Cambio de monedas con programación dinámica

Cambio de monedas con programación dinámica

Entrada
Dada una serie de denominaciones (valores enteros positivos) $d_1, d_2, \ldots d_N$ ordenados ascendentemente que compone el sistema monetario y una cantidad entera positiva *M*.

Salida
Serie r_1, r_2, r_N tal que $\sum_{i=1}^{n} r_i * d_i = M$ minimizando $\sum_{i=1}^{N} r_i$

Ejemplo: d = {11, 5, 1} y M = 15

Solución mediante programación dinámica

¿Qué forma debería tener la solución óptima de un subproblema?

Considerando las denominaciones en orden ascendente, y dados las primeras *i* denominaciones y una cantidad *j* a cambiar, la solución óptima $A_{i,j}$ para ese subproblema consistiría en elegir una de las dos siguientes opciones:

1. Emplear una unidad de la denominación *i*, sumar *1* a la mínima cantidad de monedas obtenida al restarle *i* a *j*

2. No emplear una unidad de la denominación *i*, y quedarse con la mínima cantidad de monedas obtenida con las *i-1* denominaciones restantes

Solución mediante programación dinámica

```
for i = 0 to N:
   A_{i,0} = 0
for j = 1 to M:
   A_{0,j} = inf
for i = 1 to N:
   for j = 1 to M:
      if d_i <= j:
         k = j - d_i
         A_{i,j} = MIN(1 + A_{i,k}, A_{i-1,j})
      else:
         A_{i,j} = A_{i-1,j}
print(A_{N,M})
```

¿Cuál es la eficiencia de este algoritmo?

O(*NM*), peor que O(*N*) de la solución greedy, pero mucho mejor la complejidad exponencial de la solución por búsqueda exhaustiva

Ejemplo: para d ={1, 4, 6}, *M* = 8

j Cantidad a cambiar

i Denominaciones que puedo escoger

	0	1	2	3	4	5	6	7	8
0	0	∞	∞	∞	∞	∞	∞	∞	∞
1	0	1	2	3	4	5	6	7	8
4	0	1	2	3	1	2	3	4	2
6	0	1	2	3	1	2	1	2	2