

Mais programação dinâmica

CLRS 15.4

- = “recursão-com-tabela”
- = transformação inteligente de recursão em iteração

Subsequências

$\langle z_1, \dots, z_k \rangle$ é **subsequência** de $\langle x_1, \dots, x_m \rangle$
se existem índices $i_1 < \dots < i_k$ tais que

$$z_1 = x_{i_1} \quad \dots \quad z_k = x_{i_k}$$

EXEMPLOS:

$\langle 5, 9, 2, 7 \rangle$ é subsequência de $\langle 9, 5, 6, 9, 6, 2, 7, 3 \rangle$

$\langle A, A, D, A, A \rangle$ é subsequência de
 $\langle A, B, R, A, C, A, D, A, B, R, A \rangle$

A			A			D	A			A
A	B	R	A	C	A	D	A	B	R	A

Subsequência comum máxima

Z é **subsequência comum** de X e Y
se Z é *subsequência* de X e de Y

ssco = subseq comum

Subsequência comum máxima

Z é **subsequência comum** de X e Y
se Z é *subsequência* de X e de Y

ssco = subseq comum

Exemplos: $X = A \ B \ C \ B \ D \ A \ B$
 $Y = B \ D \ C \ A \ B \ A$
ssco = $B \ C \ A$

Subsequência comum máxima

Z é **subsequência comum** de X e Y
se Z é *subsequência* de X e de Y

ssco = subseq comum

Exemplos: $X = A B C B D A B$

$Y = B D C A B A$

ssco = $B C A$

Outra ssco = $B D A B$

Problema

Problema: Encontrar uma **ssco máxima** de X e Y .

Exemplos: $X = A B C B D A B$

$Y = B D C A B A$

ssco = $B C A$

ssco maximal = $A B A$

ssco máxima = $B C A B$

Outra sscó máxima = $B D A B$

LCS = Longest Common Subsequence

diff

```
> more abracadabra
```

```
A
```

```
B
```

```
R
```

```
A
```

```
C
```

```
A
```

```
D
```

```
A
```

```
B
```

```
R
```

```
A
```

```
> more yabbadabbadoo
```

```
Y
```

```
A
```

```
B
```

```
B
```

```
A
```

```
D
```

```
A
```

```
B
```

```
B
```

```
A
```

```
D
```

```
D
```

```
O
```

diff -u abracadabra yabbadabbadoo

+Y

A

B

-R

-A

-C

+B

A

D

A

B

-R

+B

A

+D

+O

+O

Subestrutura ótima

Suponha que $Z[1..k]$ é **ssco máxima** de $X[1..m]$ e $Y[1..n]$.

Subestrutura ótima

Suponha que $Z[1..k]$ é **ssco máxima** de $X[1..m]$ e $Y[1..n]$.

- ▶ Se $X[m] = Y[n]$,

Subestrutura ótima

Suponha que $Z[1..k]$ é **ssco máxima** de $X[1..m]$ e $Y[1..n]$.

- ▶ Se $X[m] = Y[n]$,
então $Z[k] = X[m] = Y[n]$ e
 $Z[1..k-1]$ é **ssco máxima** de $X[1..m-1]$ e $Y[1..n-1]$.

Subestrutura ótima

Suponha que $Z[1..k]$ é **ssco máxima** de $X[1..m]$ e $Y[1..n]$.

- ▶ Se $X[m] = Y[n]$,
então $Z[k] = X[m] = Y[n]$ e
 $Z[1..k-1]$ é ssco máxima de $X[1..m-1]$ e $Y[1..n-1]$.
- ▶ Se $X[m] \neq Y[n]$,

Subestrutura ótima

Suponha que $Z[1..k]$ é **ssco máxima** de $X[1..m]$ e $Y[1..n]$.

- ▶ Se $X[m] = Y[n]$,
então $Z[k] = X[m] = Y[n]$ e
 $Z[1..k-1]$ é ssco máxima de $X[1..m-1]$ e $Y[1..n-1]$.
- ▶ Se $X[m] \neq Y[n]$,
então $Z[k] \neq X[m]$ implica que
 $Z[1..k]$ é ssco máxima de $X[1..m-1]$ e $Y[1..n]$.

