

Subsecuencia más larga

1. Planteamiento del problema

La Programación Dinámica se aplica en cuatro fases:

1. Identificar la naturaleza n-etápica del problema
2. Verificación del Principio de Optimalidad de Bellman
3. Planteamiento de una recurrencia
4. Cálculo de una solución.

Naturaleza n-etápica del problema

Sea la subsecuencia más larga x_1, x_2, \dots, x_n ; esta subsecuencia es resultado de una serie de sucesiones ya que tenemos que decidir los valores de x_i , con $1 \leq i \leq n$. Así primero tomaríamos una decisión sobre x_1 , luego sobre x_2 , y así sucesivamente.

Por lo que podemos ver que estamos ante un problema de decisión n-etápico.

Principio de Optimalidad de Bellman

Si x_i es un elemento intermedio de la subsecuencia más larga, entonces la subsecuencia x_1, x_2, \dots, x_i es una solución optimal, y también lo es la subsecuencia x_i, x_{i+1}, \dots, x_n .

Planteamiento de una recurrencia

[AÑADIR TEXTO]

Cálculo de una solución

[AÑADIR TEXTO]

2. Pseudocódigo

Sean S1 y S2 las secuencias de las cuales queremos hallar la subsecuencia común más larga, el algoritmo se describiría por el siguiente pseudocódigo.

```
1  INICIO DEL ALGORITMO
2      cadena reconstruccion(matriz M, cadena S1, cadena S2,
3          ↪ entero i, entero j)
4  INICIO DE LA FUNCION
5      Si i o j son 0:
6          Devolver {}
7      Si no, si S1[i-1] es igual a S2[j-1]:
8          Devolver reconstruccion(M, S1, S2, i-1, j-1)
9      Si no, si M[i-1][j] es mayor que M[i][j-1]:
10         Devolver reconstruccion(M, S1, S2, i-1, j)
11     Si no:
12         Devolver reconstruccion(M, S1, S2, i, j-1)
13 FIN DE LA FUNCION
14
15 TAM1=|S1|
16 TAM2=|S2|
17 Crear una matriz M con TAM1 filas y TAM2 columnas
18 Rellenar la primera fila y la primera columna de M con
19 ↪ ceros
20 Repetir desde i=1 hasta TAM1:
21     Repetir desde j=1 hasta TAM2:
22         Si S1[i-1] es igual a S2[j-1]:
23             M[i][j]=M[i-1][j-1]+1
24         Si no:
25             M[i][j]=max(M[i-1][j],M[i][j-1])
26 Devolver reconstruccion(M, S1, S2, TAM1, TAM2)
27 FIN DEL ALGORITMO
```

NOTA: El algoritmo muestra al principio una función *reconstrucción* que es recursiva y a la cual se la llama en primer lugar en la línea 24 del código mostrado.



3. Eficiencia y ecuación recursiva

4. Código

Aquí se muestra el código utilizado escrito en lenguaje C++.

```
1 //Funcion de reconstruccion recursiva
2 string reconstruccion(vector <vector <int>> matriz, string
   ↪ a, string b, int i, int j){
3     if (i==0 || j==0)
4         return "";
5     else if (a[i-1] == b[j-1])
6         return reconstruccion (matriz, a, b, i-1, j-1) + a
   ↪ [i-1];
7     else if (matriz[i-1][j]>matriz[i][j-1])
8         return reconstruccion (matriz, a, b, i-1, j);
9     else
10        return reconstruccion (matriz, a, b, i, j-1);
11 }
12
13
14 //Funcion para hallar la subsecuencia mas corta con
   ↪ programacion dinamica
15 string subsecuencia(string a, string b){
16     int a_tam= a.size();
17     int b_tam= b.size();
18
19     //Creacion de la matriz con los valores y la matriz
   ↪ con las direcciones
20     vector <vector <int>> matriz(a_tam+1, vector <int> (
   ↪ b_tam+1, 0));
21
22     for (int i=1; i<a_tam+1; i++){
23         for (int j=1; j<b_tam+1; j++){
24             if (a[i-1]==b[j-1])
25                 matriz[i][j]= matriz[i-1][j-1]+1;
26             else{
27                 if (matriz[i-1][j]>matriz[i][j-1])
28                     matriz[i][j]= matriz[i-1][j];
29                 else
30                     matriz[i][j]= matriz[i][j-1];
31             }
32         }
33     }
34
35     //Return
36     return reconstruccion(matriz, a, b, a_tam, b_tam);
37 }
```



5. Escenarios de ejecución

Utilizando las secuencias *jacobocasadodegracia* y *jesusjosemariamaldonadoarroyo*, la sub-secuencia común más larga es *josadoaa*.

