Algoritmos programación dinámica

Algoritmo: Envases de pintura

Planteamiento de la solución como sucesión de decisiones

- ➔ $x_1, x_2, \ldots, x_n$ tal que x_1 representa cuántos envases de tipo 1 se utilizan, x_2 cuántos envases de tipo 2 se utilizan…

- ➔ **PRINCIPIO DE OPTIMALIDAD DE BELLMAN:** sea C(i, j) el mínimo coste usando los i tipos de envases cuyo volumen satisface un pedido de j litros. El problema original es C(n, V), siendo n el número de tipos de envases diferentes y V el volumen total a repartir entre los envases.
- ➔ Si $x_1 \ldots x_n$ es óptima, hay que demostrar que $x_1 \ldots x_{n-1}$ es óptima para C(n-1, V - x_n* l_n), siendo x_n el número de envases de tipo n utilizado y l_n el volumen de los envases de tipo n.

- ➔ Por contradicción:

Si no fuese así entonces existe $y_1 \ldots y_{n-1}$ tal que $\sum_{i=1}^{n-1} x_i * coste[i] > \sum_{i=1}^{n-1} y_i * coste[i]$ y $\sum_{i=1}^{n} y_i * litros[i] \geq V - x_n * litros[n]$.

Pero entonces, haciendo $y_n = x_n$ tenemos que $\sum_{i=1}^{n} y_i * litros[i] \geq V$, luego $y_1 \ldots y_n$ es una solución para el problema original, y además $\sum_{i=1}^{n} x_i * coste[i] > \sum_{i=1}^{n} y_i * coste[i]$, luego $x_1 \ldots x_n$ no sería óptima.

Definición recursiva de la solución óptima

➔ Suponiendo que usamos *k* envases de tipo *i*, siendo *0 ≤ k ≤ unidades[i]*, definimos **C(i,j)** como:

$$C(i,j) = \min_{0 <= k <= v[i]} \begin{cases} \min\left(C(i-1,j),\ k \cdot c[i]\right) & \text{si} \quad k \cdot \ell[i] > j \\ C(i-1, j-k\cdot \ell[i]) + k \cdot c[i] & \text{en otro caso} \end{cases}$$

Calcular el valor de la solución óptima empleando un enfoque ascendente

Tamaño de la tabla → n x V, siendo n los tipos distintos de envase y V el volumen a repartir entre ellos.

Casos Base:

- ★ Para V = 0 ⇒ C(i, j) = 0.
- ★ Para n = 1 y V ≥ 1 ⇒ min { 0 ≤ k ≤ unidades[i] } (k * costes[i]) si k * litros[i] >= j
- ★ Para cada i, si $\sum_{i=0}^{i} unidades[i] * litros[i] < j \Rightarrow C(i, j) = +\infty$, es decir si con i recipientes nos sobran litros no habría solución.

Utilizamos también una segunda tabla para ir guardando en cada posición (i, j) el número de recipientes de tipo i que se usan para repartir j litros y obtener un coste de C(i, j).

Pseudocódigo del algoritmo

```
for i=0 to n
      C(i,0) = 0
done

for j = 1 to V do
      C(0,j) = +∞
      if(u[0] * l[0] > j)
            for k=1 to u[0] do
                  coste = k * c[0]
                  if(coste < C(0,j))
                        C(0,j) = coste
                        K(0,j) = K
                  endif
            done
       endif
done
```

```
sumaLitros = u[0] * l[0]
for i=1 to n do
      sumaLitros += u[i] * l[i]
      for j=1 to V do
            C(i,j) = +∞
            if(sumaLitros >=j)
                  for k=0 to u[i] do
                        if(K * l[i] >= j)
                              coste = C(i-1, i)
                        else
                              coste = C(i-1, j-k*l[i])
                        if(coste < C(i,j))
                              C(i,j) = coste
                              MejorK(i,j) = k
                        endif
                  done
            endif
      done
done
```

Calcular el valor de la solución óptima empleando un enfoque ascendente

Costes

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	5	5	5	10	10	10	15	15	15	+∞
2	0	5	5	5	9	9	10	14	14	15	18
3	0	5	5	5	9	9	10	12	14	15	17

