ALGORÍTMICA

Práctica 4: Programación dinámica

Grupo Petazetas: Sara Bellarabi El Fazazi, Manuel Villatoro Guevara , Arturo Cortés Sánchez, Sergio Vargas Martin

1. Descripción del problema

Problema: Subsecuencia de caracteres más larga

El problema consiste en encontrar la máxima subsecuencia de caracteres común que aparecen en ambas cadenas de izquierda a derecha . Las subsecuencias no necesitan tener posiciones consecutivas en la secuencia original.

Eficiencia:

Un algoritmo de fuerza bruta para resolver la subsecuencia de caracteres más larga, es enumerar todas las posibles subsecuencias de x, controlar si también es subsecuencia de Y y recordar las más larga de ellas.
Este algoritmo es de eficiencia O(2 ^m) y por lo tanto no viable para valores de m grandes

Para el caso de dos secuencias de n y m elementos, el tiempo de ejecución para la programación dinámica es de O($n \times m$).

Teorema:
Sean X = (x1, x2,....xm) e Y=(y1, y2, ...yn). Luego si Z = (z1, z2,...zk) es subsecuencia común de X,Y

- xm = yn entonces Zk-1 es subsecuencia común de Xm-1, yn-1
- xm != ym y zk != xm entonces Z es subsecuencia común de Xm-1, Y
- xm != ym y zk != yn entonces Z es subsecuencia común de X, Yn-1

Para demostrarlo, se prueban los tres puntos por contradicción, llegando en todos los casos a mostrar que Z no es LCS de X, Y

El teorema mencionado antes, sugiere la siguiente recurrencia para resolverlo siendo C[i,j] la subsecuencia común de Xi, Yj

C[i,j] = { 0	si i=0 o j=0
C[i-1, j-1]+1	si i>0, j >0 y xi = yj
max(C[i-1,j], C[i,j-1])	si i>0, j>0 y xi != yj

Aquí, podemos ver que existe superposición de problemas

2. Pseudocódigo.

```
/*
EJECUCIÓN : ./subsecuencia <s1> <s2>
Se encarga de representar internamente dos secuencias de
strings en una matriz, y encontrar las subsecuencias que
coinciden en ambas.

Pseudocódigo

Idea( Asumir un string de secuencias como un vector <s1> <s2>
)
```

Coger primer elemento de <s1> y comparar con el resto de elementos de <s2> si coinciden representar en la Matriz dicha coincidencia(subsecuencia).
Llevando en todo momento un contador de las coincidencias encontradas
Así desde el primer elemento de <s1> hasta el último.

```
M[s1.size()+1][s2.size()+1] //Crear una matriz de incidencias de tamaño de las secuencias <s1+1> <s2+1>
for(i = 0 hasta s1.size()){
  for(j = 0 hasta s2.size()){
     if( i == 0 || j == 0) // La primera columna y fila son las de referencia (Comenzamos con 0 incidencias)
     M[i][j] = 0;
     else if(s1[i-1] == s2[j-1]) // Si coinciden el elemento de la <s1> con el de la <s2>, se coge el contador de incidencias y se suma +1 y se marca la incidencia en la posición (i,j) de la matriz
     M[i][j] = M[i - 1][j - 1] + 1;
     else
     M[i][j] = max(M[i - 1][j], M[i][j - 1]); //Si no coincide se escribe el (máximo) de la fila anterior y columna anterior, (pondrá el contador de incidencias) para así marcar que ya ocurrido una incidencia anteriormente
  }
}
```

```
Muestra_matriz_subsecuencias(M); //Muestra la matriz de incidencias
```

Para mostrar la subsecuencia se recorre desde desde la esquina inferior derecha
Buscando las incidencias, si coinciden las añado a mi string subsecuencia, que es lo que se devolverá.

