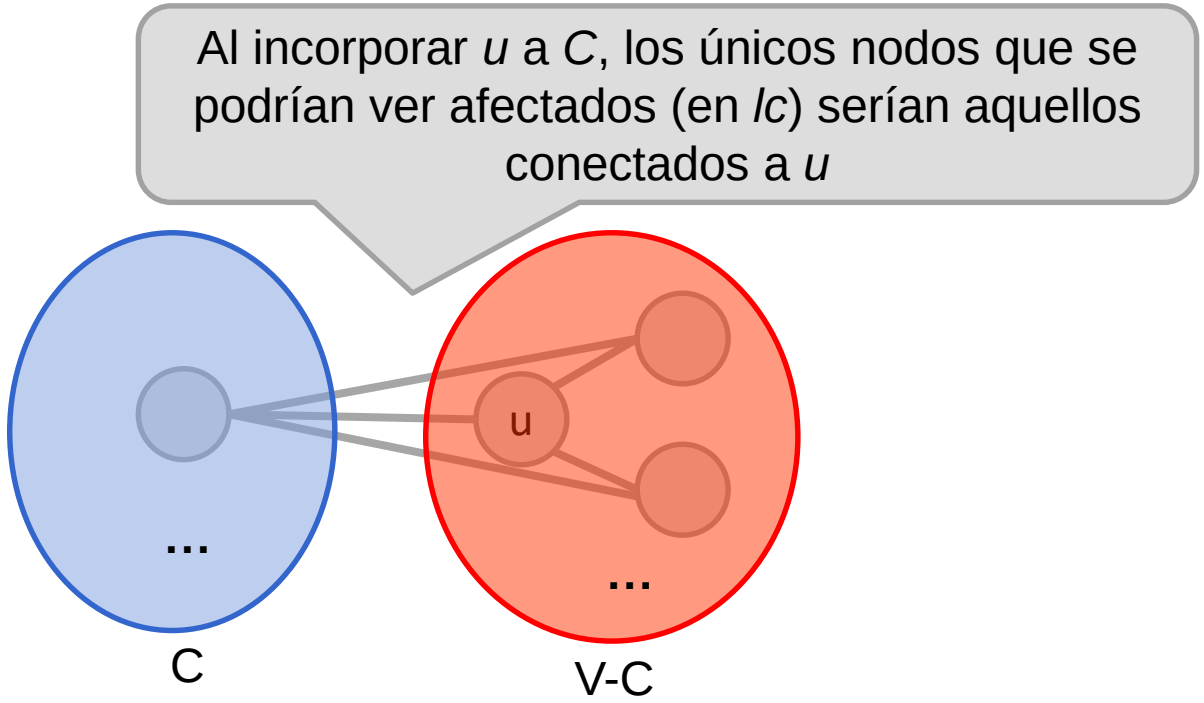
Algoritmo de Prim, parte 2

Algoritmo de Prim

```
function Prim(grafo G)
    a = cualquier nodo ∈ V
    C = {a} //Nodos ya revisados
    T = NULL
    for  v ∈ V-C:
        v.lc = menor costo de la arista (u,v) con u ∈ C ó INF si tal arista no existe
        v.be = (u,v) ó v.be = NULL en los mismos casos
    Q.add(V-C) //Cola con prioridad
    while C ≠ V{
        u = Q.pop()
        if u ∈ V-C{
            C.add(u)
            T.add(u.be)
            for  (u, v) ∈ E{
                if v ∈ V-C {
                    if (u,v).le < v.lc{
                        v.lc = (u,v).le
                        v.be = (u,v)
                        Q.add(v)
```

¿Cuál es la eficiencia resultante?

