Mais programação dinâmica

CLRS 15.4 e 15.5

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

Subseqüências

$\langle z_1, \ldots, z_k \rangle$ é subseqüência de $\langle x_1, \ldots, x_m \rangle$
se existem índices $i_1 < \cdots < i_k$ tais que

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

EXEMPLOS:

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

$\langle$A, A, D, A, A$\rangle$ é subseqüência 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

Exercício

Problema: Decidir se $Z[1\,..\,m]$ é subseqüência de $X[1\,..\,n]$

Exercício

Problema: Decidir se $Z[1\,.\,.\,m]$ é subseqüência de $X[1\,.\,.\,n]$

SUB-SEQ- (Z, m, X, n)

$i \leftarrow m$
$j \leftarrow n$
enquanto $i \geq 1$ **e** $j \geq 1$ **faça**
    se $Z[i] = X[j]$
        então $i \leftarrow i - 1$
    $j \leftarrow j - 1$
se $i \geq 1$
    então devolva "`não é subseqüência`"
    senão devolva "`é subseqüência`"

Exercício

Problema: Decidir se $Z[1\,..\,m]$ é subseqüência de $X[1\,..\,n]$

SUB-SEQ- (Z, m, X, n)

```
i ← m
j ← n
enquanto i ≥ 1 e j ≥ 1 faça
    se Z[i] = X[j]
        então i ← i − 1
    j ← j − 1
se i ≥ 1
    então devolva "não é subseqüência"
    senão devolva "é subseqüência"
```

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

Exercício

Problema: Decidir se $Z[1\,..\,m]$ é subseqüência de $X[1\,..\,n]$

```
SUB-SEQ- (Z, m, X, n)
    i ← m
    j ← n
    enquanto i ≥ 1 e j ≥ 1 faça
        se Z[i] = X[j]
            então i ← i − 1
        j ← j − 1
    se i ≥ 1
        então devolva "não é subseqüência"
        senão devolva "é subseqüência"
```

Invariantes:

(i0) $Z[i{+}1\,..\,m]$ é subseqüência de $X[j{+}1\,..\,n]$

(i1) $Z[i\,..\,m]$ **não** é subseqüência de $X[j{+}1\,..\,n]$

Subseqüência comum máxima

Z é subseq comum de X e Y

se Z é subseqüência 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 ssco 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
+D
+O
```

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

Simplificação

Problema: encontrar o comprimento de uma ssco máxima.

Simplificação

Problema: encontrar o comprimento de uma ssco máxima.

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

Recorrência:

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

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

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

Algoritmo recursivo

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

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