Aquí podemos ver la matriz de números que se construye.

```

      j e s u s j o s e m a r i a m a l d o n a d o a r r o y o
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
a 0 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
c 0 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
o 0 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3
b 0 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3
o 0 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 4 4 4 4
c 0 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 4 4 4 4
a 0 1 1 1 1 1 1 1 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 5 5 5 5
s 0 1 1 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 5 5 5 5
a 0 1 1 2 2 2 2 2 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5
d 0 1 1 2 2 2 2 2 3 3 3 3 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5
o 0 1 1 2 2 2 2 2 3 3 3 3 4 4 4 4 4 4 4 4 5 6 6 6 6 6 6 6 6
d 0 1 1 2 2 2 2 2 3 3 3 3 4 4 4 4 4 4 4 4 5 6 6 6 7 7 7 7 7 7
e 0 1 2 2 2 2 2 3 3 3 4 4 4 4 4 4 4 4 4 4 5 6 6 6 7 7 7 7 7 7
g 0 1 2 2 2 2 2 3 3 3 4 4 4 4 4 4 4 4 4 4 5 6 6 6 7 7 7 7 7 7
r 0 1 2 2 2 2 2 3 3 3 4 4 4 4 5 5 5 5 5 5 5 5 6 6 6 7 7 7 8 8 8
a 0 1 2 2 2 2 2 3 3 3 4 4 4 4 5 5 5 6 6 6 6 6 6 6 7 7 7 8 8 8 8
c 0 1 2 2 2 2 2 3 3 3 4 4 4 4 5 5 5 6 6 6 6 6 6 6 7 7 7 8 8 8 8
i 0 1 2 2 2 2 2 3 3 3 4 4 4 4 5 5 6 6 6 6 6 6 6 6 7 7 7 8 8 8 8
a 0 1 2 2 2 2 2 3 3 3 4 4 4 4 5 5 6 7 7 7 7 7 7 7 7 7 7 8 8 8 8

```

Y aquí podemos ver la matriz de direcciones que se construye.

```

      j e s u s j o s e m a r i a m a l d o n a d o a r r o y o
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 0 \ - - - - \ - - - - - - - - - - - - - - - - - - - -
a 0 | - - - - - - - - \ - - \ - \ - - - - \ - - \ - - -
c 0 | - - - - - - - - | - - - - - - - - - - - - - - - -
o 0 | - - - - - \ - - - - - - - - - - - \ - - - \ - - \
b 0 | - - - - - | - - - - - - - - - - - | - - - - - - - -
o 0 | - - - - - \ - - - - - - - - - - - \ - - - \ - - \
c 0 | - - - - - | - - - - - - - - - - - | - - - | - - - -
a 0 | - - - - - | - - - \ - - \ - \ - - - - \ - - \ - - -
s 0 | - \ - \ - - \ - - - - - - - - - - - | - - | - - - -
a 0 | - | - - - - | - - - \ - - \ - \ - - - - \ - - \ - - -
d 0 | - | - - - - | - - - | - - - - - - \ - - - \ - - - -
o 0 | - | - - - - \ - - - | - - - - - - | \ - - - \ - - - \
d 0 | - | - - - - | - - - | - - - - - - \ | - - - \ - - - -
e 0 | \ - - - - | - \ - - - - - - - - - | | - - - | - - - -
g 0 | | - - - - | - | - - - - - - - - - | | - - - | - - - -
r 0 | | - - - - | - | - - \ - - - - - - | - - - | - - \ - -
a 0 | | - - - - | - | - \ - - \ - \ - - - - \ - - \ - - -
c 0 | | - - - - | - | - | - - | - - - - - - | - - | - - - -
i 0 | | - - - - | - | - | - \ - - - - - - | - - | - - - -
a 0 | | - - - - | - | - \ - | \ - \ - - - - \ - - \ - - -

