Aula 16
Programação Dinâmica
Linha de Montagem

Projeto e Análise de Algoritmos

Professor Eurinardo Rodrigues Costa
Universidade Federal do Ceará
Campus Russas

2021.1

Sumário

Aula Passada

Aula Passada

Definição (Programação Dinâmica)

Seguir os passos:

Aula Passada

Definição (Programação Dinâmica)

Seguir os passos:

(1) *Verificar propriedade de* **subestrutura ótima**

Aula Passada

Definição (Programação Dinâmica)

Seguir os passos:

(1) *Verificar propriedade de* **subestrutura ótima**

(2) *Obter uma recursão para o* **valor ótimo** *do problema*

Aula Passada

Definição (Programação Dinâmica)

Seguir os passos:

(1) *Verificar propriedade de* **subestrutura ótima**

(2) *Obter uma recursão para o* **valor ótimo** *do problema*

(3) *Obter um algoritmo Bottom-UP para calcular o* **valor ótimo**

Aula Passada

Definição (Programação Dinâmica)

Seguir os passos:

(1) *Verificar propriedade de* **subestrutura ótima**

(2) *Obter uma recursão para o* **valor ótimo** *do problema*

(3) *Obter um algoritmo Bottom-UP para calcular o* **valor ótimo**

(4) *Obter uma* **solução ótima** *do problema.*

Linha de Montagem

Linha de Montagem - Memoização

Linha de Montagem - Memoização

Algoritmo 2: Linha-Rec(a, t, e, x, n)

Entrada: a, $t \rightarrow$ matrizes $2 \times n$

Linha de Montagem - Memoização

Algoritmo 3: Linha-Rec(a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$

$e, x \rightarrow$ vetores de tamanho 2

Linha de Montagem - Memoização

Algoritmo 4: Linha-Rec(a, t, e, x, n)

Entrada: a, $t \rightarrow$ matrizes $2 \times n$

$e, x \rightarrow$ vetores de tamanho 2

$n \rightarrow$ número de estações

Linha de Montagem - Memoização

Algoritmo 5: Linha-Rec(a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$

$e, x \rightarrow$ vetores de tamanho 2

$n \rightarrow$ número de estações

Saída: Valor Ótimo

Linha de Montagem - Memoização

Algoritmo 6: Linha-Rec(a, t, e, x, n)

Entrada: a, $t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetores de tamanho 2
$n \rightarrow$ número de estações

Saída: Valor Ótimo

1 **criar vetor** $f_1[1 \cdots n]$, $f_2[1 \cdots n]$ com -1's

Linha de Montagem - Memoização

Algoritmo 7: Linha-Rec(a, t, e, x, n)

Entrada: a, $t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetores de tamanho 2
$n \rightarrow$ número de estações

Saída: Valor Ótimo

1 **criar vetor** $f_1[1 \cdots n]$, $f_2[1 \cdots n]$ com -1's

2 $f_1[1] \leftarrow e_1 + a_{1,1}$

Linha de Montagem - Memoização

Algoritmo 8: Linha-Rec(a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$

$e, x \rightarrow$ vetores de tamanho 2

$n \rightarrow$ número de estações

Saída: Valor Ótimo

1 **criar vetor** $f_1[1 \cdots n]$, $f_2[1 \cdots n]$ com -1's

2 $f_1[1] \leftarrow e_1 + a_{1,1}$

3 $f_2[1] \leftarrow e_2 + a_{2,1}$

Linha de Montagem - Memoização

Algoritmo 9: Linha-Rec(a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetores de tamanho 2
$n \rightarrow$ número de estações

Saída: Valor Ótimo

1 **criar vetor** $f_1[1 \cdots n]$, $f_2[1 \cdots n]$ com -1's
2 $f_1[1] \leftarrow e_1 + a_{1,1}$
3 $f_2[1] \leftarrow e_2 + a_{2,1}$
4 $z_1 \leftarrow$ Linha-Rec-Rec(1, n, a, t, e, x, n)

Linha de Montagem - Memoização

Algoritmo 10: Linha-Rec(a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetores de tamanho 2
$n \rightarrow$ número de estações