```
se i = 0 ou j = 0
    então devolva 0
se X[i] = Y[j]
    então c[i, j] ← REC-LCS-LENGTH (X, i−1, Y, j−1) + 1
    senão q1 ← REC-LCS-LENGTH (X, i−1, Y, j)
          q2 ← REC-LCS-LENGTH (X, i, Y, j−1)
          se q1 ≥ q2
              então c[i, j] ← q1
              senãoc[i, j] ← q2
devolva c[i, j]
```

Consumo de tempo

$T(m, n) :=$ número de comparações feitas por
REC-LCS-LENGTH (X, m, Y, n)

Recorrência

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

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

$$T(m, n) \leq T(m-1, n) + T(m, n-1) + 1 \text{ para } n \geq 0 \text{ e } m \geq 0$$

A que classe Ω pertence $T(m, n)$?

Recorrência

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

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

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

onde

$$S(0) = 0$$
$$S(k) = 2S(k-1) + 1 \quad \text{para } k = 1, 2, \ldots$$

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

$T(m, n)$ é exponecial

Conclusão

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

Programação dinâmica

Cada subproblema, comprimento de uma ssco máxima de

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

é resolvido uma só vez.

Em que ordem calcular os componentes da tabela c?

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

Programação dinâmica

Cada subproblema, comprimento de uma ssco máxima de

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

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 ssco máxima de

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

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 $i = 1, j = 0, 1, \ldots, n$,
depois todos com $i = 2, j = 0, 1, \ldots, n$,
depois todos com $i = 3, j = 0, 1, \ldots, n$,
etc.

Programação dinâmica

	1	2	3	4	5	6	7	8	j
1	0	0	0	0	0	0	0	0	
2	0								
3	0				$\star$	$\star$			
4	0				$\star$	??			
5	0								
6	0								
7	0								
8	0								
i									

Simulação

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

Simulação

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

i

Simulação

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

Simulação

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

i

Simulação

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

Simulação

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

Simulação

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

Simulação

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

Simulação

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

i

Simulação

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

Simulação

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

Simulação

