PROBLEMA DEL CAMBIO

UNIVERSIDAD COMPLUTENSE MADRID

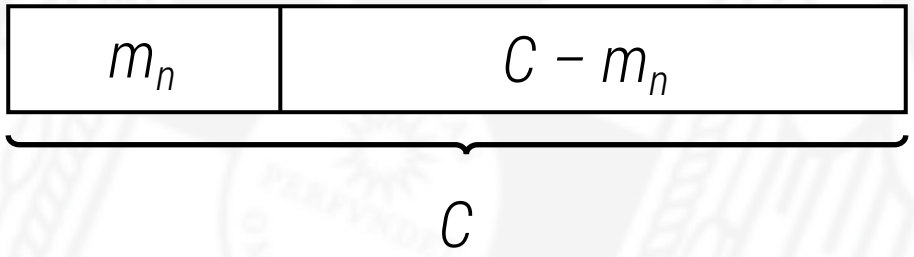
ALBERTO VERDEJO

Problema del cambio de monedas

- Conjunto finito $M = \{ m_1, m_2, \ldots, m_n \}$ de tipos de monedas, donde cada m_i es un número natural.
- Existe una cantidad ilimitada de monedas de cada valor.
- Se quiere pagar una cantidad $C > 0$ utilizando el menor número posible de monedas.

Problema del cambio de monedas

- Al no funcionar una estrategia voraz, tenemos que considerar diferentes alternativas hasta encontrar la mejor.
- Las soluciones son multiconjuntos de monedas.
- Podemos fijar el orden en el que vamos considerando los tipos de monedas, sin que eso afecte al resultado final.
- ¿Qué hacemos con las monedas de tipo n?



Problema del cambio de monedas

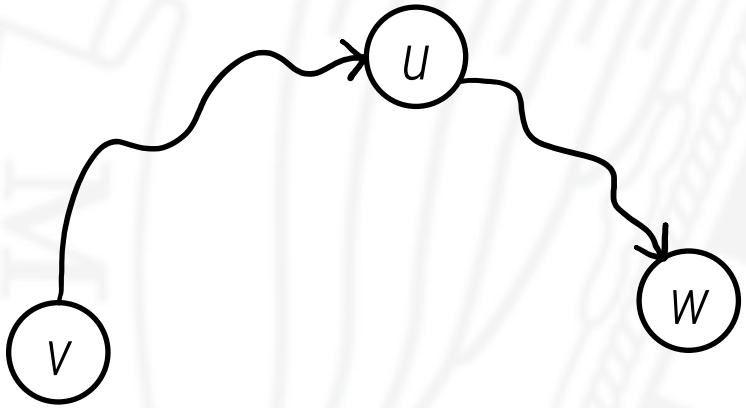
$$monedas(i,j) = \text{número } \textit{mínimo} \text{ de monedas para pagar la cantidad } j \text{ considerando los tipos de monedas del 1 al } i$$

- Principio de optimalidad de Bellman: para conseguir una solución óptima basta con considerar subsoluciones óptimas.

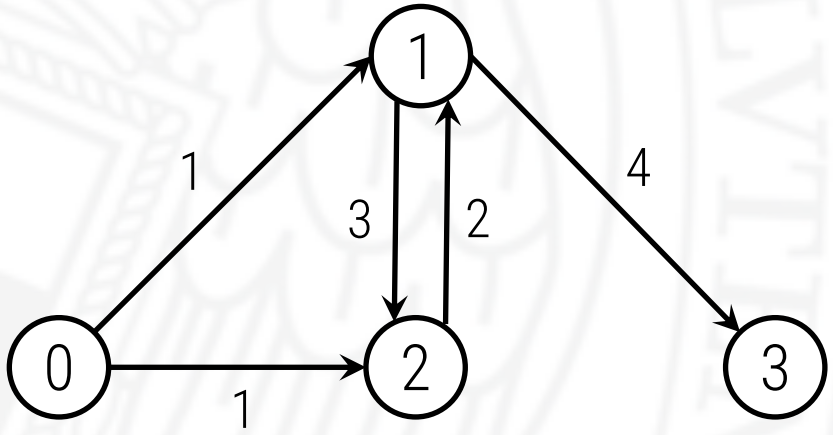
Principio de optimalidad de Bellman

- El principio de optimalidad de Bellman se cumple en un problema si una solución óptima a una instancia del problema siempre contiene soluciones óptimas a todas sus subinstancias.