Saída: Valor Ótimo

1 **criar vetor** $f_1[1 \cdots n]$, $f_2[1 \cdots n]$ com -1's
2 $f_1[1] \leftarrow e_1 + a_{1,1}$
3 $f_2[1] \leftarrow e_2 + a_{2,1}$
4 $z_1 \leftarrow$ Linha-Rec-Rec(1, n, a, t, e, x, n)
5 $z_2 \leftarrow$ Linha-Rec-Rec(2, n, a, t, e, x, n)

Linha de Montagem - Memoização

Algoritmo 11: Linha-Rec(a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetores de tamanho 2
$n \rightarrow$ número de estações

Saída: Valor Ótimo

1 **criar vetor** $f_1[1 \cdots n]$, $f_2[1 \cdots n]$ com -1's
2 $f_1[1] \leftarrow e_1 + a_{1,1}$
3 $f_2[1] \leftarrow e_2 + a_{2,1}$
4 $z_1 \leftarrow$ Linha-Rec-Rec(1, n, a, t, e, x, n)
5 $z_2 \leftarrow$ Linha-Rec-Rec(2, n, a, t, e, x, n)
6 **retorne** $\min\{z_1 + x_1, z_2 + x_2\}$

Linha de Montagem - Memoização

Linha de Montagem - Memoização

Algoritmo 13: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$

Linha de Montagem - Memoização

Algoritmo 14: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$

$e, x \rightarrow$ vetor de tamanho 2

Linha de Montagem - Memoização

Algoritmo 15: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$

$e, x \rightarrow$ vetor de tamanho 2

$n \rightarrow$ número de estações

Linha de Montagem - Memoização

Algoritmo 16: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$

$e, x \rightarrow$ vetor de tamanho 2

$n \rightarrow$ número de estações

$l \rightarrow$ linha

Linha de Montagem - Memoização

Algoritmo 17: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$

$e, x \rightarrow$ vetor de tamanho 2

$n \rightarrow$ número de estações

$l \rightarrow$ linha

$k \rightarrow$ estação

Linha de Montagem - Memoização

Algoritmo 18: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetor de tamanho 2
$n \rightarrow$ número de estações
$l \rightarrow$ linha
$k \rightarrow$ estação

Saída: Valor Ótimo para sair da (linha l, estação k)

Linha de Montagem - Memoização

Algoritmo 19: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetor de tamanho 2
$n \rightarrow$ número de estações
$l \rightarrow$ linha
$k \rightarrow$ estação

Saída: Valor Ótimo para sair da (linha l, estação k)

1 **se** $l = 1$ **então**

Linha de Montagem - Memoização

Algoritmo 20: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetor de tamanho 2
$n \rightarrow$ número de estações
$l \rightarrow$ linha
$k \rightarrow$ estação

Saída: Valor Ótimo para sair da (linha l, estação k)

```
1 se l = 1 então
2     se f_1[k] ≠ −1 então
```

Linha de Montagem - Memoização

Algoritmo 21: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetor de tamanho 2
$n \rightarrow$ número de estações
$l \rightarrow$ linha
$k \rightarrow$ estação

Saída: Valor Ótimo para sair da (linha l, estação k)

```
1 se l = 1 então
2     se f_1[k] ≠ −1 então
3         retorne f_1[k]
```

Linha de Montagem - Memoização

Algoritmo 22: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetor de tamanho 2
$n \rightarrow$ número de estações
$l \rightarrow$ linha
$k \rightarrow$ estação

Saída: Valor Ótimo para sair da (linha l, estação k)

1 **se** $l = 1$ **então**
2 **se** $f_1[k] \neq -1$ **então**
3 **retorne** $f_1[k]$
4 $f_1[k] \leftarrow \min \left\{ \begin{array}{l} \text{Linha-Rec-Rec}(1, k-1) \\ \text{Linha-Rec-Rec}(2, k-1) + t_{2,k-1} \end{array} \right\} + a_{1,k}$

Linha de Montagem - Memoização

Algoritmo 23: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetor de tamanho 2
$n \rightarrow$ número de estações
$l \rightarrow$ linha
$k \rightarrow$ estação

Saída: Valor Ótimo para sair da (linha l, estação k)

1 **se** $l = 1$ **então**
2 **se** $f_1[k] \neq -1$ **então**
3 **retorne** $f_1[k]$
4 $f_1[k] \leftarrow \min \left\{ \begin{array}{l} \text{Linha-Rec-Rec}(1, k-1) \\ \text{Linha-Rec-Rec}(2, k-1) + t_{2,k-1} \end{array} \right\} + a_{1,k}$
5 **retorne** $f_1[k]$

