Mayor subsecuencia común

Mayor subsecuencia común

Entrada: Dos cadenas $X = \{x_1, x_2, \ldots, x_M\}$, $Y = \{y_1, y_2, \ldots, y_N\}$ de algún alfabeto determinado (A-Z por ejemplo)

Salida: La mayor secuencia $Z = \{z_1, z_2, \ldots, z_k\}$ que es tanto sub-secuencia de *X* como de *Y*. Una cadena *A* es subsecuencia de *B* si todos los caracteres de *A* están en el mismo orden que en *B* aunque no necesariamente de forma consecutiva.

Ejemplo: cuál es la mayor sub-secuencia común de *X* = *ABCBDA*, *Y* = *BDCABA*

Sub-secuencias comunes:

A
B
…
AB
BC
…
BCA
BDA
…
BCBA

Solución mediante búsqueda exhaustiva

Para valores de *M* y *N* (supongamos que M>N), ¿Cuántas posibles soluciones (sub-secuencias) tiene la cadena menor? 2^N

Por tanto una solución por búsqueda exhaustiva sería generar un arreglo de *N* valores binarios, evaluar cada sub-secuencia de la cadena menor y luego verificar si hace parte de la cadena mayor.

$$M2^N$$

Solución mediante programación dinámica

¿Qué forma debería tener la solución óptima de un subproblema?

Dados los primeros *i* elementos de *X* y los *j* primeros elementos de *Y* la solución óptima para ese subproblema consistiría en una de las siguientes opciones:

1. Si $X_i = Y_j$, sumar uno a la solución óptima de los previos *i-1* elementos de *X* y los previos *j-1* elementos de Y

2. Si *Xi ≠ Yj* escoger entre la solución óptima de los *i* elementos de *X* y los previos *j-1* elementos de *Y* ó la solución óptima de los previos *i-1* elementos de *X* y los *j* elementos de *Y*

Solución mediante programación dinámica

```
for i = 0 to M:
    L_{i,0} = 0
for j = 0 to N:
    L_{0,j} = 0
 for i = 1 to M:
     for j = 1 to N:
         if Xi = Yj
             L_{i,j} = L_{i-1,j-1} + 1
         else
             L_{i,j} = MAX(L_{i-1,j}, L_{i,j-1})
```

¿Cuál es la eficiencia de este algoritmo? O(*MN*)

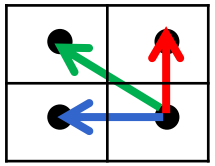
Ejemplo: X = *PUNTO*, Y = *PYTHON*

i / Caracteres de X \ j / Caracteres de Y		*P*	*Y*	*T*	*H*	*O*	*N*
	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
P 1	0	1	1	1	1	1	1
U 2	0	1	1	1	1	1	1
N 3	0	1	1	1	1	1	2
T 4	0	1	1	2	2	2	2
O 5	0	1	1	2	2	3	3

¿Cuál debe ser la solución para los casos base?, es decir, para $L_{i,0}$ y $L_{0,j}$? 0

Backtracking para obtener la subsecuencia

```
if Xi = Yj:
    se agrega Xi ó Yj
else:
    MAX(Li,j-1, Li-1,j)
```



i / Caracteres de X	j / Caracteres de Y		P	Y	T	H	O	N
		0	1	2	3	4	5	6
	0	0	0	0	0	0	0	0
P	1	0	1	1	1	1	1	1
U	2	0	1	1	1	1	1	1
N	3	0	1	1	1	1	1	2
T	4	0	1	1	2	2	2	2
O	5	0	1	1	2	2	3	**3**