Análisis y Diseño de Algoritmos II

Jesús Alexander Aranda Ph.D   Robinson Duque, Ph.D
Juan Francisco Díaz, Ph. D

Universidad del Valle

jesus.aranda@correounivalle.edu.co
robinson.duque@correounivalle.edu.co
juanfco.diaz@correounivalle.edu.co
Programa de Ingeniería de Sistemas
Escuela de Ingeniería de Sistemas y Computación

AVISPA
AMBIENTES VISUALES DE PROGRAMACIÓN APLICATIVA

Universidad
del Valle

Programación Dinámica

Caso de Estudio

LCS (Longest Common Subsequence)

Programación dinámica

Subsecuencia

Dada una secuencia $X = \langle x_1, x_2, \ldots x_m \rangle$, se dice que $Z = \langle z_1, z_2, \ldots, z_k \rangle$ es una subsecuencia de X si existe una secuencia incremental de índices $\langle i_1, i_2, \ldots, i_k \rangle$ de X tal que

$$\forall\, j \in \{1, 2, 3, \ldots, k\}\ x_{i_j} = z_j$$

(es decir, los símbolos de Z se dan en X siguiendo el mismo orden que se dan en Z (no necesariamente uno seguido de otro en X)

Programación dinámica

Ejemplos:

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>

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

Longest Common Subsequence

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 X y Y

X=ABCBDAB

Y=BDCABA

Indique algunas subsecuencias

Indique una solución al problema

Programación dinámica

Longest Common Subsequence

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 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

Longest Common Subsequence

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 X y Y

¿Cuál podría ser una solución al problema?

Programación dinámica

Longest Common Subsequence (Fuerza Bruta)

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 X y Y

Solución Inicial:

Estrategia por Fuerza Bruta

- Enumerar todas las subsecuencias de X y examinar si es también una subsecuencia de Y (o viceversa). Tomar al final la subsecuencia más larga.

Programación dinámica

Longest Common Subsequence (Fuerza Bruta)

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 X y Y

Solución Inicial:

Estrategia por Fuerza Bruta

- Enumerar todas las subsecuencias de X y examinar si es también una subsecuencia de Y (o viceversa). Tomar al final la subsecuencia más larga.

¿ Cuántas subsecuencias tiene X (o Y) ?

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

Programación dinámica

Longest Common Subsequence (Fuerza Bruta)

¿Cuántas subsecuencias tiene X (o Y) ?

Ejemplo: La secuencia < A,B,C> tiene:

A	B	C	Subsecuencias
0	0	0	Cadena vacía (ε)
0	0	1	C
0	1	0	B
0	1	1	BC
1	0	0	A
1	0	1	AC
1	1	0	AB
1	1	1	ABC
Total subsecuencias			8

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

Programación dinámica

Longest Common Subsequence (Fuerza Bruta)

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 X y Y

Estrategia por Fuerza Bruta es muy costosa computacionalmente.

¿Alternativas más eficientes para resolver el problema LCS?

Programación dinámica

Longest Common Subsequence (Fuerza Bruta)

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 X y Y

Estrategia por Fuerza Bruta es muy costosa computacionalmente.

¿Alternativas más eficientes para resolver el problema LCS?

R/ Programación Dinámica

Programación dinámica

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

Tener en cuenta que al ser un problema de optimización. Cada instancia tiene al menos una solución (óptima) y un valor (el cual indica que es la solución efectivamente mejor)

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 X y Y

Tener en cuenta que al ser un problema de optimización. Cada instancia tiene al menos una solución (óptima) y su respectivo valor (el cual precisamente permite establecer que es la mejor solución)

Instancias	Solución	Valor de la Solución
X=<A,B,A,C> Y=<B,D,A>	<B,A>	2
X= <C,B,A,D> Y= <E,F >	ε (< > secuencia vacía)	0
X = <A,B,A,C > Y = <A,B,A >	<A,B,A>	3

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 *subsecuencia más larga (LCS)* entre X y Y

Al aplicar programación dinámica sobre un problema de optimización se siguen los siguientes pasos:

1) Caracterizar la estructura de la solución óptima (subestructura óptima).
2) Definir recursivamente el valor de la solución óptima.
3) Calcular el valor de la solución óptima (algoritmo).
4) Construir la solución óptima (algoritmo).

