Fundamentos de análisis y diseño de algoritmos

Programación dinámica

LCS (Longest Common Subsequence)

Multiplicación de matrices

Propiedades generales de la programación dinamica

Programación dinámica

Dada una secuencia $X=\langle x_1,x_2,\ldots x_m\rangle$, se dice que $Z=\langle z_1,z_2,\ldots,z_n\rangle$ es una subsecuencia de X si los simbolos z_i se dan en X siguiendo el orden que se dan en Z (no necesariamente uno seguido de otro)

Por ejemplo, para $X=\langle A,B,C,B,D,A,B\rangle$, algunas subsecuencias son:

- $\langle A,B,C\rangle$,
- $\langle A,B,C,D\rangle$
- $\langle B,C,D,B\rangle$

Programación dinámica

Por ejemplo, para X=<A,B,C,B,D,A,B>, algunas subsecuencias son:

<A,B,C>,

<A,B,C,D>

<B,C,D,B>

Sin embargo, no son subsecuencias:

<A,B,A,A>,

<B,A,D>

<D,A,D>

Programación dinámica

Problema; Dadas dos secuencias $X=\langle x_1,x_2,\ldots x_m\rangle$ y $Y=\langle y_1,y_2,\ldots y_n\rangle$, encontrar la(s) *subsecuencia(s) más larga(s)* entre a X y Y

X=ABCBDAB

Y=BDCABA

Indique algunas subsecuencias

ABA, BCA, DAB

Indique una solución al problema

BCBA

BDAB

Programación dinámica

Problema: Dadas dos secuencias $X=\langle x_1,x_2,\ldots x_m\rangle$ y $Y=\langle y_1,y_2,\ldots y_n\rangle$, encontrar la(s) *subsecuencia(s) más larga(s)* entre a X y Y

X=ABCBDAB

Y=BDCABA

Algunas subsecuencias son: A, CB, ABA, BDAB, CAB, DAB

El problema de la subsecuencia más larga tiene varias soluciones: BCBA, BCAB y BDAB

Programación dinámica

Problema; Dadas dos secuencias $X=\langle x_1,x_2,\ldots x_m\rangle$ y $Y=\langle y_1,y_2,\ldots y_n\rangle$, encontrar la(s) *subsecuencia(s) más larga(s)* entre a X y Y

¿Cuál podría ser una solución al problema?

ABCAB

BCDB

LCS(ABCA, BCD)+B

BCB

Programación dinámica

Problema; Dadas dos secuencias $X=<x_1,x_2,\ldots x_m>$ y $Y=<y_1,y_2,\ldots y_n>$, encontrar la(s) *subsecuencia(s) más larga(s)* entre a X y Y

¿Cuál podría ser una solución al problema?

Enumerar todas las subsecuencias de X y examinar si es también una subsecencia de Y. Tomar al final la subsecuencia más larga

Existen 2^m posibles suconjuntos (subsecuencias) del conjunto de m elementos →Tiempo exponencial

Programación dinámica

Problema; Dadas dos secuencias $X=\langle x_1,x_2,\ldots x_m\rangle$ y $Y=\langle y_1,y_2,\ldots y_n\rangle$, encontrar la(s) *subsecuencia(s) más larga(s)* entre a X y Y

La solución al problema presenta una característica especial

Programación dinámica

Problema; Dadas dos secuencias $X=<x_1,x_2,...x_m>$ y $Y=<y_1,y_2,...y_n>$, encontrar la *subsecuencia más larga (LCS)* entre a X y Y