Subestrutura ótima

Suponha que $Z[1..k]$ é **ssco máxima** de $X[1..m]$ e $Y[1..n]$.

- ▶ Se $X[m] = Y[n]$,
então $Z[k] = X[m] = Y[n]$ e
 $Z[1..k-1]$ é ssc máxima de $X[1..m-1]$ e $Y[1..n-1]$.
- ▶ Se $X[m] \neq Y[n]$,
então $Z[k] \neq X[m]$ implica que
 $Z[1..k]$ é ssc máxima de $X[1..m-1]$ e $Y[1..n]$.
- ▶ Se $X[m] \neq Y[n]$,

Subestrutura ótima

Suponha que $Z[1..k]$ é sscó máxima de $X[1..m]$ e $Y[1..n]$.

- ▶ Se $X[m] = Y[n]$,
então $Z[k] = X[m] = Y[n]$ e
 $Z[1..k-1]$ é sscó máxima de $X[1..m-1]$ e $Y[1..n-1]$.
- ▶ Se $X[m] \neq Y[n]$,
então $Z[k] \neq X[m]$ implica que
 $Z[1..k]$ é sscó máxima de $X[1..m-1]$ e $Y[1..n]$.
- ▶ Se $X[m] \neq Y[n]$,
então $Z[k] \neq Y[n]$ implica que
 $Z[1..k]$ é sscó máxima de $X[1..m]$ e $Y[1..n-1]$.

Simplificação

Problema: encontrar o comprimento de uma sscv máxima.

Simplificação

Problema: encontrar o **comprimento** de uma sscó máxima.

$c[i,j]$ = comprimento de uma sscó máxima
de $X[1..i]$ e $Y[1..j]$

Simplificação

Problema: encontrar o **comprimento** de uma sscó máxima.

$c[i, j]$ = comprimento de uma sscó máxima
de $X[1..i]$ e $Y[1..j]$

Recorrência:

$$c[0, j] = c[i, 0] = 0$$

Simplificação

Problema: encontrar o **comprimento** de uma sscó máxima.

$c[i, j]$ = comprimento de uma sscó máxima
de $X[1..i]$ e $Y[1..j]$

Recorrência:

$$c[0, j] = c[i, 0] = 0$$

$$c[i, j] = c[i-1, j-1] + 1 \text{ se } X[i] = Y[j]$$

Simplificação

Problema: encontrar o **comprimento** de uma sscó máxima.

$c[i,j]$ = comprimento de uma sscó máxima
de $X[1..i]$ e $Y[1..j]$

Recorrência:

$$c[0,j] = c[i,0] = 0$$

$$c[i,j] = c[i-1,j-1] + 1 \text{ se } X[i] = Y[j]$$

$$c[i,j] = \max(c[i,j-1], c[i-1,j]) \text{ se } X[i] \neq Y[j]$$

Algoritmo recursivo

Devolve o comprimento de uma sscó máxima de $X[1..i]$ e $Y[1..j]$.

REC-LCS-LENGTH (X, i, Y, j)

1 se $i = 0$ ou $j = 0$ então devolva 0

Algoritmo recursivo

Devolve o comprimento de uma sscó máxima de $X[1..i]$ e $Y[1..j]$.

REC-LCS-LENGTH (X, i, Y, j)

1 se $i = 0$ ou $j = 0$ então devolva 0

2 se $X[i] = Y[j]$

3 então $c \leftarrow$ REC-LCS-LENGTH ($X, i-1, Y, j-1$) + 1

Algoritmo recursivo

Devolve o comprimento de uma sscó máxima de $X[1..i]$ e $Y[1..j]$.

