Mais programação dinâmica

CLRS 15.4

= "recursão–com–tabela"
= transformação inteligente de recursão em iteração

Subsequências

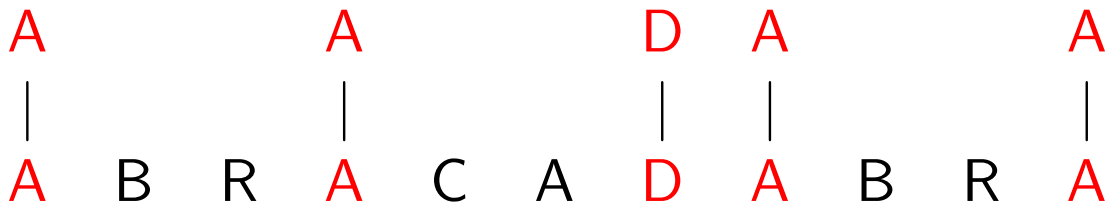
$\langle z_1, \ldots, z_k \rangle$ é subsequência de $\langle x_1, \ldots, x_m \rangle$
se existem índices $i_1 < \cdots < i_k$ tais que

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

EXEMPLOS:

$\langle 5, 9, 2, 7 \rangle$ é 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$\rangle$



Subsequência comum máxima

Z é subseq comum de X e Y

se Z é subsequência de X e de Y

ssco = subseq comum

Subsequência comum máxima

Z é subseq 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 subseq 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 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
+O
+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 $Y[1..n-1]$.

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 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 $Y[1..j]$

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:

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

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 $X[i] = Y[j]$

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 $X[i] = Y[j]$

$c[i,j] = \max(c[i,j-1], c[i-1,j])$ 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, j)
1   se i = 0 ou j = 0 então devolva 0
2   se X[i] = Y[j]
3       então c[i,j] ← REC-LCS-LENGTH (X, i−1, Y, j−1) + 1
4       senão q1 ← REC-LCS-LENGTH (X, i−1, Y, j)
5              q2 ← REC-LCS-LENGTH (X, i, Y, j−1)
6              se q1 ≥ q2
7                  então c[i,j] ← q1
8                  senãoc[i,j] ← q2
9   devolva c[i,j]
```

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 $n = 0,1,\ldots$ e $m = 0,1,\ldots$.

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

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

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

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

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

Recorrência

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

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

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

onde

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

$S(k)$ 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 $i = 1$ e $j = 0,1,\ldots,n$,
depois todos com $i = 2$ e $j = 0,1,\ldots,n$,
depois todos com $i = 3$ e $j = 0,1,\ldots,n$,
etc.

Programação dinâmica

i \ j	1	2	3	4	5	6	7	8
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							

Simulação

			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

			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

			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

			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

			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							

Y

i

Simulação

			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

			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							

Y

i

Simulação

			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							

Y

i

Simulação

			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							

Y

i

Simulação

			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							

Y

i

Simulação

			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

			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							

Y

i

Simulação

			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

			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

			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

			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

			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

			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							

Y

i

Simulação

			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

			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

			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

			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

			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

			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

			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							

Y

i

Simulação

			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

			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

			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	??	
A	6	0							
B	7	0							
i									

Simulação

			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

			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

			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

			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

			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

			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

			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

			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

			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

			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

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	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)
 1  para i ← 0 até m faça c[i,0] ← 0
 2  para j ← 1 até n faça c[0,j] ← 0
 3  para i ← 1 até m faça
 4      para j ← 1 até n faça
 5          se X[i] = Y[j]
 6              então c[i,j] ← c[i−1,j−1] + 1
 7              senão se c[i−1,j] ≥ c[i,j−1]
 8                          então c[i,j] ← c[i−1,j]
 9                          senão c[i,j] ← c[i,j−1]
10  devolva c[m,n]
```