Aplicado al ejemplo: V = 10

	capacidad l_i	número u_i	coste c_i
e_1	3	3	5
e_2	5	3	9
e_3	7	2	12

Mejor k

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	2	2	2	3	3	3	0
2	0	0	0	0	1	1	0	1	1	1	2
3	0	0	0	0	0	0	0	1	0	0	1

Tabla MejorK(i,j) : almacena el número de envases usados de cada tipo

Determinación de la solución óptima

```
costeOptimo = C(n,V)
mejorK (K, n, V, solucion)
```

```
mejorK (vector<vector<int>> K, int i, int V, vector<int> solución){
    int volumenActual = V - K(i,V) * l[i]
    solucion [i] = K(i,V)
    if (i>0 and volumenActual > 0)
        mejorK (K, i-1, volumenActual, solucion)
}
```

→ Con esta función recursiva se obtiene el vector solución
→ El vector solución consiste en una sucesión de decisiones obtenida a partir de la tabla en la que guardamos mejorK.

Determinación de la solución óptima

Costes

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	5	5	5	10	10	10	15	15	15	+∞
2	0	5	5	5	9	9	10	14	14	15	18
3	0	5	5	5	9	9	10	12	14	15	17

Aplicado al ejemplo: V = 10

	capacidad l_i	número u_i	coste c_i
e_1	3	3	5
e_2	5	3	9
e_3	7	2	12

Mejor k

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	2	2	2	3	3	3	0
2	0	0	0	0	1	1	0	1	1	1	2
3	0	0	0	0	0	0	0	1	0	0	1

C(n,V) = coste óptimo del algoritmo = 17€

1 envase de tipo 1 (3L -› 5€)
1 envase de tipo 3 (7L -› 12€)

EFICIENCIA TEÓRICA DEL ALGORITMO IMPLEMENTADO

Eficiencia Teórica de la función recursiva

```
mejorK (vector<vector<int>> K, int i, int V, vector<int> solución){
    int volumenActual = V - K(i,V) * l[i]
    solucion [i] = K(i,V)
    if (i>0 and volumenActual > 0)
        mejorK (K, i-1, volumenActual, solucion)
}
```

$$T(n) = T(n-1) + 1$$

$$T_n - T_{n-1} = 1$$

$$x - 1 = 1 \longrightarrow 1^n \cdot n$$

$$(x-1)(x-1) \Longrightarrow (x-1)^2$$

$$(A_n + B) \cdot 1^n = 0$$

$$A_n + B = 0 \Longrightarrow O(n)$$

```
for i = 0 to n
    C(i,0) = 0
```
O(n)

```
for j=1 to V do
    C(1,j) = +∞
    if (u[i]·l[i] >j)
        for K=0 to u[1] do
            coste = K·c[i]
            if(coste< C(1,j)) {
                C(1,j) = coste
                K(1,j) = K
            }
        done
done
```
U[i]

V * U[i]

```
sumaLitros = u[0]·l[0]
for i=1 to n do
    sumaLitros+= u[i] l[i]
    for j=1 to V do
        C(i,j) = +∞
        if ( sumaLitros >= j ){
            for K=0 to u[i] do
                if (K·l[i] >= j)
                    coste = C(i-1,i)
                else
                    coste = C(i-1,j-K·l[i]) + K·c[i];
                if (coste < C(i,j)
                    C(i,j) = coste;
                    MejorK(i,j) = K
                endif
            done
        done
done
```
Max(u[i])

Max(u[i]) * V

Max(u[i]) * V * n

Coste Óptimo = C(n,V) ⟹ O(1)

mejorK(K, n, V, solución) ⟹ O(n)

Eficiencia resultante: O(V · N · U) siendo V el volumen a repartir, N el número de tipos de envases distintos y U el máximo número de unidades de un envase