REC-LCS-LENGTH (X, i, Y, j)

1 se $i = 0$ ou $j = 0$ então devolva 0

2 se $X[i] = Y[j]$

3 então $c \leftarrow$ REC-LCS-LENGTH ($X, i-1, Y, j-1$) + 1

4 senão $q_1 \leftarrow$ REC-LCS-LENGTH ($X, i-1, Y, j$)

5 $q_2 \leftarrow$ REC-LCS-LENGTH ($X, i, Y, j-1$)

Algoritmo recursivo

Devolve o comprimento de uma sscó máxima de $X[1..i]$ e $Y[1..j]$.

REC-LCS-LENGTH (X, i, Y, j)

```
1  se  $i = 0$  ou  $j = 0$  então devolva 0
2  se  $X[i] = Y[j]$ 
3      então  $c \leftarrow \text{REC-LCS-LENGTH}(X, i-1, Y, j-1) + 1$ 
4      senão  $q_1 \leftarrow \text{REC-LCS-LENGTH}(X, i-1, Y, j)$ 
5            $q_2 \leftarrow \text{REC-LCS-LENGTH}(X, i, Y, j-1)$ 
6           se  $q_1 \geq q_2$ 
7               então  $c \leftarrow q_1$ 
8               senão  $c \leftarrow q_2$ 
9  devolva  $c$ 
```

Algoritmo recursivo

Devolve o comprimento de uma sscó máxima de $X[1..i]$ e $Y[1..j]$.

REC-LCS-LENGTH (X, i, Y, j)

```
1  se  $i = 0$  ou  $j = 0$  então devolva 0
2  se  $X[i] = Y[j]$ 
3      então  $c \leftarrow \text{REC-LCS-LENGTH}(X, i-1, Y, j-1) + 1$ 
4      senão  $q_1 \leftarrow \text{REC-LCS-LENGTH}(X, i-1, Y, j)$ 
5            $q_2 \leftarrow \text{REC-LCS-LENGTH}(X, i, Y, j-1)$ 
6           se  $q_1 \geq q_2$ 
7               então  $c \leftarrow q_1$ 
8               senão  $c \leftarrow q_2$ 
9  devolva  $c$ 
```

$T(m, n) :=$ número de comparações feitas por
REC-LCS-LENGTH (X, m, Y, n) no pior caso

Consumo de tempo

$T(m, n)$:= número de comparações feitas por
REC-LCS-LENGTH (X, m, Y, n) no pior caso

Consumo de tempo

$T(m, n)$:= número de comparações feitas por
REC-LCS-LENGTH (X, m, Y, n) no pior caso

Recorrência

$$T(0, n) = 0$$

$$T(m, 0) = 0$$

$$T(m, n) \geq T(m-1, n) + T(m, n-1) + 1 \quad \text{para } n \geq 1 \text{ e } m \geq 1$$

Consumo de tempo

$T(m, n)$:= número de comparações feitas por
REC-LCS-LENGTH (X, m, Y, n) no pior caso

Recorrência

$$T(0, n) = 0$$

$$T(m, 0) = 0$$

$$T(m, n) \geq T(m-1, n) + T(m, n-1) + 1 \quad \text{para } n \geq 1 \text{ e } m \geq 1$$

A que classe Ω pertence $T(m, n)$?

Recorrência

Note que $T(m, n) = T(n, m)$ para $n = 0, 1, \dots$ e $m = 0, 1, \dots$

Recorrência

Note que $T(m, n) = T(n, m)$ para $n = 0, 1, \dots$ e $m = 0, 1, \dots$

Seja $k := \min\{m, n\}$. Temos que

$$T(m, n) \geq T(k, k) \geq S(k),$$

onde

$$S(0) = 0$$

$$S(k) = 2S(k-1) + 1 \quad \text{para } k = 1, 2, \dots$$

Recorrência

Note que $T(m, n) = T(n, m)$ para $n = 0, 1, \dots$ e $m = 0, 1, \dots$

Seja $k := \min\{m, n\}$. Temos que

$$T(m, n) \geq T(k, k) \geq S(k),$$

onde

$$S(0) = 0$$

$$S(k) = 2S(k-1) + 1 \quad \text{para } k = 1, 2, \dots$$

$S(k)$ é $\Theta(2^k) \Rightarrow T(m, n)$ é $\Omega(2^{\min\{m, n\}})$.

