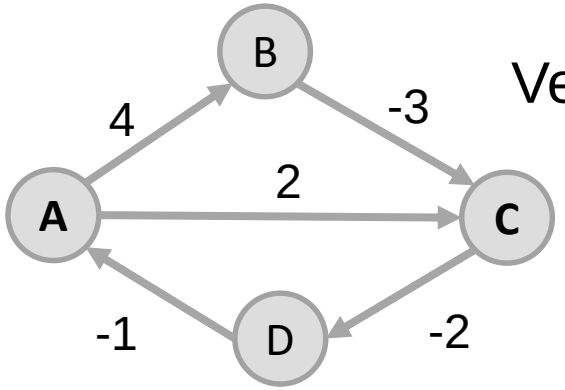
Algoritmo de Bellman-Ford

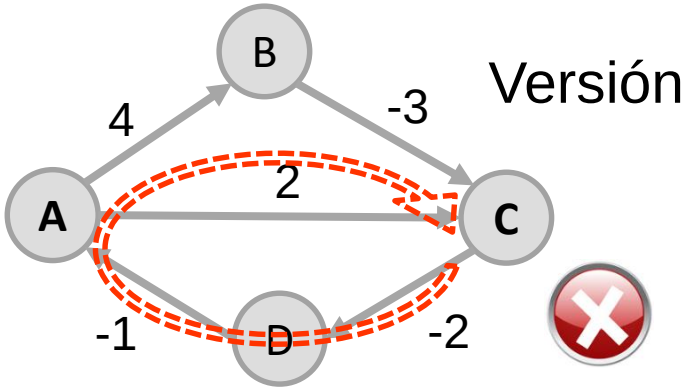
Camino más corto en un grafo con aristas negativas

¿Por qué no se puede usar el algoritmo de Dijkstra en estos casos?



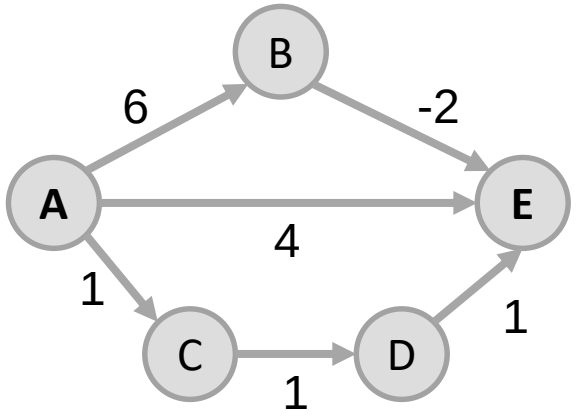
Versión sin cola

SPD = 2
{A, C}

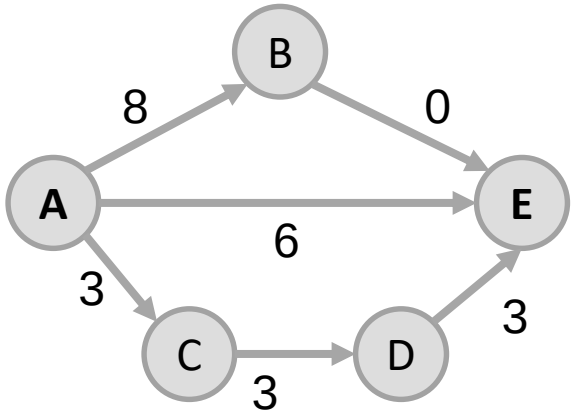


Versión con cola

¿Por qué no se puede entonces manipular las longitudes del grafo para que se pueda usar el algoritmo de Dijkstra ?



SPD = 3
{A, C, D, E}



SPD = 6
{A, E}

Algoritmo de Bellman-Ford

Nombrado así por Richard Bellman y Lester Ford Jr. quienes lo publicaron en 1958 y 1956 respectivamente.

Si un grafo contiene un ciclo negativo (la suma de las distancias de las aristas que lo componen es menor a cero), no existiría camino más corto puesto que cualquier camino a un nodo dentro de ese ciclo podría hacerse aún más corto haciendo un recorrido adicional dentro del ciclo.

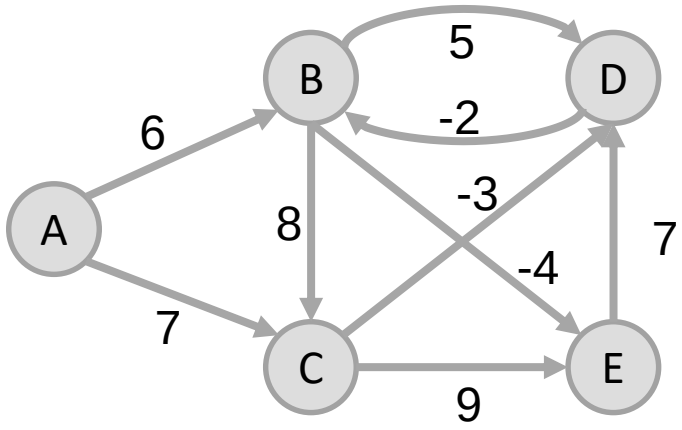
Esa precisamente es la utilidad del algoritmo de Bellman-Ford, indicar si existe en un grafo al menos un ciclo negativo.

