Práctica 4: Programación dinámica

Por: 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.)

String A	a	c	b	a	e	d
String B	a	b	c	a	d	f

Eficiencia: Este algoritmo es de eficiencia O(2 ^m).

- No es viable para valores grandes de m.
- Para el caso de dos secuencias de n y m elementos, el tiempo de ejecución para la programación dinámica es de O($n \times m$).

2. Pseudocódigo.

```
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;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
     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
```

3. Caso de ejecución.

1º Se compara cada carácter de una secuencia con cada uno de la otra secuencia.

2º Si encontramos una coincidencia apuntamos el número de coincidencias en la 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

2. Pseudocódigo.

```
for (int i = s1.size(), j = s2.size(); i > 0 && j > 0;) {
      if (s1[i - 1] == s2[j - 1]) {
            subsecuencia.insert(subsecuencia.begin(), s1[i - 1]);
            i--;
            j--;
      } else {
            M[i - 1][j] > M[i][j - 1] ? i-- : j--;
      }
}
```

En este trozo de código se recorre la matriz desde la esquina inferior derecha buscando las incidencias, si coinciden las añado al string de subsecuencia que es lo que se devolverá.

3. Caso de ejecución.

El objetivo es encontrar la subcadena de carácteres común más larga en dos cadenas.

- Compararemos las cadenas principales, obteniendo una matriz de enteros.
- Los enteros de la matriz indican la longitud del vector que contiene la subsolución en dicho punto del proceso

EJEMPLO:

Tenemos estas dos cadenas de caracteres:

obtenemos:

3. Caso de ejecución.

3º Recorremos la matriz desde la esquina inferior derecha, avanzando en la dirección del número más grande.

4º Cuando coincidan la letra de la fila y la columna, añadimos la letra a la subsecuencia común y avanzo en diagonal.

5º Se repite 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