Recorrência

Note que $T(m, n) = T(n, m)$ para $n = 0, 1, \dots$ e $m = 0, 1, \dots$

Seja $k := \min\{m, n\}$. Temos que

$$T(m, n) \geq T(k, k) \geq S(k),$$

onde

$$S(0) = 0$$

$$S(k) = 2S(k-1) + 1 \quad \text{para } k = 1, 2, \dots$$

$S(k)$ é $\Theta(2^k)$ \Rightarrow $T(m, n)$ é $\Omega(2^{\min\{m, n\}})$.

$T(m, n)$ é exponencial.

Conclusão

O consumo de tempo do algoritmo `REC-LCS-LENGTH` é $\Omega(2^{\min\{m,n\}})$.

Programação dinâmica

Cada subproblema, comprimento de uma sscó máxima de

$$X[1..i] \text{ e } Y[1..j],$$

é resolvido **uma só** vez.

Em que ordem calcular os componentes da tabela c ?

Para calcular $c[4,6]$ preciso de ...?

Programação dinâmica

	1	2	3	4	5	6	7	8	j
1	0	0	0	0	0	0	0	0	
2	0								
3	0				*	*			
4	0				*	??			
5	0								
6	0								
7	0								
8	0								

Programação dinâmica

Cada subproblema, comprimento de uma sscó máxima de

$$X[1..i] \text{ e } Y[1..j],$$

é resolvido **uma só** vez.

Em que ordem calcular os componentes da tabela c ?

Para calcular $c[4,6]$ preciso de ...

$c[4,5]$, $c[3,6]$ e de $c[3,5]$.

Programação dinâmica

Cada subproblema, comprimento de uma sscó máxima de

$$X[1..i] \text{ e } Y[1..j],$$

é resolvido **uma só** vez.

Em que ordem calcular os componentes da tabela c ?

Para calcular $c[4,6]$ preciso de ...

$c[4,5]$, $c[3,6]$ e de $c[3,5]$.

Calcule todos os $c[i,j]$ com $i = 1$ e $j = 0, 1, \dots, n$,
depois todos com $i = 2$ e $j = 0, 1, \dots, n$,
depois todos com $i = 3$ e $j = 0, 1, \dots, n$,
etc.

Simulação

X	Y		B	D	C	A	B	A	j
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	??						
B	2	0							
C	3	0							
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	??					
B	2	0							
C	3	0							
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

	<i>Y</i>		B	D	<i>C</i>	A	B	A	
<i>X</i>		0	1	2	<i>3</i>	4	5	6	<i>j</i>
	0	0	0	0	0	0	0	0	
<i>A</i>	1	0	0	0	??				
<i>B</i>	2	0							
<i>C</i>	3	0							
<i>B</i>	4	0							
<i>D</i>	5	0							
<i>A</i>	6	0							
<i>B</i>	7	0							

Simulação

	<i>Y</i>		B	D	C	A	B	A	
<i>X</i>		0	1	2	3	4	5	6	<i>j</i>
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	??			
B	2	0							
C	3	0							
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

	<i>Y</i>		B	D	C	A	B	A	
<i>X</i>		0	1	2	3	4	5	6	<i>j</i>
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	??		
B	2	0							
C	3	0							
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	??	
B	2	0							
C	3	0							
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

	<i>Y</i>		<i>B</i>	<i>D</i>	<i>C</i>	<i>A</i>	<i>B</i>	<i>A</i>	
<i>X</i>		0	1	2	3	4	5	6	<i>j</i>
	0	0	0	0	0	0	0	0	
<i>A</i>	1	0	0	0	0	1	1	1	
<i>B</i>	2	0	??						
<i>C</i>	3	0							
<i>B</i>	4	0							
<i>D</i>	5	0							
<i>A</i>	6	0							
<i>B</i>	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	??					
C	3	0							
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

	<i>Y</i>		B	D	<i>C</i>	A	B	A	
<i>X</i>		0	1	2	<i>3</i>	4	5	6	<i>j</i>
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	??				
C	3	0							
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	??			
C	3	0							
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	??		
C	3	0							
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	??	
C	3	0							
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