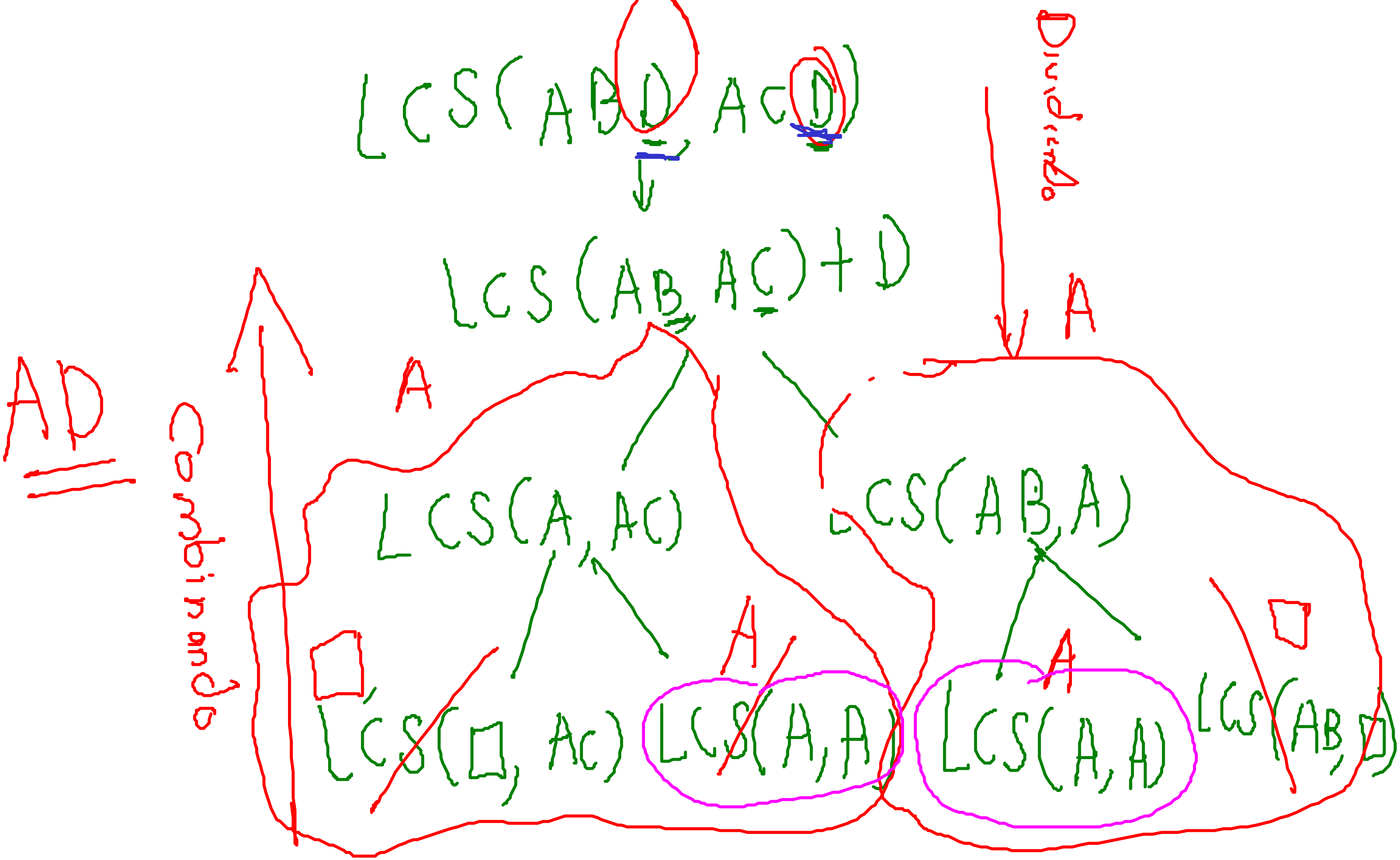
- Si $x_m=y_n$, la solución del problema $LCS(X_m,Y_n)$ será $LCS(X_{m-1},Y_{n-1})$ y pegar al final x_m
- Si $x_m \neq y_n$, la solución del problema $LCS(X_m,Y_n)$ se escoge entre $LCS(X_{m-1},Y_n)$ y $LCS(X,Y_{n-1})$, esto es, de estas dos se elije aquella de mayor longitud

Programación dinámica

Si $x_m = y_n$, la solución del problema LCS(X_m,Y_n) será LCS(X_{m-1},Y_{n-1}) y pegar al final x_m

- Si $x_m \neq y_n$, la solución del problema LCS(X_m,Y_n) se escoge entre LCS(X_{m-1},Y_n) y LCS(X,Y_{n-1}), esto es, de estas dos se elije aquella de mayor longitud

LCS(ABD, ACD)



LCS(ABD, ACD)
Dividiendo
LCS(AB, AC) + D
A
A
AD
Combinando
LCS(A, AC)
LCS(AB, A)
A
A
LCS(□, AC)
LCS(A, A)
LCS(A, A)
LCS(AB, □)

Programación dinámica

Si $x_m = y_n$, la solución del problema $LCS(X_m, Y_n)$ será $LCS(X_{m-1}, Y_{n-1})$ y pegar al final x_m

- Si $x_m \neq y_n$, la solución del problema $LCS(X_m, Y_n)$ se escoge entre $LCS(X_{m-1}, Y_n)$ y $LCS(X, Y_{n-1})$, esto es, de estas dos se elije aquella de mayor longitud

LCS(ABD, ACD)

solucion= LCS(AB, AC) + D

Programación dinámica

Si $x_m=y_n$, la solución del problema LCS(X_m,Y_n) será LCS(X_{m-1},Y_{n-1}) y pegar al final x_m

- Si $x_m \neq y_n$, la solución del problema LCS(X_m,Y_n) se escoge entre LCS(X_{m-1},Y_n) y LCS(X,Y_{n-1}), esto es, de estas dos se elije aquella de mayor longitud

LCS(ABD, ACD)

solucion= LCS(AB, AC) + D

solucion'= LCS(A, AC)   solucion'= LCS(AB, A)