```



Vamos a observar otro ejemplo utilizando las secuencias de caracteres *ignaciosanchezherrera* y *juanmiguelhernandezgomez*, cuya subsecuencia común más larga es *ignaneze*.

```

      j u a n m i g u e l h e r n a n d e z g o m e z
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
i 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
g 0 0 0 0 0 0 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
n 0 0 0 0 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3
a 0 0 0 1 1 1 1 2 2 2 2 2 2 2 2 3 4 4 4 4 4 4 4 4
c 0 0 0 1 1 1 1 2 2 2 2 2 2 2 2 3 4 4 4 4 4 4 4 4
i 0 0 0 1 1 1 2 2 2 2 2 2 2 2 2 3 4 4 4 4 4 4 4 4
o 0 0 0 1 1 1 2 2 2 2 2 2 2 2 2 3 4 4 4 4 4 4 5 5 5
s 0 0 0 1 1 1 2 2 2 2 2 2 2 2 2 3 4 4 4 4 4 4 5 5 5
a 0 0 0 1 1 1 2 2 2 2 2 2 2 2 2 3 4 4 4 4 4 4 5 5 5
n 0 0 0 1 2 2 2 2 2 2 2 2 2 2 2 3 4 5 5 5 5 5 5 5 5
c 0 0 0 1 2 2 2 2 2 2 2 2 2 2 2 3 4 5 5 5 5 5 5 5 5
h 0 0 0 1 2 2 2 2 2 2 2 2 3 3 3 3 4 5 5 5 5 5 5 5 5 5
e 0 0 0 1 2 2 2 2 2 2 3 3 3 4 4 4 4 5 5 6 6 6 6 6 6 6
z 0 0 0 1 2 2 2 2 2 2 3 3 3 4 4 4 4 5 5 6 7 7 7 7 7 7
h 0 0 0 1 2 2 2 2 2 2 3 3 4 4 4 4 4 5 5 6 7 7 7 7 7 7
e 0 0 0 1 2 2 2 2 2 2 3 3 4 5 5 5 5 5 5 6 7 7 7 7 8 8
r 0 0 0 1 2 2 2 2 2 2 3 3 4 5 6 6 6 6 6 6 7 7 7 7 8 8
r 0 0 0 1 2 2 2 2 2 2 3 3 4 5 6 6 6 6 6 6 7 7 7 7 8 8
e 0 0 0 1 2 2 2 2 2 2 3 3 4 5 6 6 6 6 6 6 7 7 7 7 8 8
r 0 0 0 1 2 2 2 2 2 2 3 3 4 5 6 6 6 6 6 6 7 7 7 7 8 8
a 0 0 0 1 2 2 2 2 2 2 3 3 4 5 6 6 7 7 7 7 7 7 7 8 8

```

Y aquí podemos ver la matriz de direcciones que se construye.

```

      j u a n m i g u e l h e r n a n d e z g o m e z
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
i 0 - - - - - \ - - - - - - - - - - - - - - - -
g 0 - - - - - | \ - - - - - - - - - - - - - \ - -
n 0 - - - \ - - | - - - - - - \ - \ - - - - - - -
a 0 - - \ - - - | - - - - - - | \ - - - - - - - -
c 0 - - | - - - | - - - - - - | | - - - - - - - -
i 0 - - | - - \ - - - - - - | | - - - - - - - -
o 0 - - | - - | - - - - - - | | - - - - - \ - - -
s 0 - - | - - | - - - - - - | | - - - - - | - - -
a 0 - - \ - - | - - - - - - | \ - - - - - | - - -
n 0 - - | \ - - - - - - - \ | \ - - - - - - - -
c 0 - - | | - - - - - - - | | | - - - - - - - -
h 0 - - | | - - - - - \ - - - | | - - - - - - - -
e 0 - - | | - - - \ - - \ - - - | - \ - - - - \ -
z 0 - - | | - - - | - - | - - - | - | \ - - - - \
h 0 - - | | - - - | - \ - - - - | - | | - - - - -
e 0 - - | | - - - \ - | \ - - - - \ | - - - - \ -
r 0 - - | | - - - | - | | \ - - - - - | - - - | -
r 0 - - | | - - - | - | | \ - - - - - | - - - | -
e 0 - - | | - - - \ - | \ | - - - - \ - - - - \ -
r 0 - - | | - - - | - | | \ - - - - - | - - - | -
a 0 - - \ | - - - - | - | | | - \ - - - - - - | -

```