	<i>Y</i>		B	D	C	A	B	A	
<i>X</i>		0	1	2	3	4	5	6	<i>j</i>
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	??						
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	??					
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	??				
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	??			
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	??		
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	??	
B	4	0							
D	5	0							
A	6	0							
B	7	0							

Simulação

	<i>Y</i>		<i>B</i>	<i>D</i>	<i>C</i>	<i>A</i>	<i>B</i>	<i>A</i>	
<i>X</i>		0	1	2	3	4	5	6	<i>j</i>
	0	0	0	0	0	0	0	0	
<i>A</i>	1	0	0	0	0	1	1	1	
<i>B</i>	2	0	1	1	1	1	2	2	
<i>C</i>	3	0	1	1	2	2	2	2	
<i>B</i>	4	0	??						
<i>D</i>	5	0							
<i>A</i>	6	0							
<i>B</i>	7	0							

Simulação

	<i>Y</i>		B	D	C	A	B	A	
<i>X</i>		0	1	2	3	4	5	6	<i>j</i>
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	??					
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	??				
D	5	0							
A	6	0							
B	7	0							

Simulação

X	Y		B	D	C	A	B	A	j
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	??			
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	??		
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	??	
D	5	0							
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>	0	B 1	D 2	C 3	A 4	B 5	A 6	<i>j</i>
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	??						
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	??					
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	??				
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	??			
A	6	0							
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	??		
A	6	0							
B	7	0							

Simulação

X	Y	0	B	D	C	A	B	A	j
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	??	
A	6	0							
B	7	0							

Simulação

X	Y		B	D	C	A	B	A	j
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	??						
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	??					
B	7	0							

Simulação

	<i>Y</i>		B	D	<i>C</i>	A	B	A	
<i>X</i>		0	1	2	<i>3</i>	4	5	6	<i>j</i>
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	2	??				
B	7	0							

Simulação

X	Y		B	D	C	A	B	A	j
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	2	2	??			
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	2	2	3	??		
B	7	0							

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	2	2	3	3	??	
B	7	0							

Simulação

X	Y		B	D	C	A	B	A	j
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	2	2	3	3	4	
B	7	0	??						

Simulação

X	Y		B	D	C	A	B	A	j
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	2	2	3	3	4	
B	7	0	1	??					

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	2	2	3	3	4	
B	7	0	1	2	??				

Simulação

X	Y	0	B	D	C	A	B	A	j
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	2	2	3	3	4	
B	7	0	1	2	2	??			

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	2	2	3	3	4	
B	7	0	1	2	2	3	??		

Simulação

X	Y		B	D	C	A	B	A	j
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	2	2	3	3	4	
B	7	0	1	2	2	3	4	??	

Simulação

<i>X</i>	<i>Y</i>		B	D	C	A	B	A	<i>j</i>
		0	1	2	3	4	5	6	
	0	0	0	0	0	0	0	0	
A	1	0	0	0	0	1	1	1	
B	2	0	1	1	1	1	2	2	
C	3	0	1	1	2	2	2	2	
B	4	0	1	1	2	2	3	3	
D	5	0	1	2	2	2	3	3	
A	6	0	1	2	2	3	3	4	
B	7	0	1	2	2	3	4	4	

Algoritmo de programação dinâmica

Devolve o comprimento de
uma sscó máxima de $X[1..m]$ e $Y[1..n]$.

LCS-LENGTH (X, m, Y, n)

- 1 **para** $i \leftarrow 0$ **até** m **faça** $c[i, 0] \leftarrow 0$
- 2 **para** $j \leftarrow 1$ **até** n **faça** $c[0, j] \leftarrow 0$

Algoritmo de programação dinâmica

Devolve o comprimento de
uma sscó máxima de $X[1..m]$ e $Y[1..n]$.