Programación dinámica

- LCS(ABD, ACD)

  solucion= LCS(AB, AC) + D

solucion'= LCS(A, AC)　　solucion'= LCS(AB, A)

solucion''= LCS(, AC)　　solucion''= LCS(A,A)

Programación dinámica

- LCS(ABD, ACD)

solucion= LCS(AB, AC) + D

solucion'= LCS(A, AC) solucion'= LCS(AB, A)

solucion''= LCS(, AC) solucion''= LCS(A,A)

LCS(A,A)= LCS(,) + A

Se obtiene AD como una solución

Programación dinámica

- LCS(ABD, ACD)

solucion= LCS(AB, AC) + D

solucion'= LCS(A, AC) solucion'= LCS(AB, A)

solucion''= LCS(A, A) solucion''= LCS(AB,)

Ya fue calculado, no se calcula nuevamente.

Se retorna A

Se obtiene AD como una solución

Programación dinámica

Note que hay subproblemas que se sobrelapan, para encontrar $LCS(X_m, Y_n)$ se podría necesitar encontrar $LCS(X_{m-1}, Y_n)$ y $LCS(X_m, Y_{n-1})$, pero cada uno de los subproblemas tendrá que encontrar $LCS(X_{m-1}, Y_{n-1})$

Cuando se comparten subproblemas, no se calculan nuevamente, ya se conoce el valor, simplemente se utiliza

Estrategia: Divide y vencerás donde los subproblemas no son independientes sino que algunos son compartidos. Esto se aprovecha para no tener que hacer cálculos repetidos

$A = \{a_1, a_2, \ldots, a_n\}$

$B = \{b_1, b_2, \ldots, b_m\}$

$C[n, m]$

$LCS(A, B)$

$C[n-1, m-1]$

Programación dinámica

Se tiene una función de costo que permite conocer la longitud de la solución de óptima:

$$c[i,j] = \begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1] + 1 & \text{, si } x_i = x_j \text{ e } i,j>0 \\ \max(c[i,j-1],\ c[i-1,j]), & \text{si } x_i \neq x_j \text{ e } i,j>0 \end{cases}$$

c es una matriz que indica la longitud de la secuencia más larga entre X_i y Y_j

Programación dinámica

LCS(ABCBDAB, BDCABA)

LCS(A, B)

LCS(A, B)

MAX (LCS([]B) LCS(A,[]))

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

LCS(AB, BDCAB)

LCS(A, BDCA) + 1

LCS(ABCB, BDC)

Programación dinámica

LCS(ABCBDAB, BDCABA)

		0	1	2	3	4	5	6
			B	**D**	**C**	**A**	**B**	**A**
0								
1	**A**							
2	**B**							
3	**C**							
4	**B**							
5	**D**							
6	**A**							
7	**B**							

Construir

Programación dinámica

LCS(ABCBDAB, BDCABA)

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

C[0,1] es la longitud de la subsecuencia más larga entre las secuencias X=" " y Y="B"

LCS(A, B)

LCS(□, B) LCS(A, □)

LCS(B DCA, A)

LCS(BDC, □)+A

LCS(BDCAB, A)

LCS(BDCA, A)

1

LCS(ABCBDA, BDCA)

LCS(ABCBD, BDCAB)

BDAB BCAB BCBA

Programación dinámica

LCS(ABCBDAB, BDCABA)

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

C[0,1] es la longitud de la subsecuencia más larga entre las secuencias X=" " y Y="B"

Programación dinámica

LCS(ABCBDAB, BDCABA)

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

Programación dinámica

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

Programación dinámica

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

$$
c[i,j]= \begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1] + 1 & \text{, si } x_i=x_j \text{ e } i,j>0 \\ \max(c[i,j-1],\ c[i-1,j]), & \text{si } x_i \neq x_j \text{ e } i,j>0 \end{cases}
$$

Programación dinámica

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

$$c[i,j] = \begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1] + 1 & \text{, si } x_i = x_j \text{ e } i,j>0 \\ \max(c[i,j-1],\ c[i-1,j]), & \text{si } x_i \neq x_j \text{ e } i,j>0 \end{cases}$$

Programación dinámica

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

$$c[i,j]=\begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1]+1 & \text{, si } x_i=x_j \text{ e } i,j>0 \\ \max(c[i,j-1],\ c[i-1,j]), & \text{si } x_i \neq x_j \text{ e } i,j>0 \end{cases}$$

Programación dinámica

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

$$c[i,j] = \begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1]+1 & \text{, si } x_i = x_j \text{ e } i,j>0 \\ \max(c[i,j-1],\ c[i-1,j]) & \text{, si } x_i \neq x_j \text{ e } i,j>0 \end{cases}$$