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 X y Y

1) Caracterización de la estructura de la solución óptima (subestructura óptima)

- Toda solución (óptima) de un problema se forma de soluciones óptimas de subproblemas

Ejemplo: Considere LCS(<A,C,B>, <D,A,E,B>) (problema original)

Como x_3 (B) = y_4 (B) entonces

La solución de LCS(<A,C,B>, <D,A,E,B>) = la solución de LCS(<A,C>, <D,A,E>) + B

En general, considerando LCS($<x_1, x_2, \ldots, x_m>, <y_1, y_2, \ldots, y_n>$) (problema original)

Si $x_m = y_n$ entonces

La solución de LCS($<x_1, x_2, \ldots, x_m>, <y_1, y_2, \ldots, y_n>$) =

la solución de LCS($<x_1, x_2, \ldots, x_{m-1}>, <y_1, y_2, \ldots, y_{n-1}>$) + x_m

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 *subsecuencia más larga (LCS)* entre X y Y

1) Caracterización de la estructura de la solución óptima (subestructura óptima)

- Toda solución (óptima) de un problema se forma de soluciones óptimas de subproblemas

Ejemplo: Considere LCS(<A,C,B>, <D,A,E,B>) (problema original)

Como x_3 (B) = y_4 (B) entonces

La solución de LCS(<A,C,B>, <D,A,E,B>) = la solución de LCS(<A,C>, <D,A,E>) + B

En general, considerando LCS($<x_1, x_2, \ldots , x_m>, <y_1, y_2, \ldots , y_n>$) (problema original)

Si $x_m = y_n$ entonces

La solución de LCS($<x_1, x_2, \ldots , x_m>, <y_1, y_2, \ldots , y_n>$) =

la solución de LCS($<x_1, x_2, \ldots , x_{m-1}>, <y_1, y_2, \ldots , y_{n-1}>$) + x_m

(Si X_i corresponde a $<x_1, x_2, \ldots , x_i>$)

Si $x_m = y_n$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_{m-1}, Y_{n-1}) + x_m

Programación dinámica

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

1) Caracterización de la estructura de la solución óptima (subestructura óptima)

- Toda solución (óptima) de un problema se forma de soluciones óptimas de subproblemas

Ejemplo: Considere LCS($\langle A,B,C\rangle$, $\langle D,A,E,B\rangle$) donde su solución es $\langle A,B\rangle$

Como x_3 (C) $\neq y_4$ (B) y $x_3 \neq z_2$ (B) entonces

La solución de LCS($\langle A,B,C\rangle$, $\langle D,A,E,B\rangle$) = la solución de LCS($\langle A,B\rangle$, $\langle D,A,E,B\rangle$)

En general, considerando LCS(X_m, Y_n) y Z_k (solución)

Si $x_m \neq y_n$ y $x_m \neq z_k$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_{m-1}, Y_n)

En forma similar

Si $x_m \neq y_n$ y $y_n \neq z_k$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_m, Y_{n-1})

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 *subsecuencia más larga (LCS)* entre X y Y

1) Caracterización de la estructura de la solución óptima (subestructura óptima)

Problema = LCS(X_m, Y_n) donde su solución es Z_k

Si $x_m = y_n$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_{m-1}, Y_{n-1}) + x_m

Si $x_m \neq y_n$ y $x_m \neq z_k$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_{m-1}, Y_n)

Si $x_m \neq y_n$ y $y_n \neq z_k$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_m, Y_{n-1})

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 *subsecuencia más larga (LCS)* entre X y Y

1) Caracterización de la estructura de la solución óptima (subestructura óptima)

Desde esta caracterización se puede definir una estrategia recursiva para solucionar el problema.

- 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 $LCS(X_m,Y_{n-1})$, esto es, de estas dos se elige 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 elige aquella de mayor longitud

•(ABD se refiere a <A,B,D>, ACD se refiere a <A,C,D>)

LCS(ABD, ACD)

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,)

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

Se obtiene AD como una solución

Programación dinámica

•¿Cuántos subproblemas, en general, tiene $LCS(X_m,Y_n)$?

Todos los subproblemas identificados son de la forma:

$$LCS(X_i,Y_j) \quad 0 \leq i \leq m, 0 \leq j \leq n$$

El máximo número de estos subproblemas sería $(m+1) \times (n+1)$

Con las soluciones de estos subproblemas sería suficiente para encontrar la solución óptima.

Programación dinámica

Note que hay subproblemas que se repiten, 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 debería calcular su solución más de una vez, ya que se conoce su solución.

