Princeton university press.

Chu, C., & Antonio, J. (1999). Approximation algorithms to solve real-life multicriteria cutting stock problems. Operations Research, 47(4), 495-508.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms. MIT press.

Dasgupta, S., Papadimitriou, C. H., & Vazirani, U. V. Dynamic programming. Algorithms, 1, 169-199.

# Referências

Kabanov, J., & Vene, V. (2006, July). Recursion schemes for dynamic programming. In International Conference on Mathematics of Program Construction (pp. 235-252). Berlin, Heidelberg: Springer Berlin Heidelberg.

Kleinberg, J., & Tardos, E. (2006). Algorithm design. Pearson Education India.

Tanır, D., Ugurlu, O., Guler, A., & Nuriyev, U. (2019). One-dimensional cutting stock problem with divisible items: A case study in steel industry. TWMS Journal of Applied and Engineering Mathematics, 9(3), 473-484.

Yang, X. S. (2011, May). Metaheuristic optimization: algorithm analysis and open problems. In International symposium on experimental algorithms (pp. 21-32). Berlin, Heidelberg: Springer Berlin Heidelberg.