LCS-LENGTH (X, m, Y, n)

- 1 para $i \leftarrow 0$ até m faça $c[i, 0] \leftarrow 0$
- 2 para $j \leftarrow 1$ até n faça $c[0, j] \leftarrow 0$
- 3 para $i \leftarrow 1$ até m faça
- 4 para $j \leftarrow 1$ até n faça

Algoritmo de programação dinâmica

Devolve o comprimento de
uma sscó máxima de $X[1..m]$ e $Y[1..n]$.

LCS-LENGTH (X, m, Y, n)

```
1 para  $i \leftarrow 0$  até  $m$  faça  $c[i, 0] \leftarrow 0$ 
2 para  $j \leftarrow 1$  até  $n$  faça  $c[0, j] \leftarrow 0$ 
3 para  $i \leftarrow 1$  até  $m$  faça
4     para  $j \leftarrow 1$  até  $n$  faça
5         se  $X[i] = Y[j]$ 
6             então  $c[i, j] \leftarrow c[i - 1, j - 1] + 1$ 
```

Algoritmo de programação dinâmica

Devolve o comprimento de
uma sscó máxima de $X[1..m]$ e $Y[1..n]$.

LCS-LENGTH (X, m, Y, n)

```
1  para  $i \leftarrow 0$  até  $m$  faça  $c[i, 0] \leftarrow 0$ 
2  para  $j \leftarrow 1$  até  $n$  faça  $c[0, j] \leftarrow 0$ 
3  para  $i \leftarrow 1$  até  $m$  faça
4      para  $j \leftarrow 1$  até  $n$  faça
5          se  $X[i] = Y[j]$ 
6              então  $c[i, j] \leftarrow c[i - 1, j - 1] + 1$ 
7              senão se  $c[i - 1, j] \geq c[i, j - 1]$ 
8                  então  $c[i, j] \leftarrow c[i - 1, j]$ 
9                  senão  $c[i, j] \leftarrow c[i, j - 1]$ 
10 devolva  $c[m, n]$ 
```

Algoritmo de programação dinâmica

Devolve o comprimento de
uma sscó máxima de $X[1..m]$ e $Y[1..n]$.

LCS-LENGTH (X, m, Y, n)

```
1  para  $i \leftarrow 0$  até  $m$  faça  $c[i, 0] \leftarrow 0$ 
2  para  $j \leftarrow 1$  até  $n$  faça  $c[0, j] \leftarrow 0$ 
3  para  $i \leftarrow 1$  até  $m$  faça
4      para  $j \leftarrow 1$  até  $n$  faça
5          se  $X[i] = Y[j]$ 
6              então  $c[i, j] \leftarrow c[i - 1, j - 1] + 1$ 
7              senão se  $c[i - 1, j] \geq c[i, j - 1]$ 
8                  então  $c[i, j] \leftarrow c[i - 1, j]$ 
9                  senão  $c[i, j] \leftarrow c[i, j - 1]$ 
10 devolva  $c[m, n]$ 
```

Consumo de tempo: $\Theta(mn)$

Conclusão

O consumo de tempo do algoritmo `LCS-LENGTH` é $\Theta(mn)$.

Conclusão

O consumo de tempo do algoritmo `LCS-LENGTH` é $\Theta(mn)$.

E como encontrar uma subsequência comum máxima?

Subsequência comum máxima

		Y						
		0	B	D	C	A	B	A
X		0	1	2	3	4	5	6
	0	*	*	*	*	*	*	*
A	1	*	←	←	←	↖	↑	↖
B	2	*	↖	↑	↑	←	↖	↑
C	3	*	←	←	↖	↑	←	←
B	4	*	↖	←	←	←	↖	↑
D	5	*	←	↖	←	←	←	←
A	6	*	←	←	←	↖	←	↖
B	7	*	↖	←	←	←	↖	←