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

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 *subsecuencia más larga (LCS)* entre X y Y

1) Caracterización de la estructura de la solución óptima (subestructura óptima)

Problema = LCS(X_m, Y_n) donde su solución es Z_k

Si $x_m = y_n$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_{m-1}, Y_{n-1}) + x_m

Si $x_m \neq y_n$ y $x_m \neq z_k$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_{m-1}, Y_n)

Si $x_m \neq y_n$ y $y_n \neq z_k$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_m, Y_{n-1})

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 X y Y

2) Definir recursivamente el valor de la solución óptima

Sea $c[i,j]$ el valor (costo) de la solución óptima del problema $LCS(X_i,Y_j)$

(Ejemplo: Considerando X= ABC Y=BE, c[2,1] valdría 1 ya que sería el valor de la solución del problema LCS(AB,B)

Programación dinámica

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

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

$$0 \leq i \leq m\,(|X|), 0 \leq j \leq n(|Y|)$$

Programación dinámica

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

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

$$
0 \leq i \leq m\ (|X|),\ 0 \leq j \leq n(|Y|)
$$

¿Cómo calcular c[i, j]? ¿En particular cómo calcular c[m, n]?

Programación dinámica

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

$$0 \le i \le m\,(|X|), 0 \le j \le n(|Y|)$$

¿Cómo calcular c[i, j]? ¿En particular, cómo calcular c[m, n]?

--- Estrategia recursiva (Similar a la primera solución de fibonacci).

--- Estrategia top-down Memoization (Similar a la de la segunda solución de fibonacci).

--- Estrategia bottom-up (Similar a la de la tercera solución de fibonacci).

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 *subsecuencia más larga (LCS)* entre X y Y

3) Calcular el valor de la solución óptima (Algoritmo)

--- ¿Estrategia recursiva (Similar a la primera solución de fibonacci)? ¡¡No!! Muy costosa, recalcula las soluciones a los subproblemas.

--- Estrategia top-down Memoization (Similar a la de la segunda solución de fibonacci).

--- Estrategia bottom-up (Similar a la de la tercera solución de fibonacci).

Programación dinámica

3) Calcular el valor de la solución óptima (Algoritmo)

Estrategia Bottom-up

Se calculan los valores desde los subproblemas más pequeños hasta los más grandes, finalizando con el del problema original.

LCS(ABCBDAB, BDCABA)

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

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							

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							

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	?				
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						

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

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						

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

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						

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

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						

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

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						

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

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						

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

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						

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

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						

Longitud para LCS(AB,B)

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

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						

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

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)

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

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

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

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
```

Algoritmo para Calcular el valor de la solución óptima

```
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
```

Costo Computacional Temporal

Calcular cada posición de la matriz cuesta $\theta(1)$

Calcular toda la matriz $\theta(m * n) = \theta(|X| * |Y|)$

Costo Computacional Espacial

$$\theta(m * n) = \theta(|X| * |Y|)$$

4) Construir una solución óptima (algoritmo)

```
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)
```

Algoritmo para Calcular una solución óptima

```
PRINT-LCS(b, X, m, n)
```

Llamado inicial

4) Construir la solución óptima (algoritmo)

```
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)
```

Costo Computacional Temporal

$$O(m+n) = O(|X| + |Y|)$$

```
PRINT-LCS(b, X, m, n)
```

Llamado inicial

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 *subsecuencia más larga (LCS)* entre X y Y

1) Caracterizar la estructura de la solución óptima (subestructura óptima).

Si $x_m = y_n$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_{m-1}, Y_{n-1}) + x_m

Si $x_m \neq y_n$ y $x_m \neq z_k$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_{m-1}, Y_n)

Si $x_m \neq y_n$ y $y_n \neq z_k$ entonces

La solución de LCS(X_m, Y_n) = la solución de LCS(X_m, Y_{n-1})

2) Definir recursivamente el valor de la solución óptima.

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

$$0 \le i \le m\ (|X|), 0 \le j \le n(|Y|)$$

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 *subsecuencia más larga (LCS)* entre X y Y

3) Calcular el valor de la solución óptima (algoritmo).

```
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 xi=yj
                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
```

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 *subsecuencia más larga (LCS)* entre X y Y

4) Calcular una solución óptima (algoritmo).

```
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)
```

Costo Computacional Estrategia
Costo de calcular el valor de la solución + Costo de calcular la solución $= \theta(|X| * |Y|) + O(|X| + |Y|)$
$= \theta(|X| * |Y|)$