Programación dinámica

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

$$c[i,j]=\begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1]+1 & \text{, si } x_i=x_j \text{ e } i,j>0 \\ \max(c[i,j-1],\ c[i-1,j]), & \text{si } x_i\neq x_j \text{ e } i,j>0 \end{cases}$$

Programación dinámica

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

$$c[i,j] = \begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1]+1 & \text{, si } x_i=x_j \text{ e } i,j>0 \\ \max(c[i,j-1],\ c[i-1,j]), & \text{si } x_i \neq x_j \text{ e } i,j>0 \end{cases}$$

Programación dinámica

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

$$c[i,j]= \begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1] + 1 & \text{, si } x_i=x_j \text{ e } i,j>0 \\ \max(c[i,j-1],\ c[i-1,j]), & \text{si } x_i \neq x_j \text{ e } i,j>0 \end{cases}$$

Programación dinámica

			0	1	2	3	4	5	6
				B	D	C	A	B	A
	0		0	0	0	0	0	0	0
	1	A	0	0	0	0	1	1	1
	2	B	0	1					
Longitud para LCS(AB,B)	3	C	0						
	4	B	0						
	5	D	0						
	6	A	0						
	7	B	0						

$$c[i,j] = \begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1]+1 & \text{, si } x_i = x_j \text{ e } i,j>0 \\ \max(c[i,j-1],\ c[i-1,j]), & \text{si } x_i \neq x_j \text{ e } i,j>0 \end{cases}$$

Programación dinámica

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

$$
c[i,j] = \begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1] + 1 & \text{, si } x_i = x_j \text{ e } i,j>0 \\ \max(c[i,j-1],\ c[i-1,j]), & \text{si } x_i \neq x_j \text{ e } i,j>0 \end{cases}
$$

Programación dinámica

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

Longitud para LCS(A,BD)

Longitud para LCS(AB,B)

Longitud para LCS(AB,BD)

$$c[i,j] = \begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1] + 1 & \text{, si } x_i = x_j \text{ e } i,j>0 \\ \max(c[i,j-1],\ c[i-1,j]), & \text{si } x_i \neq x_j \text{ e } i,j>0 \end{cases}$$

Programación dinámica

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

Completar la tabla

Programación dinámica

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

Programación dinámica

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

Qué significado tiene el hecho de que c[4,4]=2

Programación dinámica

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

Qué significado tiene el hecho de que c[4,4]=2

Indica que la longitud de LCS(ABCB,BDCA) es 2

Programación dinámica

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

Qué significado tiene el hecho de que c[7,4]=3

Indica que la longitud de LCS(ABCBDAB,BDCA) es 3

Programación dinámica

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

Qué casilla de la matriz guarda la longitud de la solución al problema original

Programación dinámica

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

Cuál es la solución, es decir, cual es la subsecuencia común?

Programación dinámica

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

Las direcciones se guardan en otro arreglo llamado B

$$c[i,j] = \begin{cases} 0 & \text{, si } i=0 \text{ o } j=0 \\ c[i-1,j-1]+1 & \text{, si } x_i = x_j \text{ e } i,j>0 \; (\nwarrow) \\ \max(c[i,j-1],\, c[i-1,j]), & \text{si } x_i \neq x_j \text{ e } i,j>0 \; (\leftarrow) \; (\uparrow) \end{cases}$$

Programación dinámica

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

Solo ↖ indica que es un símbolo común, éstos se imprimen.
En los demás casos, se sigue la flecha, ↑ o ←

Programación dinámica

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

Muestre la solución al problema

Programación dinámica

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

ABCB, se invierte y se obtiene BCBA

Programación dinámica

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

Muestre la solución al problema LCS(ABCB, BDCABA)

Programación dinámica

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

Muestre la solución al problema LCS(ABCB, BDCABA)

BCB, que en orden invertido es BCB

```
LCS-LENGTH(X,Y)
m←length[X]
n←length[Y]
for i←1 to m
  do c[i,0]←0
for j←0 to n
  do c[0,j]←0
for i←1 to m
  do for j←1 to n
        do if x_i=y_j
              then c[i,j]←c[i-1,j-1]+1
                   b[i,j]←"↖"
           else if c[i-1,j]≥c[i,j-1]
                   then c[i,j]←c[i-1,j]
                        b[i,j]←"↑"
                   else c[i,j]←c[i,j-1]
                        b[i,j]←"←"
 return c and b
```

Indique la longitud de

LCS(AFCEA, CFEHA)

```
PRINT-LCS(b, X, i, j)
  if i=0 or j=0
     then return
  if b[i,j]=" ↖ "
     then PRINT-LCS(b, X, i-1, j-1)
     print xi
  else if b[i,j]=" ↑"
          then PRINT-LCS(b, X, i-1, j)
  else PRINT-LCS(b, X, i, j-1)
```

Resuelva LCS(AFCEA, CFEHA)