Ejercicios Programación Dinámica

4. (2 puntos) Tenemos que pagar la cuenta en un restaurante por valor de M euros, y disponemos de una cantidad ilimitada de monedas de n tipos diferentes, siendo $c[i]$ el valor de cada moneda de tipo i, $i = 1, \ldots, n$. Tenemos prisa y no queremos esperar que, en su caso, nos den la vuelta, pero como no hemos quedado muy satisfechos queremos pagar o la cantidad exacta M o la mínima cantidad posible mayor que M (o sea dejar ninguna propina o la mínima propina posible de acuerdo a las monedas que tenemos). Diseñar un algoritmo eficiente que determine la cantidad mínima que tenemos que pagar y cómo hacerlo.

   Aplicadlo para resolver el siguiente caso: hay $n = 3$ tipos de monedas de valores 5, 7 y 13, y queremos pagar una cantidad de $M = 11$ unidades.

n monedas diferentes, el tipo i con el valor c_i

Queremos pagar M o el mínimo de más

Sea $mc(i,j)$ el mínimo de más que hay que dejar con las primeras i tipos de moneda para una cantidad j. Nuestro problema será $mc(n,M)$. Vemos que:

$mc(i,0) = 0 \quad \forall i \geq 0 \qquad mc(1,j) = K c_1 - j$ siendo $K = \inf\{a : a c_1 \geq j\} \quad \forall j \geq 0$

i \ j	0	1	2	3	4
1	0	$c_1 - 1$	$mc(1,2)$	$mc(1,3)$	$mc(1,3)$
2	0	$mc(2,1)$			
3	0				
4	0				

$$mc(1,2) = \begin{cases} mc(1,1) - 1 \\ mc(1,1) + c_1 - 1 \end{cases}$$

$$mc(1,3) = \begin{cases} mc(1,2) - 1 \\ mc(1,2) + c_1 - 1 \end{cases}$$

$$mc(2,1) = \begin{cases} mc(1,1) \\ mc(2,0) + c_2 - 1 \end{cases}$$

$$mc(2,2) = \begin{cases} mc(2,1) - 1 \\ mc(1,2) \\ mc(2,1) + c_2 - 1 \\ mc(2,1) + c_1 - 1 \end{cases}$$

$mc(i,0) = 0$

$$mc(i,j) = \begin{cases} mc(i-1,j) \\ mc(i,j-1) - 1 \\ mc(i,j-c_i) \rightarrow +1 \text{ moneda } i \\ c_i = j \quad (\text{si } j - c_i > 0) \\ \downarrow \\ +1 \text{ de moneda } i \end{cases}$$

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	4	3	2	1 (+1 de c_1)	0	4	3	2	1	0	4	3	2
2	0	4	3	2	1	0	1	0	2	1	0	1 (+1 de c_2)	0	1
3	0	4	3	2	1	0	1	0	2	1	0	1	0	

2. (2 puntos) Para estudiar un examen de n temas disponemos de T días. Conocemos el número de días completos que necesitamos para estudiar cada uno de los temas $(t_1, t_2, \ldots, t_n)$. Se pide diseñar un algoritmo de Programación Dinámica que permita estudiarnos el máximo número posible de temas antes de que llegue la fecha del examen.

Igual que mochila 0/1

$MC(i,j)$ máximo de temas entre los i primeros en j días. $MC(n,T)$

i \ j	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	1	1	1
2	0					
3	0					
4	0					

$$MC(1,j) = \begin{cases} 0 & t_1 > j \\ 1 & t_1 \leq j \end{cases}$$

$$MC(2,j) = \begin{cases} MC(1,j) \\ MC(1,j) + 1 \end{cases}$$

$$MC(i,j) = \begin{cases} MC(i-1,j) \\ MC(i-1,j-t_i) + 1 \end{cases}$$

4. (2 puntos) Disponemos de n tipos de monedas, las monedas de tipo i tienen un valor de $v[i]$. La cantidad de monedas disponibles de tipo i es igual a $c[i]$. Se quiere devolver una cantidad exacta M utilizando el menor número posible de monedas. Diseñad un algoritmo de programación dinámica que determine el número óptimo de monedas a usar. Aplicadlo para resolver el siguiente caso del problema, con $n = 3$, construyendo la tabla correspondiente:

$$v[\,] = (2, 4, 6),\ c[\,] = (3, 1, 2),\ M = 8$$

	0	2	4	6	8	10
0	0	∞	∞	∞	∞	∞
1	0	1	2	3	∞	∞
2	0	1	1	2	3	∞
3	0	1	1	1	2	∞
	0					
	0					

$$mC[i,j] = \min\{\, mC[i-1,j],\ mC[i,j-K*v[i]] + K \,\}$$
$$\text{tq } j - K*v[i] \geq 0$$
$$K = 0, 1, \ldots c[i]$$

$$mC[2,10] = \min\{\, mC[1,10], mC[1,6] + 1 \,\} = 4$$