Algoritmo de programação dinâmica

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

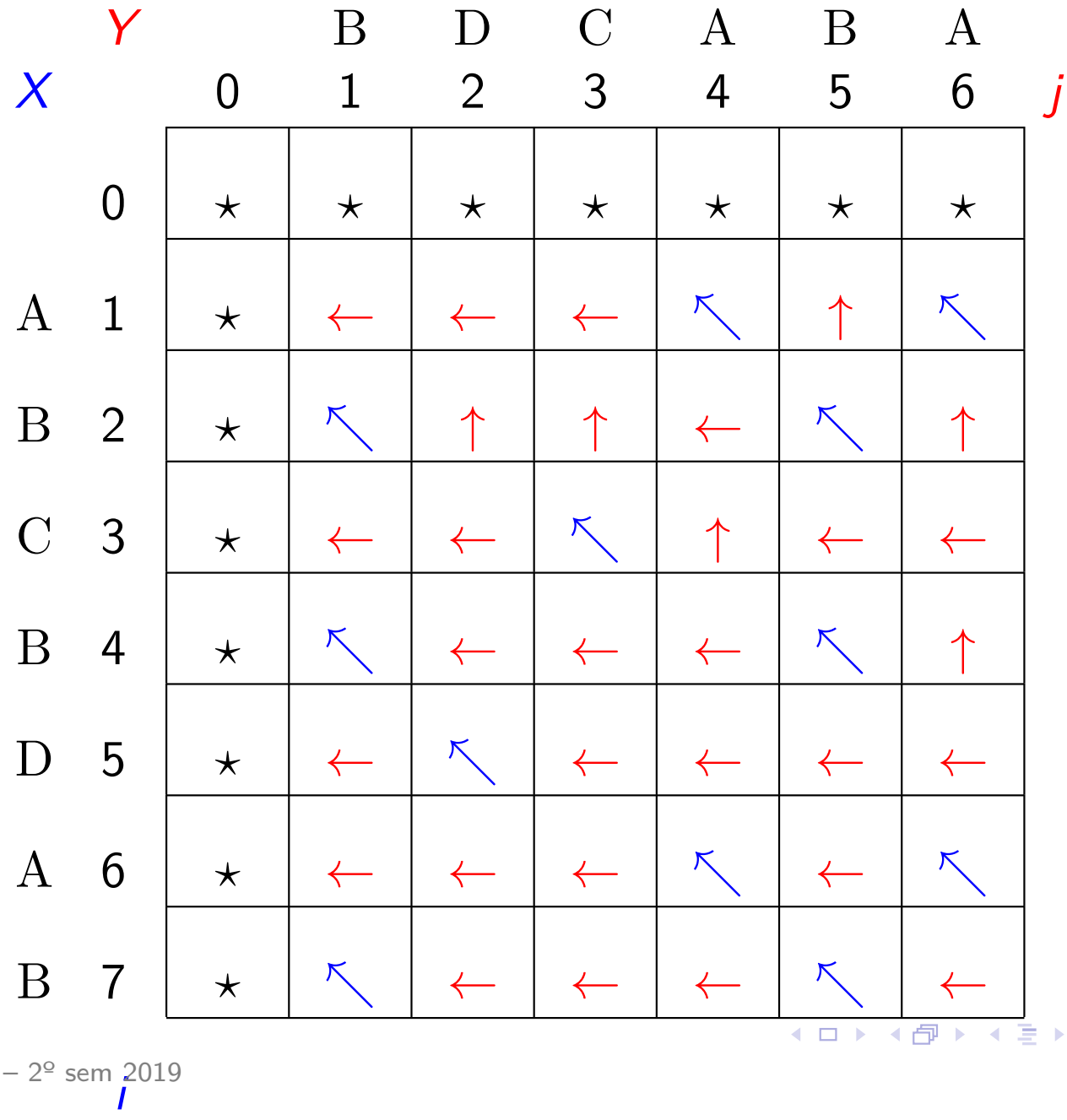
```
LEC-LENGTH (X, m, Y, n)
 1  para i ← 0 até m faça c[i,0] ← 0
 2  para j ← 1 até n faça c[0,j] ← 0
 3  para i ← 1 até m faça
 4      para j ← 1 até n faça
 5          se X[i] = Y[j]
 6              então c[i,j] ← c[i−1,j−1] + 1
 7              senão se c[i−1,j] ≥ c[i,j−1]
 8                          então c[i,j] ← c[i−1,j]
 9                          senão c[i,j] ← c[i,j−1]
10  devolva c[m,n]
```