Linha de Montagem - Memoização

Algoritmo 24: Linha-Rec-Rec(l, k, a, t, e, x, n)

Entrada: $a,\ t \rightarrow$ matrizes $2 \times n$
$e, x \rightarrow$ vetor de tamanho 2
$n \rightarrow$ número de estações
$l \rightarrow$ linha
$k \rightarrow$ estação

Saída: Valor Ótimo para sair da (linha l, estação k)

1 **se** $l = 1$ **então**
2 **se** $f_1[k] \neq -1$ **então**
3 **retorne** $f_1[k]$
4 $f_1[k] \leftarrow \min \left\{ \begin{array}{l} \text{Linha-Rec-Rec}(1, k-1) \\ \text{Linha-Rec-Rec}(2, k-1) + t_{2,k-1} \end{array} \right\} + a_{1,k}$
5 **retorne** $f_1[k]$
6 **senão**
7 **se** $f_2[k] \neq -1$ **então**
8 **retorne** $f_2[k]$
9 $f_2[k] \leftarrow \min \left\{ \begin{array}{l} \text{Linha-Rec-Rec}(2, k-1) \\ \text{Linha-Rec-Rec}(1, k-1) + t_{1,k-1} \end{array} \right\} + a_{2,k}$
10 **retorne** $f_2[k]$

Linha de Montagem - PD

Linha de Montagem - PD

Algoritmo 26: Linha-de-Motagem(a, t, e, x, n)

1 **criar vetor** $f_1[1 \cdots n]$, $f_2[1 \cdots n]$;

Linha de Montagem - PD

Algoritmo 27: Linha-de-Motagem(a, t, e, x, n)

1 **criar vetor** $f_1[1 \cdots n]$, $f_2[1 \cdots n]$;
2 $f_1[1] \leftarrow e_1 + a_{1,1}$

Linha de Montagem - PD

Algoritmo 28: Linha-de-Motagem(a, t, e, x, n)

1 **criar vetor** $f_1[1 \cdots n]$, $f_2[1 \cdots n]$;
2 $f_1[1] \leftarrow e_1 + a_{1,1}$
3 $f_2[1] \leftarrow e_2 + a_{2,1}$

Linha de Montagem - PD

Algoritmo 29: Linha-de-Motagem(a, t, e, x, n)

1 **criar vetor** $f_1[1 \cdots n]$, $f_2[1 \cdots n]$;
2 $f_1[1] \leftarrow e_1 + a_{1,1}$
3 $f_2[1] \leftarrow e_2 + a_{2,1}$
4 **para** $k \leftarrow 2$ **até** n **faça**

Linha de Montagem - PD

Algoritmo 30: Linha-de-Motagem(a, t, e, x, n)

```
1 criar vetor f1[1 ··· n], f2[1 ··· n];
2 f1[1] ← e1 + a1,1
3 f2[1] ← e2 + a2,1
4 para k ← 2 até n faça
5     se f1[k − 1] ≤ f2[k − 1] + t2,k−1 então
```

Linha de Montagem - PD

Algoritmo 31: Linha-de-Motagem(a, t, e, x, n)

1 **criar vetor** $f_1[1 \cdots n]$, $f_2[1 \cdots n]$;
2 $f_1[1] \leftarrow e_1 + a_{1,1}$
3 $f_2[1] \leftarrow e_2 + a_{2,1}$
4 **para** $k \leftarrow 2$ **até** n **faça**
5 | **se** $f_1[k-1] \leq f_2[k-1] + t_{2,k-1}$ **então**
6 | | $f_1[k] \leftarrow f_1[k-1] + a_{1,k}$

Linha de Montagem - PD

Algoritmo 32: Linha-de-Motagem(a, t, e, x, n)

```
1 criar vetor f1[1 ··· n], f2[1 ··· n];
2 f1[1] ← e1 + a1,1
3 f2[1] ← e2 + a2,1
4 para k ← 2 até n faça
5 |   se f1[k − 1] ≤ f2[k − 1] + t2,k−1 então
6 |   |   f1[k] ← f1[k − 1] + a1,k
7 |   senão
  |   |
  |
  |
  |
```