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

Simulação

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

Simulação

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

i

Simulação

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

Simulação

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

Simulação

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

Simulação

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

Simulação

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

i

Simulação

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

i

Simulação

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

i

Simulação

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

Simulação

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

Simulação

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

i

Simulação

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

Simulação

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

Simulação

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

Simulação

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

i

Simulação

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

Simulação

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

i

Simulação

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

i

Simulação

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

i

Simulação

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

Simulação

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

Simulação

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

Simulação

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

Simulação

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

Simulação

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

Simulação

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

Simulação

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

Simulação

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

Simulação

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

Simulação

	Y		B	D	C	A	B	A	
X		0	1	2	3	4	5	6	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	3	4	4	
	i								

Algoritmo de programação dinâmica

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

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

para $i \leftarrow 0$ **até** m **faça**
    $c[i, 0] \leftarrow 0$
para $j \leftarrow 1$ **até** n **faça**
    $c[0, j] \leftarrow 0$
para $i \leftarrow 1$ **até** m **faça**
    para $j \leftarrow 1$ **até** n **faça**
        se $X[i] = Y[j]$
            então $c[i, j] \leftarrow c[i-1, j-1] + 1$
            senão se $c[i-1, j] \geq c[i, j-1]$
                    então $c[i, j] \leftarrow c[i-1, j]$
                    senão $c[i, j] \leftarrow c[i, j-1]$
devolva $c[m, n]$

Conclusão

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

Subseqüência comum máxima

	Y		B	D	C	A	B	A	
X		0	1	2	3	4	5	6	j
	0	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	
A	1	$\star$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\nwarrow$	$\uparrow$	$\nwarrow$	
B	2	$\star$	$\nwarrow$	$\uparrow$	$\uparrow$	$\leftarrow$	$\nwarrow$	$\uparrow$	
C	3	$\star$	$\leftarrow$	$\leftarrow$	$\nwarrow$	$\uparrow$	$\leftarrow$	$\leftarrow$	
B	4	$\star$	$\nwarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\nwarrow$	$\uparrow$	
D	5	$\star$	$\leftarrow$	$\nwarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	
A	6	$\star$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\nwarrow$	$\leftarrow$	$\nwarrow$	
B	7	$\star$	$\nwarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\nwarrow$	$\leftarrow$	
	i								

Algoritmo de programação dinâmica

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

para $i \leftarrow 0$ **até** m **faça**
  $c[i, 0] \leftarrow 0$
para $j \leftarrow 1$ **até** n **faça**
  $c[0, j] \leftarrow 0$
para $i \leftarrow 1$ **até** m **faça**
  para $j \leftarrow 1$ **até** n **faça**
    se $X[i] = Y[j]$
      então $c[i, j] \leftarrow c[i-1, j-1] + 1$
        $b[i, j] \leftarrow$ "$\nwarrow$"
      senão se $c[i-1, j] \geq c[i, j-1]$
        então $c[i, j] \leftarrow c[i-1, j]$
          $b[i, j] \leftarrow$ "$\uparrow$"
        senão $c[i, j] \leftarrow c[i, j-1]$
          $b[i, j] \leftarrow$ "$\leftarrow$"
devolva c e b

Get-LCS

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

$k \leftarrow$ máxcomp
$i \leftarrow m$
$j \leftarrow n$
enquanto $i > 0$ **e** $j > 0$ **faça**
    se $b[i, j] =$ "$\nwarrow$"
        então $Z[k] \leftarrow X[i]$
            $k \leftarrow k - 1 \quad i \leftarrow i - 1 \quad j \leftarrow j - 1$
        senão se $b[i, j] =$ "$\leftarrow$"
            então $i \leftarrow i - 1$
            senão $j \leftarrow j - 1$
devolva Z

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

Exercícios

Exercício 20.A

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

Exercício 20.B

Suponha que os elementos de uma seqüência $\langle a_1, \ldots, a_n\rangle$ são distintos dois a dois. Quantas subseqüências tem a seqüência?

Exercício 20.C

Uma subseqüência crescente Z de uma seqüência X e é *máxima* se não existe outra subseqüência crescente mais longa. A subseqüência $\langle 5, 6, 9\rangle$ de $\langle 9, 5, 6, 9, 6, 2, 7\rangle$ é máxima? Dê uma seqüência 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 subseqüência crescente máxima de uma seqüência 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 subseqüência crescente máxima.