Algoritmo de programação dinâmica

LCS-LENGTH (X, m, Y, n)

```
1  para  $i \leftarrow 0$  até  $m$  faça  $c[i, 0] \leftarrow 0$ 
2  para  $j \leftarrow 1$  até  $n$  faça  $c[0, j] \leftarrow 0$ 
3  para  $i \leftarrow 1$  até  $m$  faça
4      para  $j \leftarrow 1$  até  $n$  faça
5          se  $X[i] = Y[j]$ 
6              então  $c[i, j] \leftarrow c[i - 1, j - 1] + 1$ 
7                   $b[i, j] \leftarrow "$  $"$ 
8              senão se  $c[i - 1, j] \geq c[i, j - 1]$ 
9                  então  $c[i, j] \leftarrow c[i - 1, j]$ 
10                      $b[i, j] \leftarrow "$  $"$ 
11                 senão  $c[i, j] \leftarrow c[i, j - 1]$ 
12                      $b[i, j] \leftarrow "$  $"$ 
13  devolva  $c$  e  $b$ 
```

Algoritmo de programação dinâmica

LCS-LENGTH (X, m, Y, n)

```
1  para  $i \leftarrow 0$  até  $m$  faça  $c[i, 0] \leftarrow 0$ 
2  para  $j \leftarrow 1$  até  $n$  faça  $c[0, j] \leftarrow 0$ 
3  para  $i \leftarrow 1$  até  $m$  faça
4      para  $j \leftarrow 1$  até  $n$  faça
5          se  $X[i] = Y[j]$ 
6              então  $c[i, j] \leftarrow c[i - 1, j - 1] + 1$ 
7                   $b[i, j] \leftarrow \text{"↖"}$ 
8              senão se  $c[i - 1, j] \geq c[i, j - 1]$ 
9                  então  $c[i, j] \leftarrow c[i - 1, j]$ 
10                      $b[i, j] \leftarrow \text{"↑"}$ 
11                 senão  $c[i, j] \leftarrow c[i, j - 1]$ 
12                     $b[i, j] \leftarrow \text{"←"}$ 
13 devolva  $c$  e  $b$ 
```

Consumo de tempo: $\Theta(mn)$

Algoritmo de programação dinâmica

LCS-LENGTH (X, m, Y, n)

```
1  para  $i \leftarrow 0$  até  $m$  faça  $c[i, 0] \leftarrow 0$ 
2  para  $j \leftarrow 1$  até  $n$  faça  $c[0, j] \leftarrow 0$ 
3  para  $i \leftarrow 1$  até  $m$  faça
4      para  $j \leftarrow 1$  até  $n$  faça
5          se  $X[i] = Y[j]$ 
6              então  $c[i, j] \leftarrow c[i - 1, j - 1] + 1$ 
7                   $b[i, j] \leftarrow "$  $"$ 
8              senão se  $c[i - 1, j] \geq c[i, j - 1]$ 
9                  então  $c[i, j] \leftarrow c[i - 1, j]$ 
10                      $b[i, j] \leftarrow "$  $"$ 
11                 senão  $c[i, j] \leftarrow c[i, j - 1]$ 
12                      $b[i, j] \leftarrow "$  $"$ 
13  devolva  $c$  e  $b$ 
```

Como obter uma sscó máxima a partir da matriz b ?

Get-LCS

Como obter uma sscó máxima a partir da matriz b ?

GET-LCS ($X, m, n, b, \text{máxcomp}$)

```
1   $k \leftarrow \text{máxcomp}$ 
2   $i \leftarrow m$      $j \leftarrow n$ 
3  enquanto  $i > 0$  e  $j > 0$  faça
4      se  $b[i, j] = \swarrow$ 
5          então  $Z[k] \leftarrow X[i]$ 
6               $k \leftarrow k - 1$      $i \leftarrow i - 1$      $j \leftarrow j - 1$ 
7      senão se  $b[i, j] = \leftarrow$ 
8          então  $i \leftarrow i - 1$ 
9          senão  $j \leftarrow j - 1$ 
10 devolva  $Z$ 
```

Get-LCS

Como obter uma sscó máxima a partir da matriz b ?