Camino mínimo



Camino simple más largo



Definición recursiva

- Casos recursivos:

$$monedas(i,j) = \begin{cases} monedas(i-1,j) & \text{si } m_i > j \\ \text{mín}(monedas(i-1,j), monedas(i,j-m_i)+1) & \text{si } m_i \leq j \end{cases}$$

donde $1 \leq i \leq n$ y $1 \leq j \leq C$

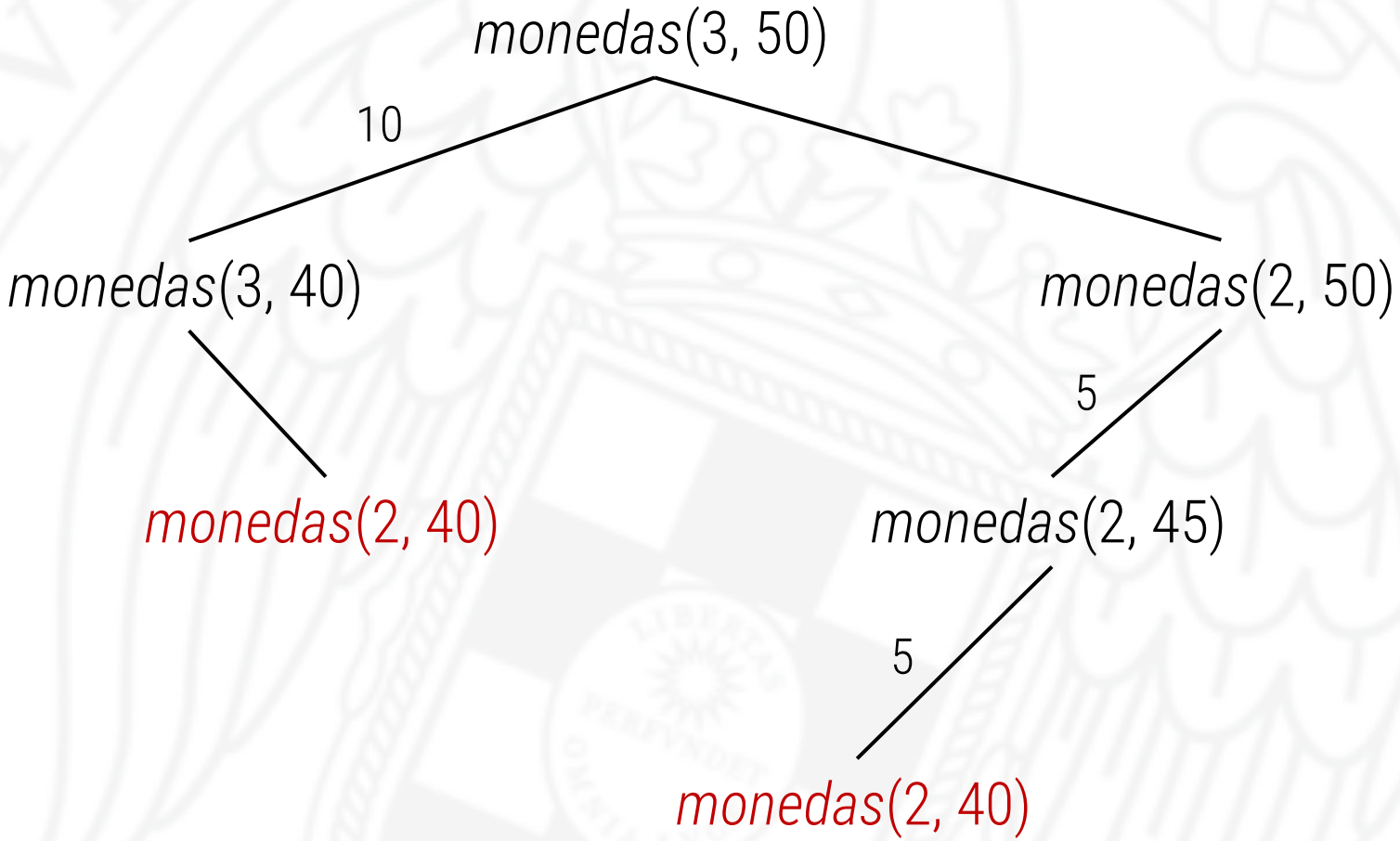
- Casos básicos:

