Algoritmo de Dijkstra, parte 2

Algoritmo de Dijkstra

¿Cuál es la complejidad del algoritmo?

- El proceso *while C ≠ V* realiza exactamente *N-1* iteraciones
- Dentro de ese proceso se revisan <u>todas</u> las aristas buscando aquellas que comiencen en *C* y terminen en *V-C,* entre las que cumplan se busca un mínimo según el criterio de Dijkstra
- La eficiencia resultante por tanto es *O(N*M)*

```
function Dijkstra(grafo G, nodo a):
    C = {a} //Nodos ya revisados
    a.SPD = 0
    while C ≠ V:
        entre todas las aristas (v, w) Є V
        con v Є C y w ¬ Є C, escoger (v,’ w’)
        que minimize v.SPD + (v, w).le
        C.add(w’)
        w’.SPD = v’.SPD + (v’, w’).le
```

¿Se puede hacer mejor?

Si, pero no cambiando el algoritmo como tal si no empleando una estructura de datos apropiada

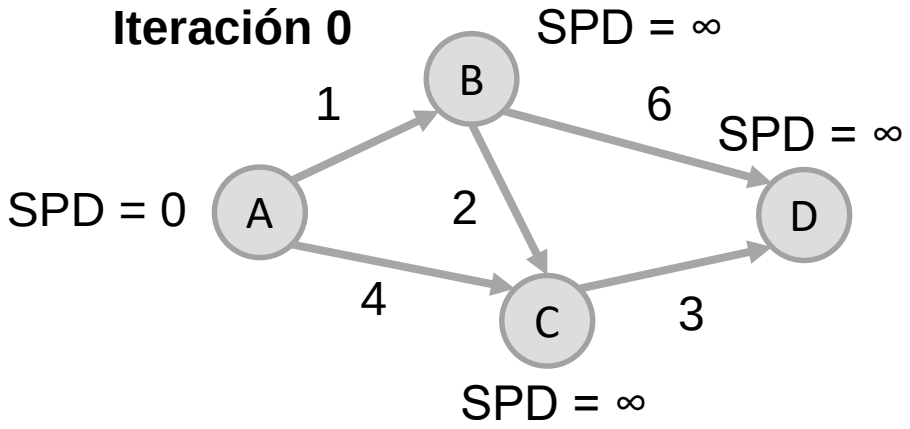
Algoritmo de Dijkstra

Se puede usar una cola con prioridad (implementada mediante un heap) para que nos ayude a escoger el siguiente nodo en cada iteración

```
function Dijkstra(grafo G, nodo a):
   for  u ∈ V, u ≠ a: u.SPD = INF
   a.SPD = 0
   Q.add(a) //Cola con prioridad
   while Q no esté vacía:
      u = Q.pop()
      for  (u, v) ∈ E:
         if u.SPD + (u, v).le < v.SPD{
            v.SPD = u.SPD + (u, v).le
            Q.add(v)
```

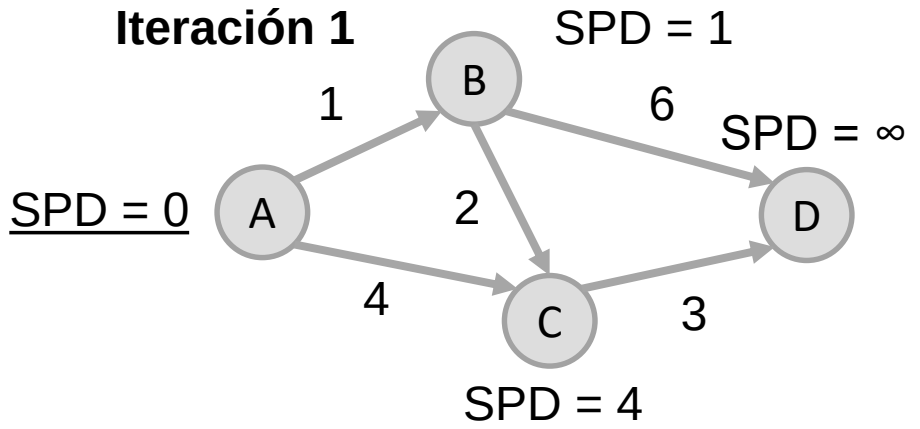
Para determinar la eficiencia resultante, miremos un ejemplo …

Iteración 0



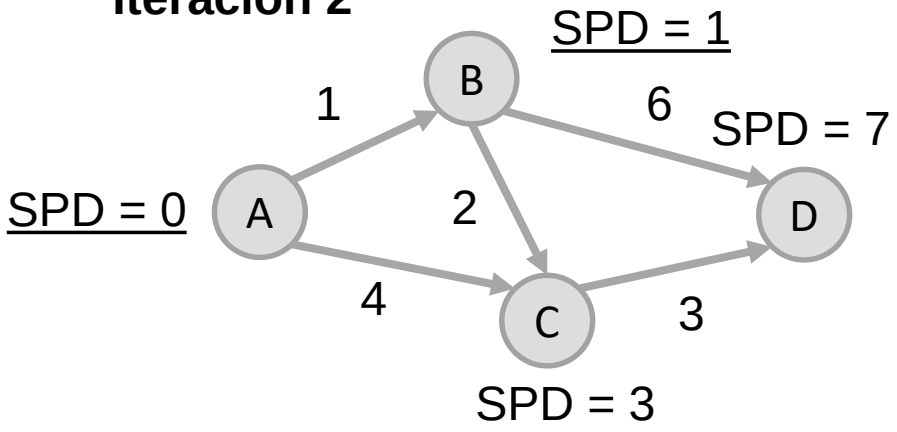
Q = {A(0)}

Iteración 1



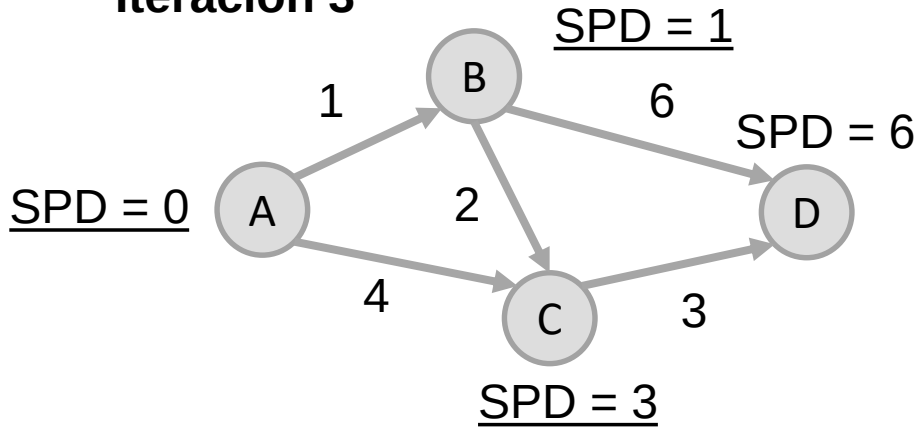
Q = {B(1), C(4)}

Iteración 2



Q = {C(3), C(4), D(7)}

Iteración 3



Q = {D(6), D(7)}

Complejidad resultante: *N*log(N)* para hacer *N* veces el pop más
*N*log(N)* para hacer push/actualizaciones → *(M+N)*log(N)*