Consumo de tempo: $O(mn)$

Conclusão

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

Subsequência comum máxima



Algoritmo de programação dinâmica

```
LEC-LENGTH (X, m, Y, n)
 1  para i ← 0 até m faça c[i,0] ← 0
 2  para j ← 1 até n faça c[0,j] ← 0
 3  para i ← 1 até m faça
 4      para j ← 1 até n faça
 5          se X[i] = Y[j]
 6              então c[i,j] ← c[i − 1,j − 1] + 1
 7                    b[i,j] ← "↖"
 8              senão se c[i − 1,j] ≥ c[i,j − 1]
 9                          então c[i,j] ← c[i − 1,j]
10                                b[i,j] ← "↑"
11                          senão c[i,j] ← c[i,j − 1]
12                                b[i,j] ← "←"
13  devolva c e b
```

Algoritmo de programação dinâmica

```
LEC-LENGTH (X, m, Y, n)
 1  para i ← 0 até m faça c[i,0] ← 0
 2  para j ← 1 até n faça c[0,j] ← 0
 3  para i ← 1 até m faça
 4      para j ← 1 até n faça
 5          se X[i] = Y[j]
 6              então c[i,j] ← c[i−1,j−1] + 1
 7                    b[i,j] ← "↖"
 8              senão se c[i−1,j] ≥ c[i,j−1]
 9                          então c[i,j] ← c[i−1,j]
10                                b[i,j] ← "↑"
11                          senão c[i,j] ← c[i,j−1]
12                                b[i,j] ← "←"
13  devolva c e b
```

Consumo de tempo: $O(mn)$

Get-LCS

```
GET-LCS (X, m, n, b, máxcomp)
 1  k ← máxcomp
 2  i ← m
 3  j ← n
 4  enquanto i > 0 e j > 0 faça
 5      se b[i,j] = "↖"
 6          então Z[k] ← X[i]
 7                k ← k − 1    i ← i − 1    j ← j − 1
 8          senão se b[i,j] = "←"
 9                then i ← i − 1
10                senão j ← j − 1
11  devolva Z
```

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

Exercícios

Exercício 20.A
Escreva um algoritmo para decidir se $\langle z_1, \ldots, z_k \rangle$ é subsequência 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 sequência $\langle a_1, \ldots, a_n \rangle$ são distintos dois a dois. Quantas subsequências tem a sequência?

Exercício 20.C
Uma subsequência crescente Z de uma sequência X e 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 codições: 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 $P_1, P_2, \ldots, P_n$ 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.

Mais programação dinâmica

CLRS 15.5

= "recursão–com–tabela"
= transformação inteligente de recursão em iteração

Buscas em um conjunto conhecido

Considere um inteiro n e um vetor $v[1..n]$ de `tipoOrd`.

Problema: Dado $v[1..n]$ e uma sequência de k elementos do `tipoOrd`, decidir se cada elemento está ou não em v.

Buscas em um conjunto conhecido

Considere um inteiro n e um vetor $v[1..n]$ de `tipoOrd`.

Problema: Dado $v[1..n]$ e uma sequência de k elementos do `tipoOrd`, decidir se cada elemento 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 `tipoOrd`.

Problema: Dado $v[1..n]$ e uma sequência de k elementos do `tipoOrd`, decidir se cada elemento 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 `tipoOrd`.

Problema: Dado $v[1..n]$ e uma sequência de k elementos do `tipoOrd`, decidir se cada elemento 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 ordenar v e aplicar busca binária.

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 $e_1 = 10$, $e_2 = 20$, $e_3 = 40$.