Mais exercícios

Exercício 20.E [CLRS 15.4-5]
Mostre como o algoritmo da subseqüência comum máxima pode ser usado para resolver o problema da subseqüência crescente máxima de uma seqüência 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 seqüência $P_1, P_2, \ldots, P_n$ 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 $P_i, P_{i+1}, \ldots, P_j$ (com $i \leq j$) e há exatamente um espaço entre cada par de palavras consecutivas, o número de espaços no fim da linha é

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

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 $1, 2, \ldots, L$ e b_p espaços no fim da linha p, queremos minimizar $b_1^3 + b_2^3 + \cdots + b_{L-1}^3$).
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.

Buscas em um conjunto conhecido

Considere um inteiro n e um vetor $v[1\,..\,n]$ de inteiros.

Problema: Dado $v[1\,..\,n]$ e uma sequência de k inteiros, decidir se cada inteiro está ou não em v.

Buscas em um conjunto conhecido

Considere um inteiro n e um vetor $v[1\,..\,n]$ de inteiros.

Problema: Dado $v[1\,..\,n]$ e uma sequência de k inteiros, decidir se cada inteiro está ou não em v.

Se k é grande, como devemos armazenar o v?

Buscas em um conjunto conhecido

Considere um inteiro n e um vetor $v[1\,..\,n]$ de inteiros.

Problema: Dado $v[1\,..\,n]$ e uma sequência de k inteiros, decidir se cada inteiro está ou não em v.

Se k é grande, como devemos armazenar o v?

E se v armazena um conjunto bem conhecido,
como por exemplo as palavras de uma língua?
(A ser usado por um tradutor, ou um speller.)

Buscas em um conjunto conhecido

Considere um inteiro n e um vetor $v[1\,.\,.\,n]$ de inteiros.

Problema: Dado $v[1\,.\,.\,n]$ e uma sequência de k inteiros, decidir se cada inteiro está ou não em v.

Se k é grande, como devemos armazenar o v?

E se v armazena um conjunto bem conhecido, como por exemplo as palavras de uma língua?
(A ser usado por um tradutor, ou um speller.)

Podemos fazer algo melhor?

Buscas em conjunto conhecido

Dadas estimativas do número de acessos a cada elemento de $v[1\,..\,n]$, qual é a melhor estrutura de dados para v?

Buscas em conjunto conhecido

Dadas estimativas do número de acessos a cada elemento de $v[1\,.\,.\,n]$, qual é a melhor estrutura de dados para v?

Árvore de busca binária (ABB)?

Buscas em conjunto conhecido

Dadas estimativas do número de acessos a cada elemento de $v[1\,.\,.\,n]$, qual é a melhor estrutura de dados para v?

Árvore de busca binária (ABB)?

Exemplo: $n = 3$ e $e_1 = 10$, $e_2 = 20$, $e_3 = 40$.

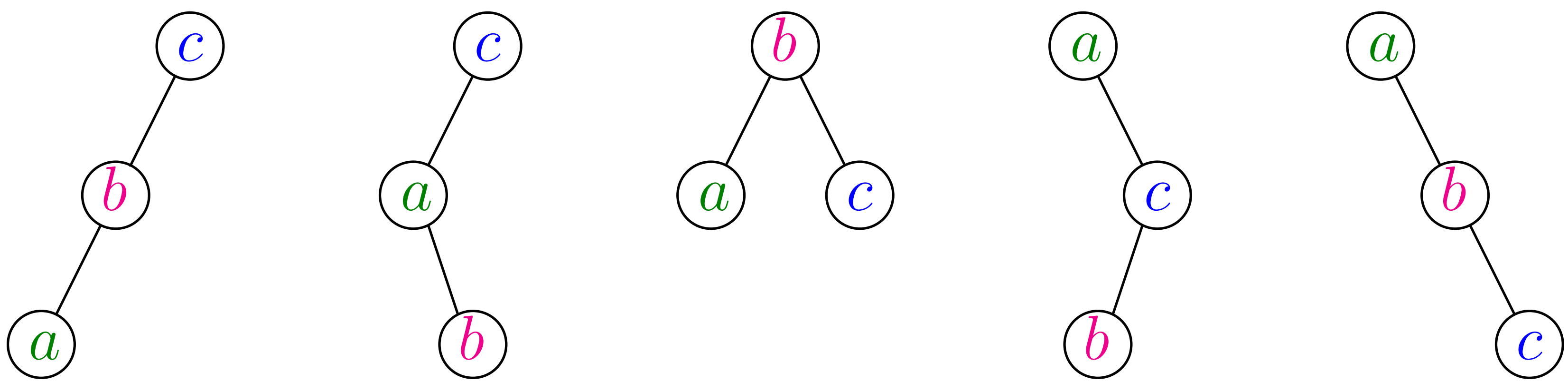