Algoritmo de Bellman-Ford

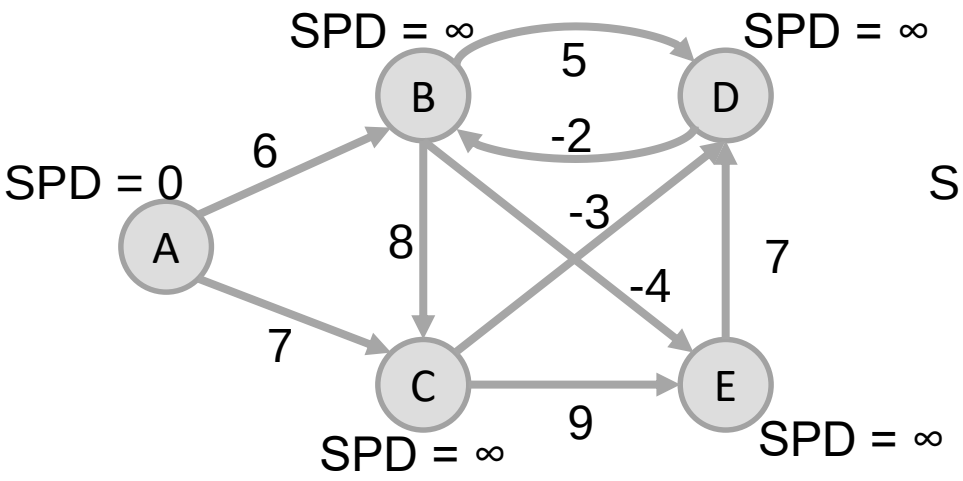
```
function BellmanFord(grafo G, nodo a):
   for  u ∈ V, u ≠ a: u.SPD = INF
   a.SPD = 0
   for  i=1:n-1:
      for (u, v) ∈ E:
         if u.SPD + (u, v).le < v.SPD:
            v.SPD = u.SPD + (u, v).le
   for all (u, v) ∈ E:
      if u.SPD + (u, v).le < v.SPD: return false
   return true
```