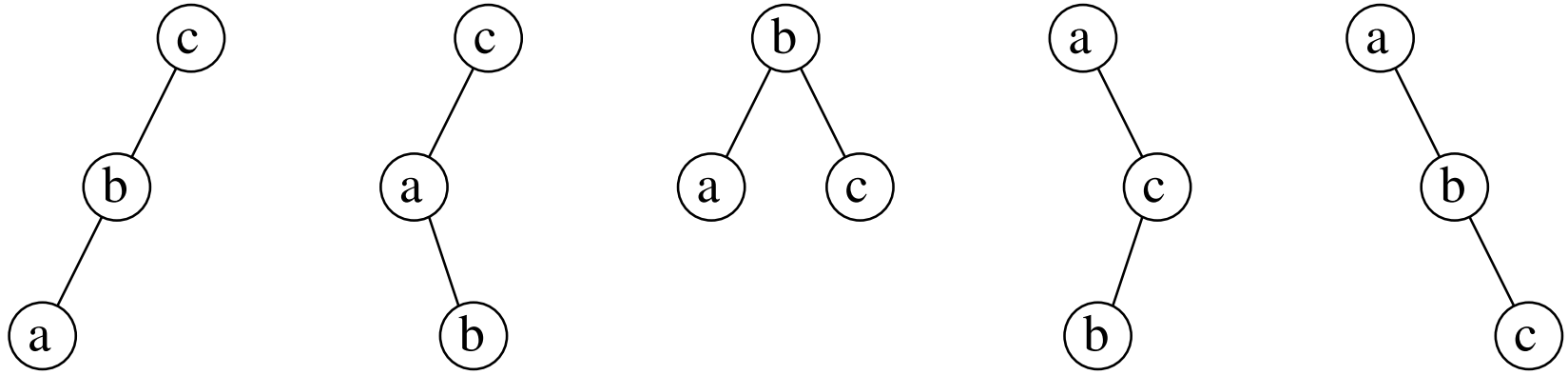
Buscas em conjunto conhecido

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)?

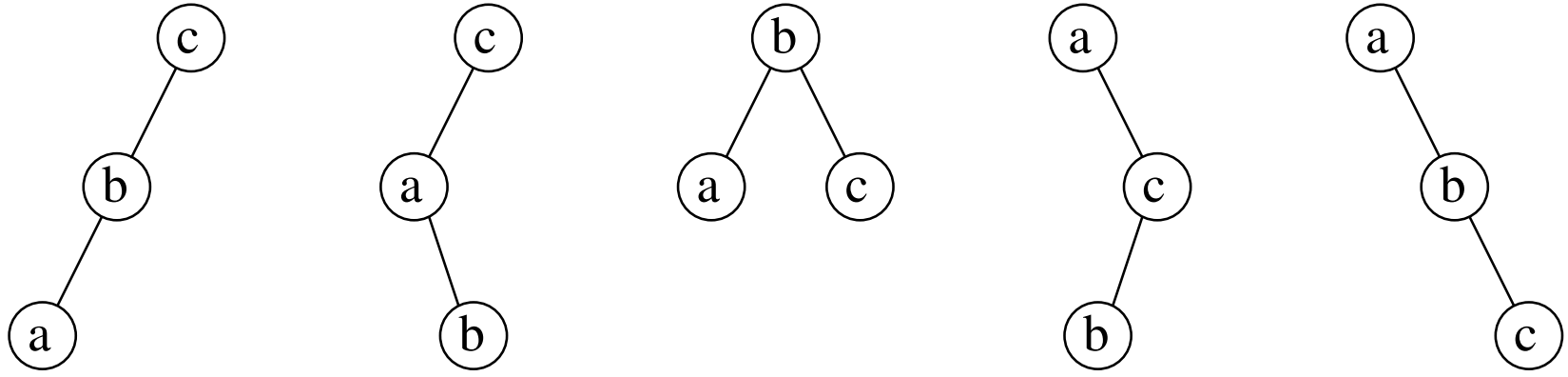
Exemplo: $n = 3$ e $e_1 = 10$, $e_2 = 20$, $e_3 = 40$.

Qual a melhor das ABBs?