Por ejemplo "hola" "adiol" contiene la subsecuencia "ol"

```
  0 0 0 0 0 0
  0 0 0 0 0 0
  0 0 0 0 1 1
  0 0 0 0 1 2
  0 1 1 1 1 2
```

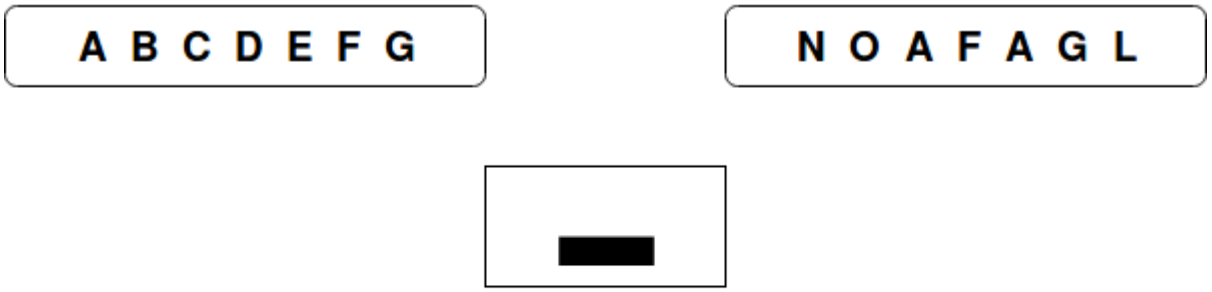
```
*/
```

3. Caso de ejecución.

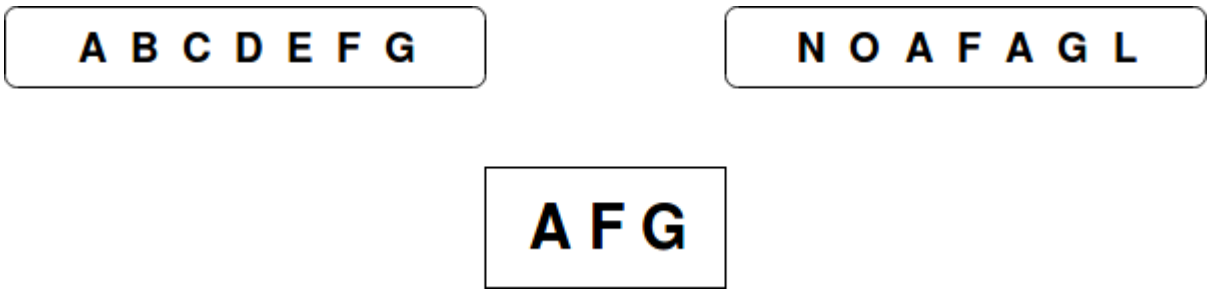
Como se ha planteado, el problema principal es encontrar la subcadena de caracteres común más larga en dos cadenas principales cualquiera. De esta forma, el procedimiento trata de comparar las cadenas principales, obteniendo una matriz de enteros. Estos enteros, indican la longitud del vector que contiene la subsolución en dicho punto del proceso.

Un caso de su uso sería el siguiente:

Dadas estas dos cadenas de caracteres,



el objetivo es obtener la subcadena de mayor longitud que sea común a ambas, en este caso sería:



Para obtener este resultado, se ha generado una matriz con el número de caracteres de la subcadena solución obtenida en ese momento. Para ello se elige el primer elemento de la

primera secuencia y se compara con el resto de elementos de la segunda subsecuencia. Si encontramos una coincidencia apuntamos el número de coincidencias en la matriz. Después de seguir este proceso obtenemos la siguiente matriz:

	Ø	N	O	A	F	A	G	L
Ø	0	0	0	0	0	0	0	0
A	0	0	0	1	1	1	1	1
B	0	0	0	1	1	1	1	1
C	0	0	0	1	1	1	1	1
D	0	0	0	1	1	1	1	1
E	0	0	0	1	1	1	1	1
F	0	0	0	1	2	2	2	2
G	0	0	0	1	2	2	3	3

Para obtener la subsecuencia se empieza a recorrer la matriz desde la esquina inferior derecha. Voy avanzando en la dirección del número más grande hasta que llegue a una posición en la que la letra de la columna y la de la fila coincidan. Cuando coincidan añado dicha letra a mi subsecuencia común y avanzo en diagonal. Repito el proceso hasta llegar a (0, 0).

	Ø	N	O	A	F	A	G	L
Ø	0	0	0	0	0	0	0	0
A	0	0	0	1	1	1	1	1
B	0	0	0	1	1	1	1	1
C	0	0	0	1	1	1	1	1
D	0	0	0	1	1	1	1	1
E	0	0	0	1	1	1	1	1
F	0	0	0	1	2	2	2	2
G	0	0	0	1	2	2	3	3

De esta forma obtenemos la solución AFG.