¿Cuál es la complejidad resultante? *O(N*M)*

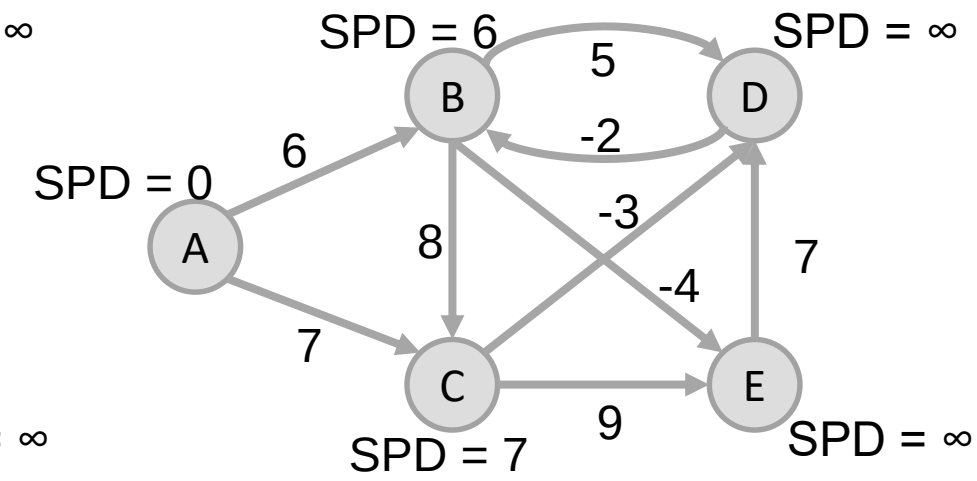
Algoritmo de Bellman-Ford



Iteración 0



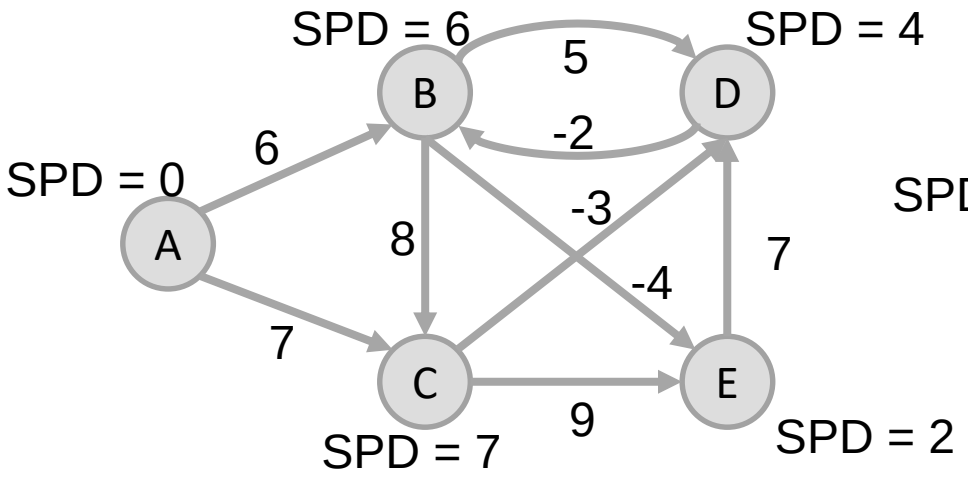
Iteración 1



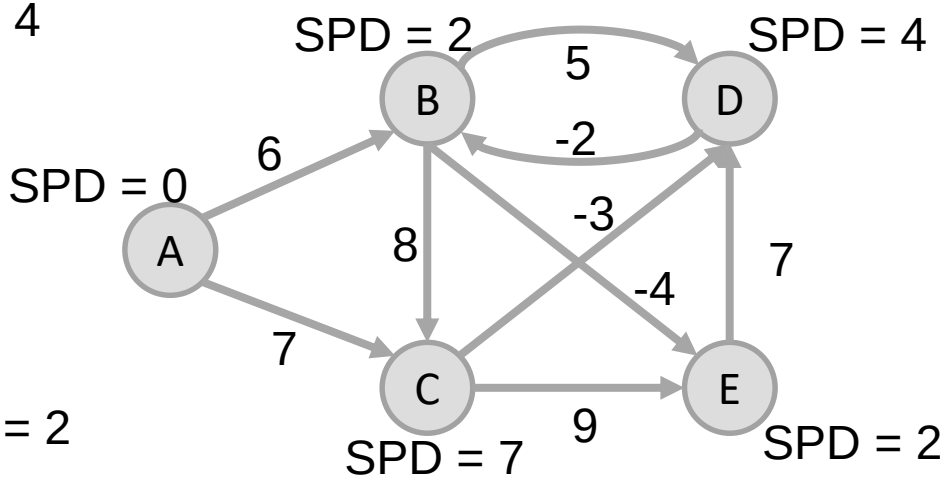
*Suponiendo que siempre se recorren las aristas en este orden: (B,D), (B,C), (B,E), (D,B), (C,D), (C,E), (E,D), (A,B), (A,C)

Algoritmo de Bellman-Ford

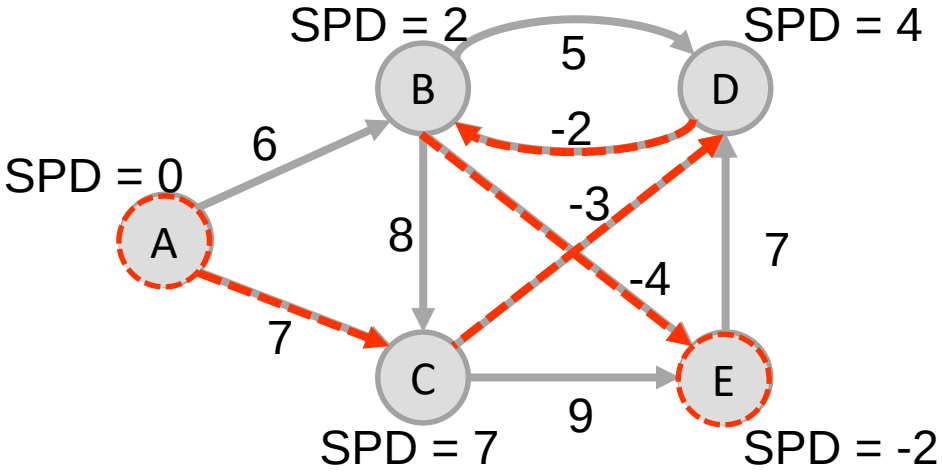
Iteración 2



Iteración 3



Iteración 4



y finalmente la función devolvería TRUE pues para todo (u, v) Є E u.SPD + (u, v).le ≥ v.SPD

*Suponiendo que siempre se recorren las aristas en este orden: (B,D), (B,C), (B,E), (D,B), (C,D), (C,E), (E,D), (A,B), (A,C)