GET-LCS ($X, m, n, b, \text{máxcomp}$)

```
1  $k \leftarrow \text{máxcomp}$ 
2  $i \leftarrow m$       $j \leftarrow n$ 
3 enquanto  $i > 0$  e  $j > 0$  faça
4     se  $b[i, j] = \nwarrow$ 
5         então  $Z[k] \leftarrow X[i]$ 
6              $k \leftarrow k - 1$       $i \leftarrow i - 1$       $j \leftarrow j - 1$ 
7     senão se  $b[i, j] = \leftarrow$ 
8         então  $i \leftarrow i - 1$ 
9         senão  $j \leftarrow j - 1$ 
10 devolva  $Z$ 
```

Consumo de tempo é $O(m + n)$ e $\Omega(\min\{m, n\})$.

Exercícios

Exercício 20.A

Escreva um algoritmo para decidir se $\langle z_1, \dots, z_k \rangle$ é subsequência de $\langle x_1, \dots, x_m \rangle$. Prove rigorosamente que o seu algoritmo está correto.

Exercício 20.B

Suponha que os elementos de uma sequência $\langle a_1, \dots, a_n \rangle$ são distintos dois a dois. Quantas subsequências tem a sequência?

Exercício 20.C

Uma subsequência crescente Z de uma sequência X é *máxima* se não existe outra subsequência crescente mais longa. A subsequência $\langle 5, 6, 9 \rangle$ de $\langle 9, 5, 6, 9, 6, 2, 7 \rangle$ é máxima? Dê uma sequência crescente máxima de $\langle 9, 5, 6, 9, 6, 2, 7 \rangle$. Mostre que o algoritmo “guloso” óbvio não é capaz, em geral, de encontrar uma subsequência crescente máxima de uma sequência dada. (Algoritmo guloso óbvio: escolha o menor elemento de X ; a partir daí, escolha sempre o próximo elemento de X que seja maior ou igual ao último escolhido.)

Exercício 20.D

Escreva um algoritmo de programação dinâmica para resolver o problema da subsequência crescente máxima.

Exercício das bandeiras

No dia da Bandeira na Rússia o proprietário de uma loja decidiu decorar a vitrine de sua loja com faixas de tecido das cores branca, azul e vermelha.

Ele deseja satisfazer as seguintes condições: faixas da mesma cor não podem ser colocadas uma ao lado da outra. Uma faixa azul sempre está entre uma branca e uma vermelha, ou uma vermelha e uma branca.

Escreva um programa que, dado o número n de faixas a serem colocadas na vitrine, calcule o número de maneiras de satisfazer as condições do proprietário.

Exemplo: Para $n = 3$, o resultado são as seguintes combinações: BVB, VBV, BAV, VAB.

Mais exercícios

Exercício 20.E [CLRS 15.4-5]

Mostre como o algoritmo da subsequência comum máxima pode ser usado para resolver o problema da subsequência crescente máxima de uma sequência numérica. Dê uma delimitação justa, em notação Θ , do consumo de tempo de sua solução.

Exercício 20.F [Printing neatly. CLRS 15-2]

Considere a sequência P_1, P_2, \dots, P_n de palavras que constitui um parágrafo de texto. A palavra P_i tem l_i caracteres. Queremos imprimir as palavras em linhas, na ordem dada, de modo que cada linha tenha no máximo M caracteres. Se uma determinada linha contém as palavras P_i, P_{i+1}, \dots, P_j (com $i \leq j$) e há exatamente um espaço entre cada par de palavras consecutivas, o número de espaços no fim da linha é

$$M - (l_i + 1 + l_{i+1} + 1 + \dots + 1 + l_j).$$

É claro que não devemos permitir que esse número seja negativo. Queremos minimizar, com relação a todas as linhas exceto a última, a soma dos cubos dos números de espaços no fim de cada linha. (Assim, se temos linhas $1, 2, \dots, L$ e b_p espaços no fim da linha p , queremos minimizar $b_1^3 + b_2^3 + \dots + b_{L-1}^3$.)

Dê um exemplo para mostrar que algoritmos inocentes não resolvem o problema. Dê um algoritmo de programação dinâmica que resolva o problema. Qual a “optimal substructure property” para esse problema? Faça uma análise do consumo de tempo do algoritmo.