17. En el *problema del vuelto* tenemos una cantidad ilimitada de monedas de distintos valores $w_1, \ldots, w_k$ y queremos dar un vuelto v utilizando la menor cantidad de monedas posibles (ver Teórica 2). Por ejemplo, si los valores son $w_1 = 1$, $w_2 = 5$, y $w_3 = 12$ y el vuelto es $v = 15$, entonces el resultado es 3 ya que alcanza con dar 3 monedas de \$5. Modelar este problema como un problema de camino mínimo e indicar un algoritmo eficiente para resolverlo. El algoritmo sobre el modelo debe tener complejidad $O(vk)$. **Opcional:** discutir cómo se relaciona este modelo con el algoritmo de programación dinámica correspondiente.

Podemos modelarlo tal que:

Armamos un $G = (V, E)$

Cada posible vuelto $v_i = i, \forall 0 \leq i \leq v$ es un nodo de V

Y tenemos $v_i \to \left(v_{i+w_i}\right) \in E$ con $c(u \to w) = 1, \forall (u \to w) \in E$

Nota: *En el ejemplo podemos ver que el camino minimo es* $v_0 \underset{+1}{\to} v_5 \underset{+1}{\to} v_{10} \underset{+1}{\to} v_{15}$

Una vez tenemos el grafo armado, notamos que es un DAG, luego podemos recorrerlo de forma lineal:

```
ady = lista de adyacencias de G //viene dado o armarla es O(n+m)

pred = []*v

//esto es O(n+m)
for u in ady:
  for v in ady[u]:
    pred[v].add(u)

//O(n+m), es un toposort
ord = posorder(DFS(G))

memo = [inf]*v //O(v)
memo[0]=0

f rec(v)
  si v = 0:
    ret 0

  si memo[v] != inf:
    ret memo[v]

  para u en pred[v]:
    memo[v] = min(1+rec(u),memo[v])

  ret memo[v]

rec(v) es el resultado //O(v) (está acotado)
```

Luego nos queda $O(vk + (n+m) + v) \in O(vk + (n+m))$ y $n = v, m = k \Rightarrow$ es $O(vk + v + k) \in O(vk)$