$$\begin{aligned} monedas(i,0) &= 0 && 0 \leq i \leq n \\ monedas(0,j) &= +\infty && 1 \leq j \leq C \end{aligned}$$

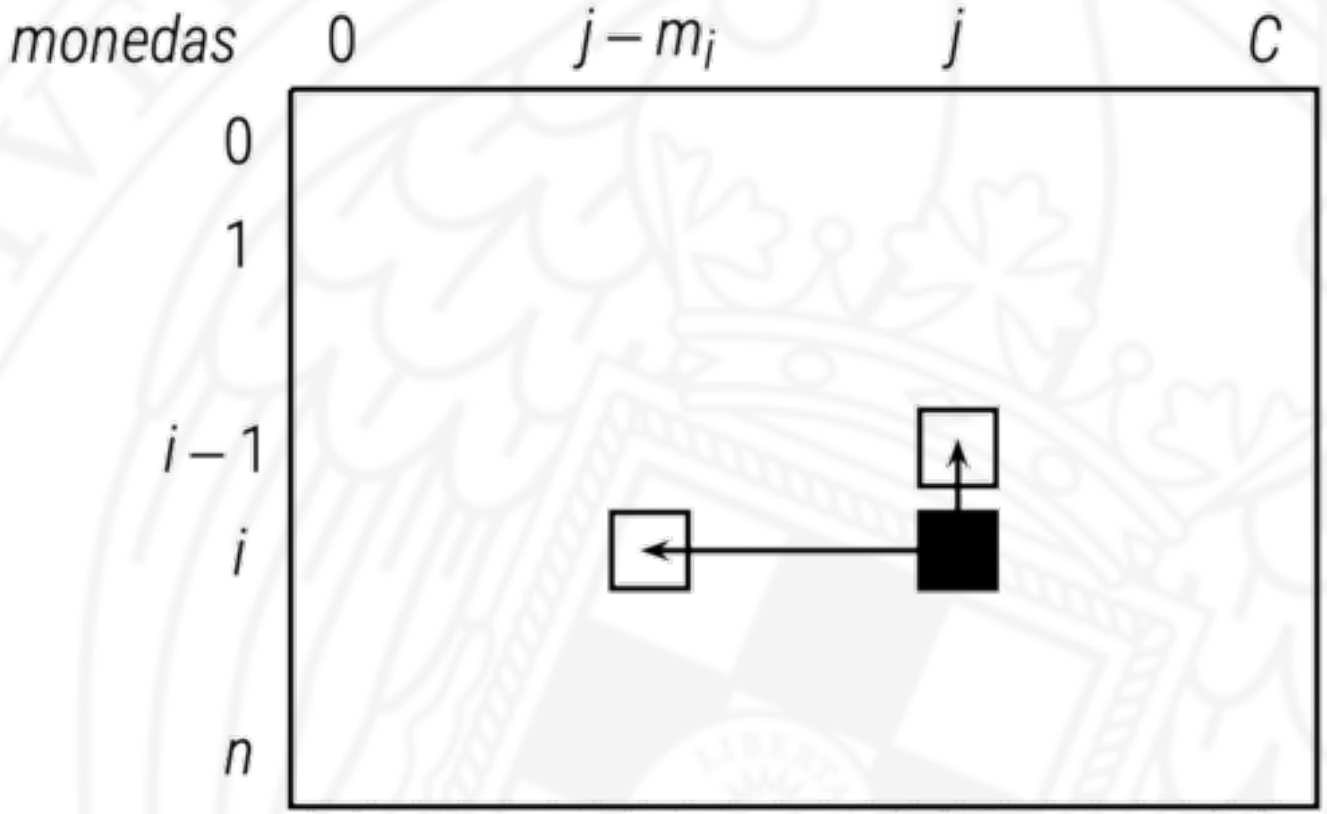
- Llamada inicial: *monedas*(*n*, *C*)