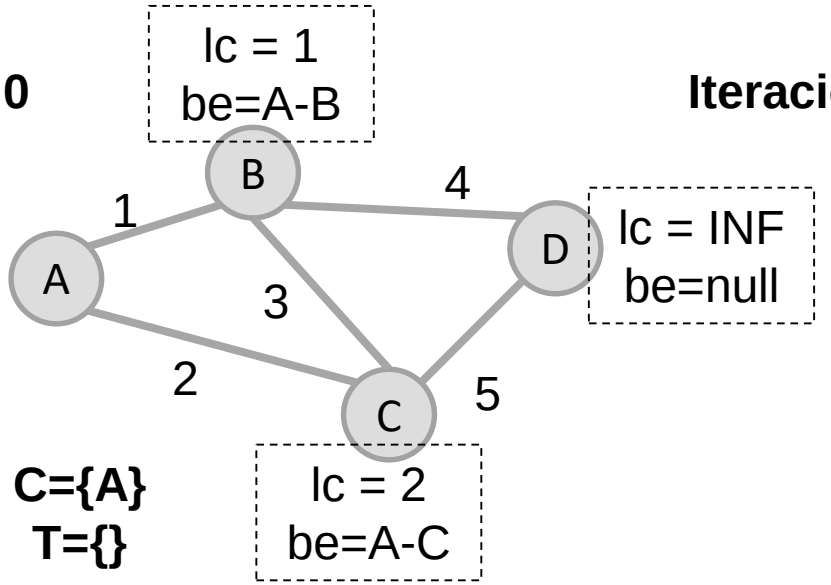
Igual que en Dijkstra $O((N+M)log(N))$



Algoritmo de Prim

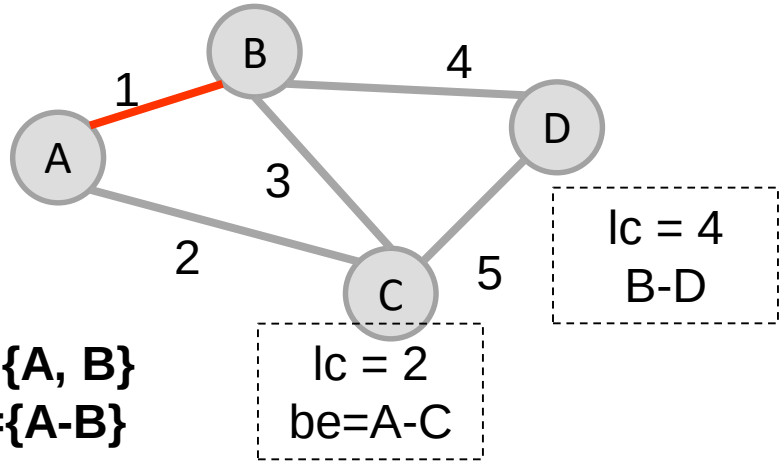
Ejemplo:

Iteración 0



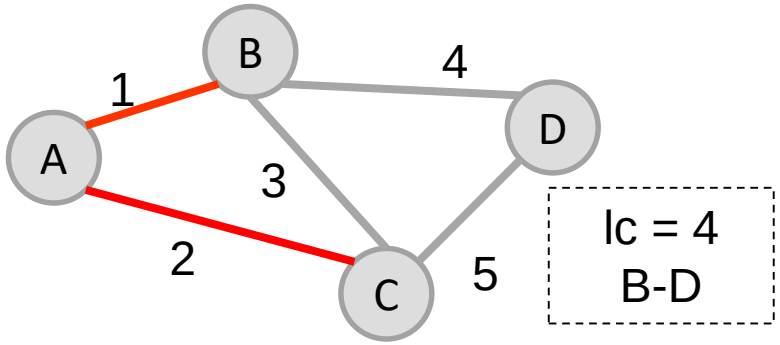
C={A}
T={}

Iteración 1



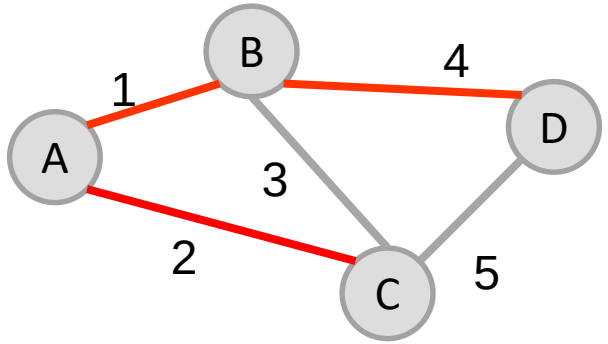
C={A, B}
T={A-B}

Iteración 2



C={A, B, C}
T={A-B, A-C}

Iteración 2



C={A, B, C, D}
T={A-B, A-C, B-D}