Buscas em conjunto conhecido

Dadas estimativas do número de acessos a cada elemento de $v[1\,..\,n]$, qual é a melhor estrutura de dados para v?

Árvore de busca binária (ABB)?

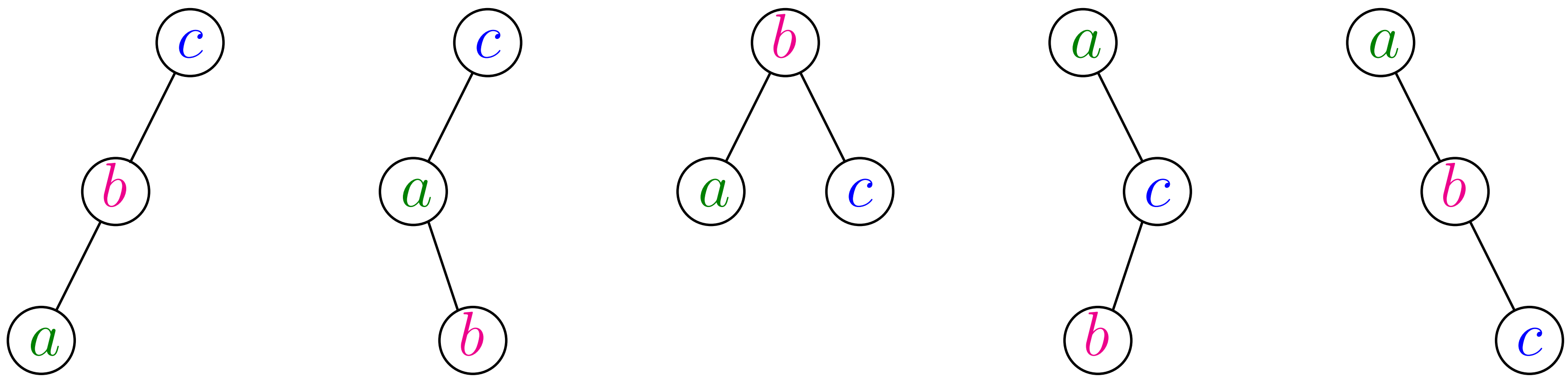
Exemplo: $n = 3$ e $e_1 = 10$, $e_2 = 20$, $e_3 = 40$.

Qual a melhor das ABBs?



Exemplo

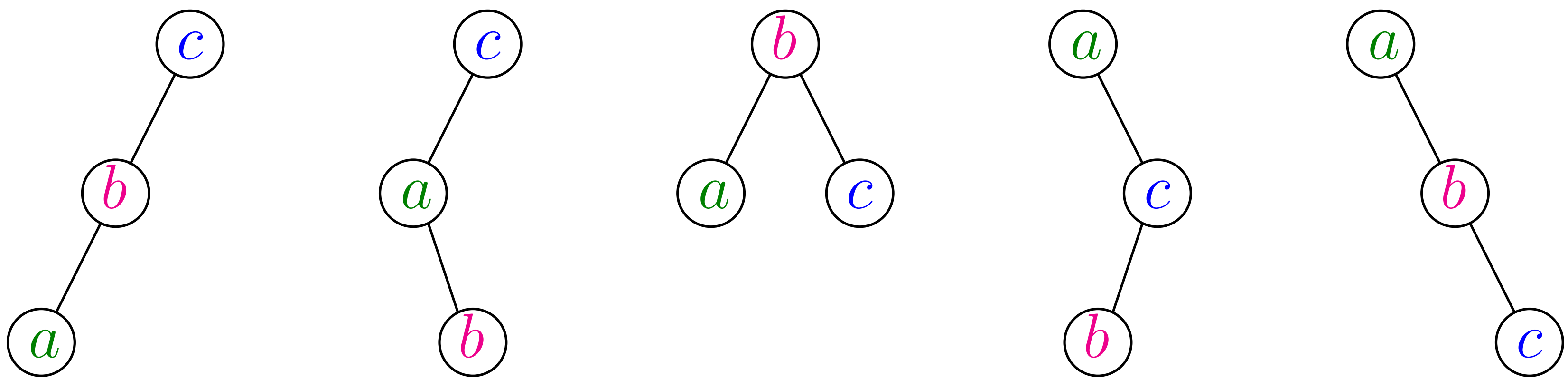
Exemplo: $n = 3$ e $e_1 = 10$, $e_2 = 20$, $e_3 = 40$.



Qual a melhor das ABBs?

Exemplo

Exemplo: $n = 3$ e $e_1 = 10$, $e_2 = 20$, $e_3 = 40$.

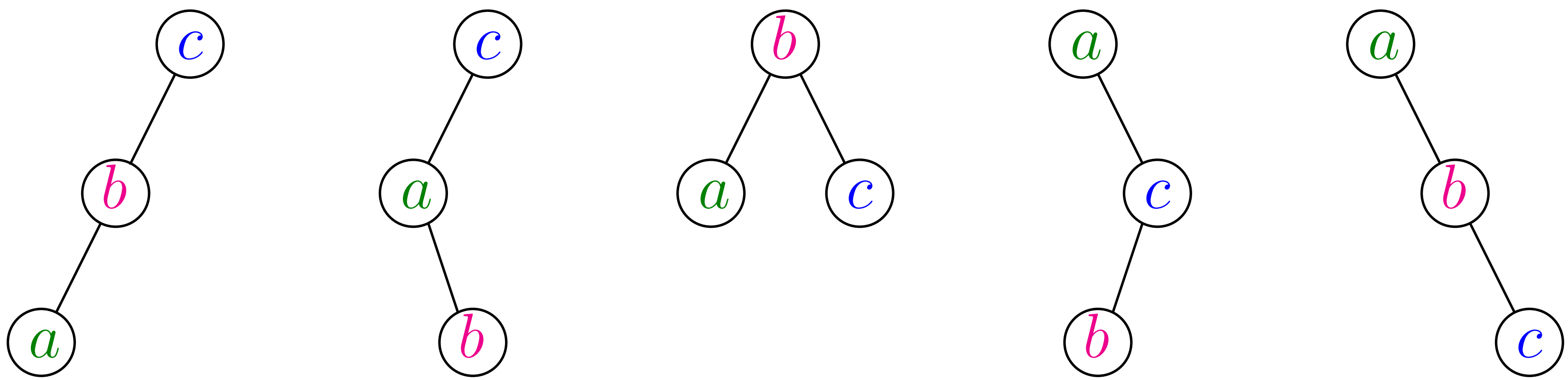


Número esperado de comparações:

- $10 \cdot 3 + 20 \cdot 2 + 40 \cdot 1 = 110$
- $10 \cdot 2 + 20 \cdot 3 + 40 \cdot 1 = 120$
- $10 \cdot 2 + 20 \cdot 1 + 40 \cdot 2 = 120$
- $10 \cdot 1 + 20 \cdot 3 + 40 \cdot 2 = 150$
- $10 \cdot 1 + 20 \cdot 2 + 40 \cdot 2 = 170$

Exemplo

Exemplo: $n = 3$ e $e_1 = 10$, $e_2 = 20$, $e_3 = 40$.



Número esperado de comparações:

- $10 \cdot 3 + 20 \cdot 2 + 40 \cdot 1 = 110$ $\longleftarrow$ ABB ótima
- $10 \cdot 2 + 20 \cdot 3 + 40 \cdot 1 = 120$
- $10 \cdot 2 + 20 \cdot 1 + 40 \cdot 2 = 120$
- $10 \cdot 1 + 20 \cdot 3 + 40 \cdot 2 = 150$
- $10 \cdot 1 + 20 \cdot 2 + 40 \cdot 2 = 170$

Árvore de busca ótima

Considere um vetor $e[1\,..\,n]$ de inteiros com uma estimativa do número de acessos a cada elemento de $\{1, \ldots, n\}$.

Árvore de busca ótima

Considere um vetor $e[1\,..\,n]$ de inteiros com uma estimativa do número de acessos a cada elemento de $\{1, \ldots, n\}$.

Uma ABB ótima com respeito ao vetor e é uma ABB para o conjunto $\{1, \ldots, n\}$ que minimiza o número

$$\sum_{i=1}^{n} h_i\, e_i,$$

onde h_i é o número de nós no caminho de i até a raiz da árvore.

Árvore de busca ótima

Considere um vetor $e[1\,..\,n]$ de inteiros com uma estimativa do número de acessos a cada elemento de $\{1,\ldots,n\}$.

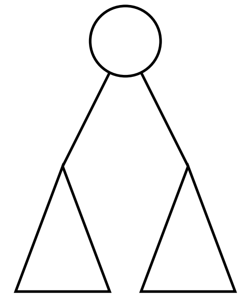
Uma ABB ótima com respeito ao vetor e é uma ABB para o conjunto $\{1,\ldots,n\}$ que minimiza o número

$$\sum_{i=1}^{n} h_i\, e_i,$$

onde h_i é o número de nós no caminho de i até a raiz da árvore.

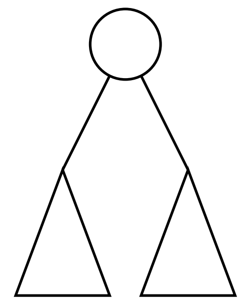
Problema (ABB Ótima): Dado $e[1\,..\,n]$, encontrar uma árvore de busca binária ótima com respeito a e.

Subestrutura ótima



Subárvores esquerda e direita de uma ABB ótima
são ABBs ótimas.

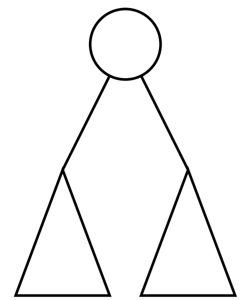
Subestrutura ótima



Subárvores esquerda e direita de uma ABB ótima
são ABBs ótimas.

Resta determinar a raiz da ABB ótima.

Subestrutura ótima

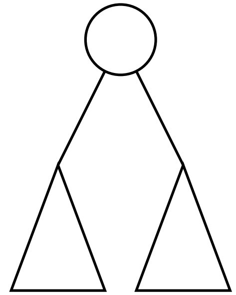


Subárvores esquerda e direita de uma ABB ótima são ABBs ótimas.

Resta determinar a raiz da ABB ótima.

$c[i, j]$: custo min de uma ABB para $e[i \mathrel{..} j]$