PRUEBAS REALIZADAS Y RESULTADOS OBTENIDOS

N = 3, V = 10

```
Litros[1] = 3   Unidades[1] = 3 Costes[1] = 5
Litros[2] = 5   Unidades[2] = 3 Costes[2] = 9
Litros[3] = 7   Unidades[3] = 2 Costes[3] = 12
0   5   5   5   10  10  10  15  15  15  2147483647
0   5   5   5   9   9   10  14  14  15  18
0   5   5   5   9   9   10  12  14  15  17

0   1   1   1   2   2   2   3   3   3   0
0   0   0   0   1   1   0   1   1   0   2
0   0   0   0   0   0   0   1   0   0   1
Mejor Coste = 17
Solucion:
   Envase 1 --> Numero de envases usados 1
   Envase 2 --> Numero de envases usados 0
   Envase 3 --> Numero de envases usados 1
```

N = 4, V = 10

```
Litros[1] = 4   Unidades[1] = 4 Costes[1] = 5
Litros[2] = 9   Unidades[2] = 1 Costes[2] = 1
Litros[3] = 9   Unidades[3] = 2 Costes[3] = 13
Litros[4] = 7   Unidades[4] = 5 Costes[4] = 9
0       5       5       5       5       10      10      10      10      15      15
0       1       1       1       1       1       1       1       1       1       6
0       1       1       1       1       1       1       1       1       1       6
0       1       1       1       1       1       1       1       1       1       6

0       1       1       1       1       2       2       2       2       3       3
0       1       1       1       1       1       1       1       1       1       1
0       0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0       0
Mejor Coste = 6
Solucion:
   Envase 1 --> Numero de envases usados 1
   Envase 2 --> Numero de envases usados 1
   Envase 3 --> Numero de envases usados 0
   Envase 4 --> Numero de envases usados 0
```

N = 6, V = 10

```
Litros[1] = 10  Unidades[1] = 3 Costes[1] = 19
Litros[2] = 2   Unidades[2] = 2 Costes[2] = 1
Litros[3] = 9   Unidades[3] = 2 Costes[3] = 17
Litros[4] = 9   Unidades[4] = 8 Costes[4] = 15
Litros[5] = 2   Unidades[5] = 2 Costes[5] = 1
Litros[6] = 4   Unidades[6] = 2 Costes[6] = 7
0       19      19      19      19      19      19      19      19      19      19
0       1       1       2       2       19      19      19      19      19      19
0       1       1       2       2       17      17      17      17      17      18
0       1       1       2       2       15      15      15      15      15      16
0       1       1       2       2       3       3       4       4       15      16
0       1       1       2       2       3       3       4       4       10      10

0       1       1       1       1       1       1       1       1       1       1
0       1       1       2       2       0       0       0       0       0       0
0       0       0       0       0       1       1       1       1       1       1
0       0       0       0       0       1       1       1       1       1       1
0       0       0       0       0       1       1       2       2       0       0
0       0       0       0       0       0       0       0       0       1       1
Mejor Coste = 10
Solucion:
   Envase 1 --> Numero de envases usados 0
   Envase 2 --> Numero de envases usados 2
   Envase 3 --> Numero de envases usados 0
   Envase 4 --> Numero de envases usados 0
   Envase 5 --> Numero de envases usados 1
   Envase 6 --> Numero de envases usados 1
```

N = 4, V = 15

```
Litros[1] = 12  Unidades[1] = 6 Costes[1] = 13
Litros[2] = 4   Unidades[2] = 4 Costes[2] = 5
Litros[3] = 10  Unidades[3] = 2 Costes[3] = 11
Litros[4] = 11  Unidades[4] = 8 Costes[4] = 19
0       13      13      13      13      13      13      13      13      13      13      13      13      26      26      26
0       5       5       5       5       10      10      10      10      13      13      13      13      18      18      18
0       5       5       5       5       10      10      10      10      11      11      13      13      16      16      18
0       5       5       5       5       10      10      10      10      11      11      13      13      16      16      18

0       1       1       1       1       1       1       1       1       1       1       1       1       2       2       2
0       1       1       1       1       2       2       2       2       0       0       0       0       1       1       1
0       0       0       0       0       0       0       0       0       1       1       0       0       1       1       0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
Mejor Coste = 18
Solucion:
   Envase 1 --> Numero de envases usados 1
   Envase 2 --> Numero de envases usados 1
   Envase 3 --> Numero de envases usados 0
   Envase 4 --> Numero de envases usados 0
```