Subproblemas repetidos

- $m_3 = 10, m_2 = 5$ (entre otras)



Tabla



Cómo tratar el infinito

$$x + \infty = \infty \qquad x < \infty$$

- `std::numeric_limits<T>` proporciona información sobre los tipos numéricos.
- `has_infinity` dice si el tipo T tiene valor infinito o no. Y la función `infinity()` lo devuelve, si existe.
- Podemos definir nuestra clase `EntInf` con la funcionalidad deseada. En particular, una constante `Infinito` que represente el infinito.

Implementación

$$monedas(i,j) = \begin{cases} monedas(i-1,j) & \text{si } m_i > j \\ \min(monedas(i-1,j), monedas(i,j-m_i)+1) & \text{si } m_i \leq j \end{cases}$$

```
EntInf devolver_cambio(vector<int> const& M, int C) {
  int n = M.size();
  Matriz<EntInf> monedas(n+1, C+1, Infinito);
  monedas[0][0] = 0;
  for (int i = 1; i <= n; ++i) {
    monedas[i][0] = 0;
    for (int j = 1; j <= C; ++j)
      if (M[i-1] > j)
        monedas[i][j] = monedas[i-1][j];
      else
        monedas[i][j] = min(monedas[i-1][j], monedas[i][j - M[i-1]] + 1);
  }
  return monedas[n][C];
}
```

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

	0	1	2	3	4	5	6	7	8
0	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	0								
2	0								
3	0								

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

	0	1	2	3	4	5	6	7	8
0	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	0	1	2	3	4	5	6	7	8
2	0								
3	0								

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

	0	1	2	3	4	5	6	7	8
0	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	0	1	2	3	4	5	6	7	8
2	0	1	2	3	1	2	3	4	2
3	0								

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Ejemplo

$C = 8, n = 3$

$m_1 = 1, m_2 = 4$ y $m_3 = 6$

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

Reconstrucción de la solución óptima

$$monedas(i,j) = \text{mín}(\underbrace{monedas(i-1,j)}_{\text{no cogemos moneda } m_i}, \underbrace{monedas(i,j-m_i)+1}_{\text{sí cogemos moneda } m_i})$$

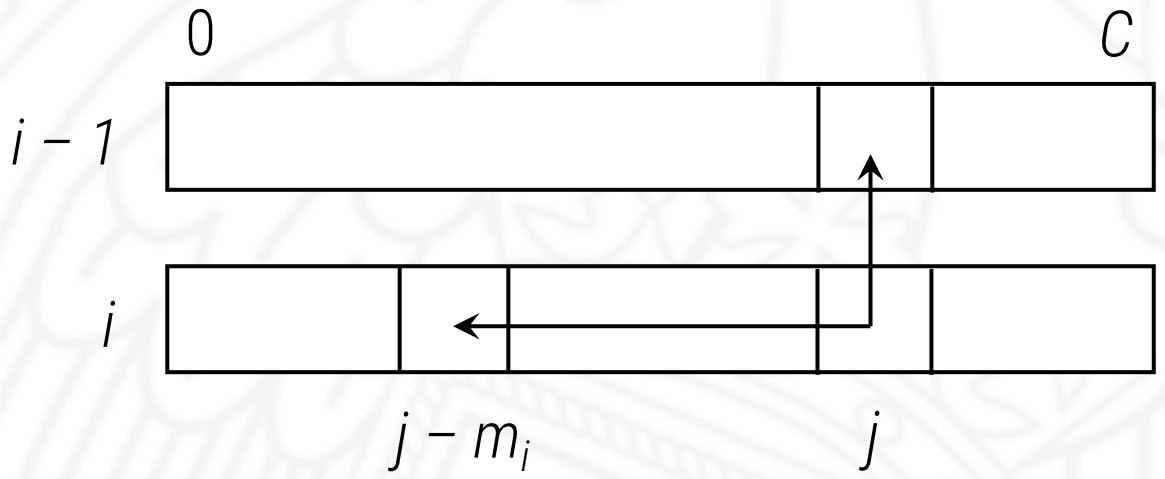
- Sabemos que si $monedas(i,j) = monedas(i-1,j)$ es porque podemos no coger monedas de tipo i para pagar la cantidad j.
- Mientras que si $monedas(i,j) \neq monedas(i-1,j)$ debemos coger al menos una moneda de tipo i para pagar j.