$s[i, j]$: soma dos acessos em $e[i \mathrel{..} j]$

Subestrutura ótima



Subárvores esquerda e direita de uma ABB ótima são ABBs ótimas.

Resta determinar a raiz da ABB ótima.

$c[i,j]$: custo min de uma ABB para $e[i\,..\,j]$
$s[i,j]$: soma dos acessos em $e[i\,..\,j]$

$$c[i,j] = \begin{cases} 0 & \text{se } i > j \\ \min_{i \le k \le j}\{c[i,k-1] + c[k+1,j] + s[i,j]\} & \text{se } i \le j \end{cases}$$

Custo de uma ABB ótima

$c[i,j]$: custo min de uma ABB para $e[i \,..\, j]$
$s[i,j]$: soma dos acessos em $e[i \,..\, j]$

$$c[i,j] = \begin{cases} 0 & \text{se } i > j \\ \min_{i \le k \le j}\{c[i,k-1] + c[k+1,j] + a[i,j]\} & \text{se } i \le j \end{cases}$$

Custo de uma ABB ótima

$c[i, j]$: custo min de uma ABB para $e[i \,..\, j]$
$s[i, j]$: soma dos acessos em $e[i \,..\, j]$

$$c[i, j] = \begin{cases} 0 & \text{se } i > j \\ \min_{i \le k \le j} \{c[i, k-1] + c[k+1, j] + a[i, j]\} & \text{se } i \le j \end{cases}$$

$$s[i, j] = \begin{cases} 0 & \text{se } i > j \\ s[i, j-1] + e_j & \text{se } i \le j \end{cases}$$

Custo de uma ABB ótima

$c[i,j]$: custo min de uma ABB para $e[i\,.\,.\,j]$
$s[i,j]$: soma dos acessos em $e[i\,.\,.\,j]$

$$c[i,j] = \begin{cases} 0 & \text{se } i > j \\ \min_{i \le k \le j}\{c[i,k-1] + c[k+1,j] + a[i,j]\} & \text{se } i \le j \end{cases}$$

$$s[i,j] = \begin{cases} 0 & \text{se } i > j \\ s[i,j-1] + e_j & \text{se } i \le j \end{cases}$$

Como preencher as matrizes c e s?
Em que ordem?

Custo de uma ABB ótima

$c[i,j]$: custo min de uma ABB para $e[i\,..\,j]$
$s[i,j]$: soma dos acessos em $e[i\,..\,j]$

$$c[i,j] = \begin{cases} 0 & \text{se } i > j \\ \min_{i \le k \le j}\{c[i,k-1] + c[k+1,j] + a[i,j]\} & \text{se } i \le j \end{cases}$$

$$s[i,j] = \begin{cases} 0 & \text{se } i > j \\ s[i,j-1] + e_j & \text{se } i \le j \end{cases}$$

Como preencher as matrizes c e s?
Em que ordem?
Como no problema da parentização! Pelas diagonais!

Árvore de busca ótima

ABB-ÓTIMA (e, n)

$s[0] = 0$
para $i \leftarrow 1$ **até** n **faça**
    $s[i] \leftarrow s[i-1] + e[i]$
para $i \leftarrow 1$ **até** $n+1$ **faça**
    $c[i][i-1] \leftarrow 0$
para $\ell \leftarrow 1$ **até** n **faça**
    para $i \leftarrow 1$ **até** $n-\ell+1$ **faça**
        $j \leftarrow i+\ell-1$
        $c[i][j] \leftarrow c[i+1][j]$
        para $k \leftarrow i+1$ **até** j **faça**
            se $c[i][k-1] + c[k+1][j] < c[i][j]$
            então $c[i][j] \leftarrow c[i][k-1] + c[k+1][j]$
        $c[i][j] \leftarrow c[i][j] + s[j] - s[i-1]$
devolva $c[1, n]$

Árvore de busca ótima

Exercício: Como fazer para obter uma ABB ótima e não apenas o seu custo? Complete o serviço!

Lista 5 na página em breve!