16. Sea D un digrafo sin ciclos dirigidos en el que todo vértice es alcanzable desde un vértice v. Sea $c\colon E(D) \to \mathbb{Z}$ una función de pesos.

a. Definir una función recursiva $d\colon V(D) \to \mathbb{Z}$ tal que $d(w)$ es el peso del camino mínimo de v a w para todo $w \in V(D)$. **Ayuda:** considerar que el camino mínimo de v a w se obtiene yendo de v hacia z y luego tomando la arista $z \to w$, para algún vecino de entrada z de w; notar que la función recursiva está bien definida porque D no tiene ciclos.

b. Diseñar un algoritmo de programación dinámica top-down para el problema de camino mínimo en digrafos sin ciclos y calcular su complejidad.

c. (Integrador y opcional) Diseñar un algoritmo de programación dinámica bottom-up para el problema. **Ayuda:** computar d de acuerdo a un orden topológico $v = v_1, \ldots, v_n$ donde $v_i \to v_j$ solo si $i < j$. Este orden se puede computar en $O(n+m)$ (guía 3).

a

$$d(w) = \begin{cases} 0 & \text{si } w = v \\ \min_{(z\to w)\in E}(d(z) + c(z \to w)) & \text{sino} \end{cases}$$

b

```
pred = [[]*n]
para cada (u,v) en E(D): //O(m)
  pred[v].add(u)

memo = [null]*n
f rec(w):
  si w ≡ v:
    ret 0

  si memo[w] es null:
    memo[w] = min(d(z)+c(z,w) para cada z en pred[w]) //O(n) acotado por estados

  return memo[w]

  //O(n+m)
```

c

```
ady = lista de adyacencias de E

ord = reverse(dfs(v,E)) //toposort O(n+m)

memo [[null]*n]
memo[v] = 0

para u en ord: //O(n)
  para z in ady[u]: //O(d(u))
    memo[z] = min(memo[z],memo[u]+c(u,z))

ret memo[w]


//O(n+m)
```