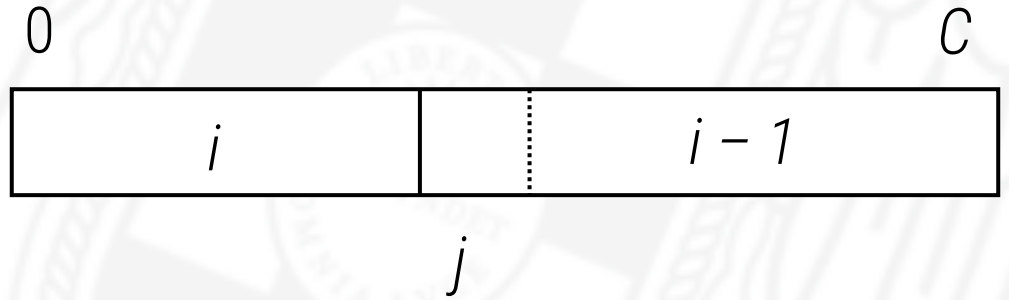
Implementación

```
vector<int> devolver_cambio(vector<int> const& M, int C) {

      ... // rellenar la matriz monedas como antes

    vector<int> sol;
    if (monedas[n][C] != Infinito) {
       int i = n, j = C;
       while (j > 0) {  // no se ha pagado todo
           if (M[i-1] <= j && monedas[i][j] != monedas[i-1][j]) {
               // tomamos una moneda de tipo i
               sol.push_back(M[i-1]); j = j - M[i-1];
           } else // no tomamos más monedas de tipo i
               --i;
       }
    }
    return sol;
}
```

Mejorar espacio adicional

- ¿Podemos reducir la memoria necesaria?





Mejorar espacio adicional

- ¿Y podemos seguir reconstruyendo la solución?
- La última fila contiene la información sobre el número de monedas mínimo necesario para cada cantidad, con el sistema monetario completo.

$$monedas(n,j) = monedas(n,j - m_i) + 1$$

- Al tener un número ilimitado de monedas de cada tipo, el sistema monetario no cambia al ir gastando monedas.

Implementación

```
vector<int> devolver_cambio(vector<int> const& M, int C) {
    int n = M.size();
    vector<EntInf> monedas(C+1, Infinito);
    monedas[0] = 0;
    // calcular la matriz sobre el propio vector
    for (int i = 1; i <= n; ++i) {
        for (int j = M[i-1]; j <= C; ++j) {
            monedas[j] = min(monedas[j], monedas[j - M[i-1]] + 1);
        }
    }
```

Implementación

```
    vector<int> sol;
    if (monedas[C] != Infinito) {
        int i = n, j = C;
        while (j > 0) {  // no se ha pagado todo
            if (M[i-1] <= j && monedas[j] == monedas[j - M[i-1]] + 1) {
                // tomamos una moneda de tipo i
                sol.push_back(M[i-1]);
                j = j - M[i-1];
            } else // no tomamos más monedas de tipo i
                --i;
        }
    }
    return sol;
}
```