Exemplo

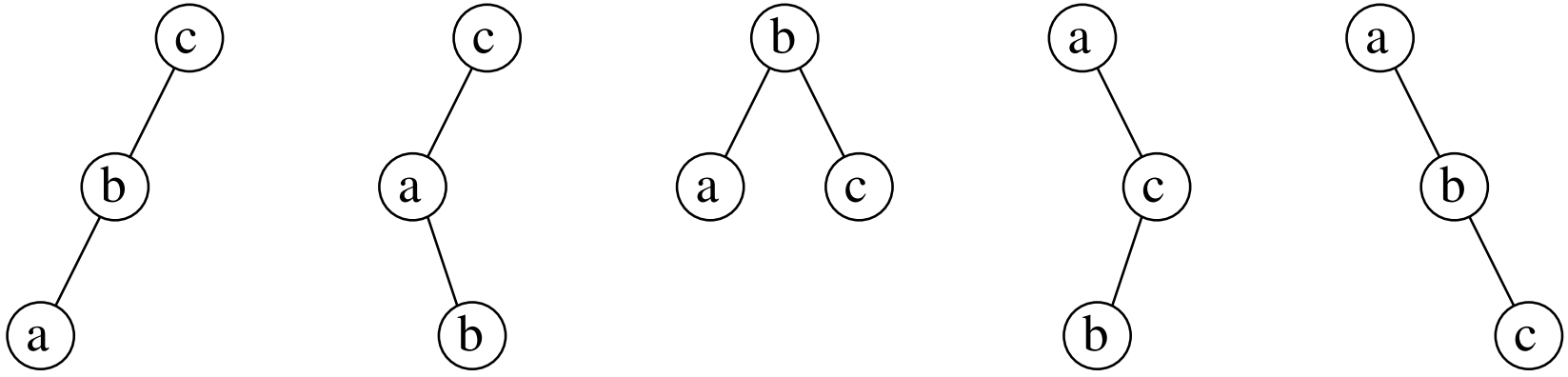
Exemplo: $n = 3$ e $e_1 = 10$, $e_2 = 20$, $e_3 = 40$.



Qual a melhor das ABBs?

Exemplo

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

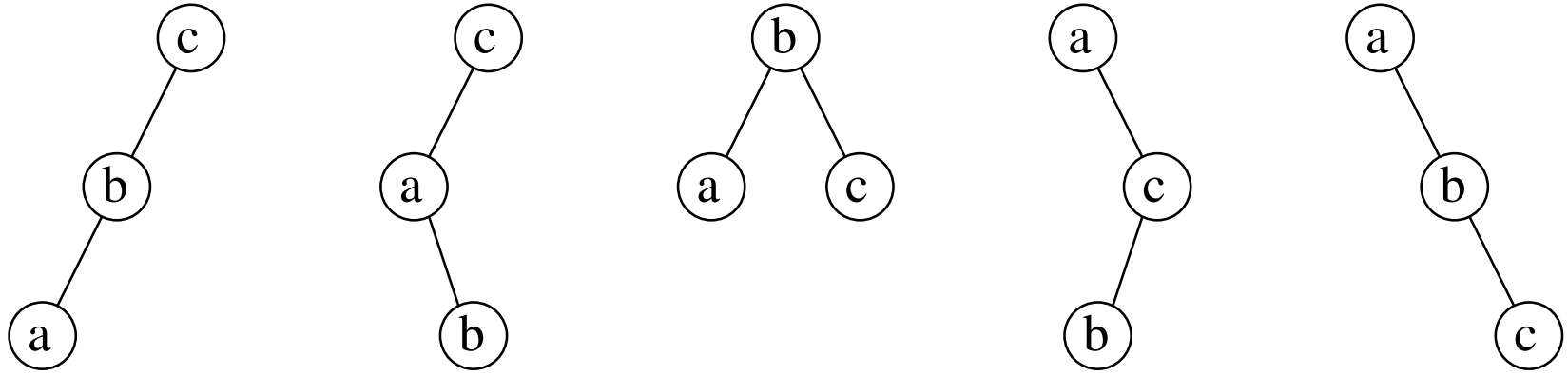


Número esperado de comparações:

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

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