Linha de Montagem - PD

Algoritmo 33: Linha-de-Motagem(a, t, e, x, n)

```
1 criar vetor f1[1 ··· n], f2[1 ··· n];
2 f1[1] ← e1 + a1,1
3 f2[1] ← e2 + a2,1
4 para k ← 2 até n faça
5     se f1[k − 1] ≤ f2[k − 1] + t2,k−1 então
6         f1[k] ← f1[k − 1] + a1,k
7     senão
8         f1[k] ← f2[k − 1] + t2,k−1 + a1,k
```

Linha de Montagem - PD

Algoritmo 34: Linha-de-Motagem(a, t, e, x, n)

```
1  criar vetor f1[1 ··· n], f2[1 ··· n];
2  f1[1] ← e1 + a1,1
3  f2[1] ← e2 + a2,1
4  para k ← 2 até n faça
5      se f1[k − 1] ≤ f2[k − 1] + t2,k−1 então
6          f1[k] ← f1[k − 1] + a1,k
7      senão
8          f1[k] ← f2[k − 1] + t2,k−1 + a1,k
9      se f2[k − 1] ≤ f1[k − 1] + t1,k−1 então
10         f2[k] ← f2[k − 1] + a2,k
11     senão
12         f2[k] ← f1[k − 1] + t1,k−1 + a2,k
```

Linha de Montagem - PD

Algoritmo 35: Linha-de-Motagem(a, t, e, x, n)

```
1  criar vetor f1[1 ··· n], f2[1 ··· n];
2  f1[1] ← e1 + a1,1
3  f2[1] ← e2 + a2,1
4  para k ← 2 até n faça
5      se f1[k − 1] ≤ f2[k − 1] + t2,k−1 então
6          f1[k] ← f1[k − 1] + a1,k
7      senão
8          f1[k] ← f2[k − 1] + t2,k−1 + a1,k
9      se f2[k − 1] ≤ f1[k − 1] + t1,k−1 então
10         f2[k] ← f2[k − 1] + a2,k
11     senão
12         f2[k] ← f1[k − 1] + t1,k−1 + a2,k
13 se f1[n] + x1 ≤ f2[n] + x2 então
```

Linha de Montagem - PD

Algoritmo 36: Linha-de-Motagem(a, t, e, x, n)

```
1  criar vetor f1[1 ··· n], f2[1 ··· n];
2  f1[1] ← e1 + a1,1
3  f2[1] ← e2 + a2,1
4  para k ← 2 até n faça
5      se f1[k − 1] ≤ f2[k − 1] + t2,k−1 então
6          f1[k] ← f1[k − 1] + a1,k
7      senão
8          f1[k] ← f2[k − 1] + t2,k−1 + a1,k
9      se f2[k − 1] ≤ f1[k − 1] + t1,k−1 então
10         f2[k] ← f2[k − 1] + a2,k
11     senão
12         f2[k] ← f1[k − 1] + t1,k−1 + a2,k
13 se f1[n] + x1 ≤ f2[n] + x2 então
14     f* ← f1[n] + x1
```

Linha de Montagem - PD

Algoritmo 37: Linha-de-Motagem(a, t, e, x, n)

```
1  criar vetor f1[1 ··· n], f2[1 ··· n];
2  f1[1] ← e1 + a1,1
3  f2[1] ← e2 + a2,1
4  para k ← 2 até n faça
5      se f1[k − 1] ≤ f2[k − 1] + t2,k−1 então
6          f1[k] ← f1[k − 1] + a1,k
7      senão
8          f1[k] ← f2[k − 1] + t2,k−1 + a1,k
9      se f2[k − 1] ≤ f1[k − 1] + t1,k−1 então
10         f2[k] ← f2[k − 1] + a2,k
11     senão
12         f2[k] ← f1[k − 1] + t1,k−1 + a2,k
13 se f1[n] + x1 ≤ f2[n] + x2 então
14     f* ← f1[n] + x1
15 senão
```

Linha de Montagem - PD

Algoritmo 38: Linha-de-Motagem(a, t, e, x, n)

```
1  criar vetor f1[1 ··· n], f2[1 ··· n];
2  f1[1] ← e1 + a1,1
3  f2[1] ← e2 + a2,1
4  para k ← 2 até n faça
5      se f1[k − 1] ≤ f2[k − 1] + t2,k−1 então
6          f1[k] ← f1[k − 1] + a1,k
7      senão
8          f1[k] ← f2[k − 1] + t2,k−1 + a1,k
9      se f2[k − 1] ≤ f1[k − 1] + t1,k−1 então
10         f2[k] ← f2[k − 1] + a2,k
11     senão
12         f2[k] ← f1[k − 1] + t1,k−1 + a2,k
13 se f1[n] + x1 ≤ f2[n] + x2 então
14     f* ← f1[n] + x1
15 senão
16     f* ← f2[n] + x2
```

Linha de Montagem - PD

Algoritmo 39: Linha-de-Motagem(a, t, e, x, n)

```
1  criar vetor f1[1 ··· n], f2[1 ··· n];
2  f1[1] ← e1 + a1,1
3  f2[1] ← e2 + a2,1
4  para k ← 2 até n faça
5      se f1[k − 1] ≤ f2[k − 1] + t2,k−1 então
6          f1[k] ← f1[k − 1] + a1,k
7      senão
8          f1[k] ← f2[k − 1] + t2,k−1 + a1,k
9      se f2[k − 1] ≤ f1[k − 1] + t1,k−1 então
10         f2[k] ← f2[k − 1] + a2,k
11     senão
12         f2[k] ← f1[k − 1] + t1,k−1 + a2,k
13 se f1[n] + x1 ≤ f2[n] + x2 então
14     f* ← f1[n] + x1
15 senão
16     f* ← f2[n] + x2
17 retorne f*
```

Linha de Montagem - PD

Algoritmo 40: Linha-de-Motagem(a, t, e, x, n)

```
1  criar vetor f_1[1 ··· n], f_2[1 ··· n], ℓ_1[1 ··· n], ℓ_2[1 ··· n];
2  f_1[1] ← e_1 + a_{1,1}; ℓ_1[1] ← 1;
3  f_2[1] ← e_2 + a_{2,1}; ℓ_2[1] ← 2;
4  para k ← 2 até n faça
5      se f_1[k − 1] ≤ f_2[k − 1] + t_{2,k−1} então
6          f_1[k] ← f_1[k − 1] + a_{1,k}; ℓ_1[k] ← 1;
7      senão
8          f_1[k] ← f_2[k − 1] + t_{2,k−1} + a_{1,k}; ℓ_1[k] ← 2;
9      se f_2[k − 1] ≤ f_1[k − 1] + t_{1,k−1} então
10         f_2[k] ← f_2[k − 1] + a_{2,k}; ℓ_2[k] ← 2;
11     senão
12         f_2[k] ← f_1[k − 1] + t_{1,k−1} + a_{2,k}; ℓ_2[k] ← 1;
13 se f_1[n] + x_1 ≤ f_2[n] + x_2 então
14     f* ← f_1[n] + x_1; ℓ* ← 1;
15 senão
16     f* ← f_2[n] + x_2; ℓ* ← 2;
17 retorne f*, ℓ_1, ℓ_2, ℓ*;
```

Linha de Montagem - PD

Linha de Montagem - PD

Algoritmo 42: Imprimir(ℓ_1, ℓ_2, ℓ^*)

Linha de Montagem - PD

Algoritmo 43: Imprimir(ℓ_1, ℓ_2, ℓ^*)

```
1  i ← ℓ*
```

Linha de Montagem - PD

Algoritmo 44: Imprimir(ℓ_1, ℓ_2, ℓ^*)

```
1 i ← ℓ*
2 escreva "linha" i "Estação" n
```

Linha de Montagem - PD

Algoritmo 45: Imprimir(ℓ_1, ℓ_2, ℓ^*)

1 $i \leftarrow \ell^*$
2 **escreva** “linha” i “Estação” n
3 **para** $j \leftarrow n$ **até** 2 **faça**

Linha de Montagem - PD

Algoritmo 46: Imprimir(ℓ_1, ℓ_2, ℓ^*)

```
1 i ← ℓ*
2 escreva "linha" i "Estação" n
3 para j ← n até 2 faça
4 |   i ← ℓ_i[j]
  |
```

Linha de Montagem - PD

Algoritmo 47: Imprimir(ℓ_1, ℓ_2, ℓ^*)

```
1 i ← ℓ*
2 escreva "linha" i "Estação" n
3 para j ← n até 2 faça
4     i ← ℓ_i[j]
5     escreva "linha" i "Estação" j − 1
```

Bibliografia

LEISERSON, C.E., STEIN, C., RIVEST, R.L., CORMEN T.H.
Algoritmos: teoria e prática, 3ed.
Editora Campus, ano 2012.

Obrigado!