Ayudantía 10

MST
Prim
Kruskal

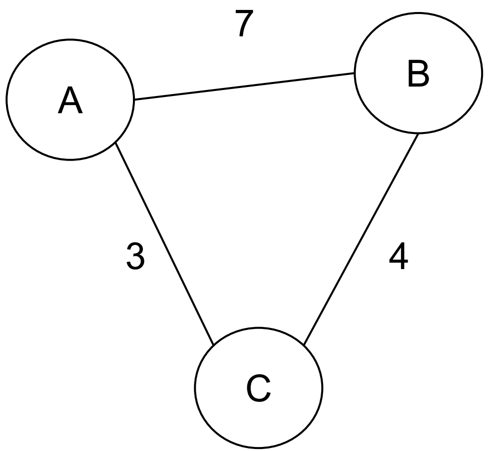
Martín Orrego - @Orregol

MST

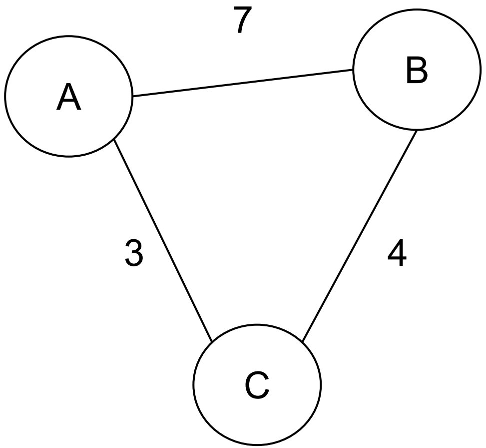
Definición

- Dado un grafo G **no dirigido**, un subgrafo T se dice **MST** de G si:
    1. T es un **árbol**
    2. V(T) = V(G)
    3. **No** existe otro MST T' para G con menor costo total

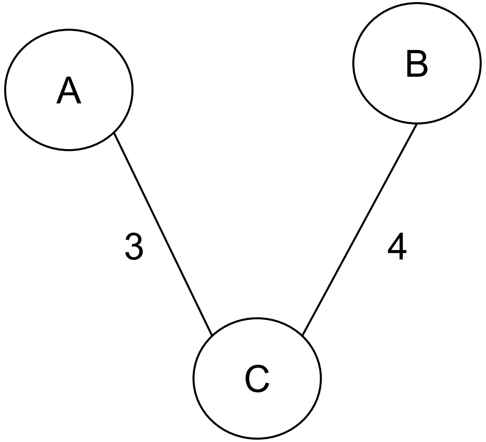
Ejemplos rápidos #1



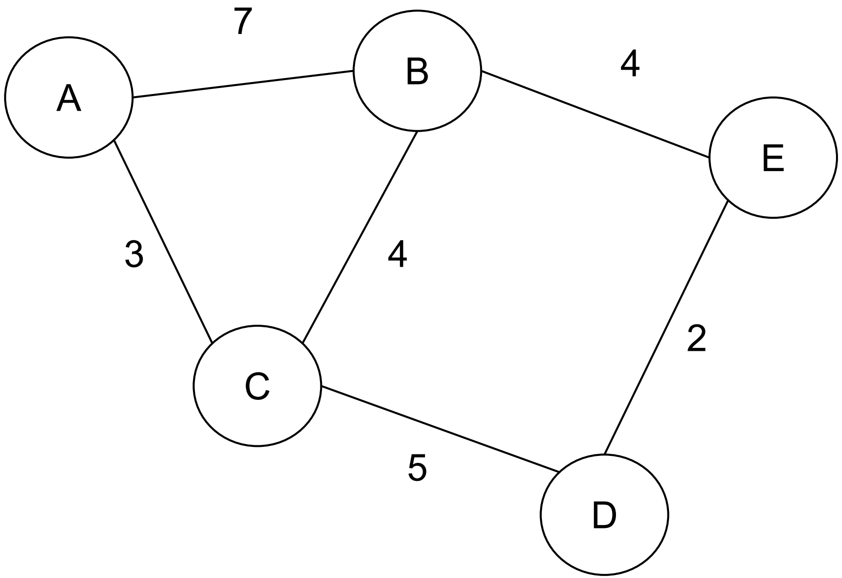
Ejemplos rápidos #1



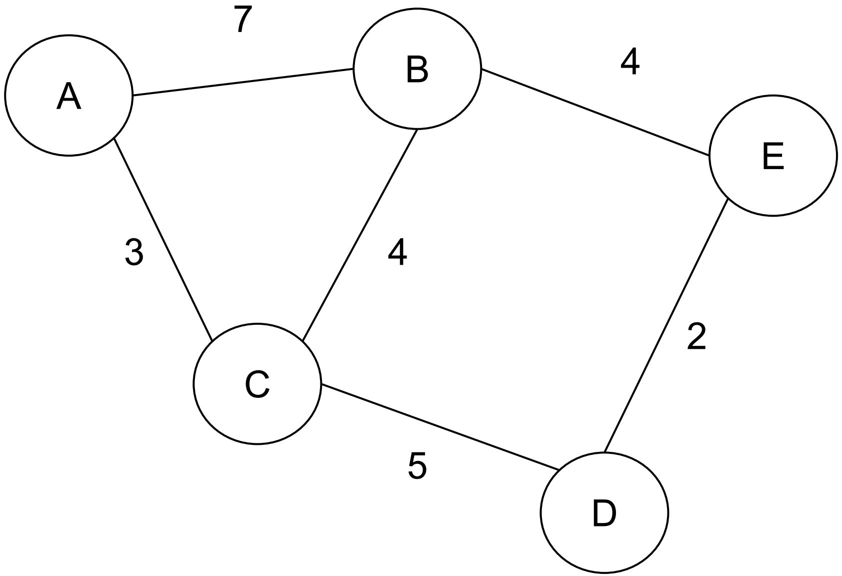
MST
Costo = 7



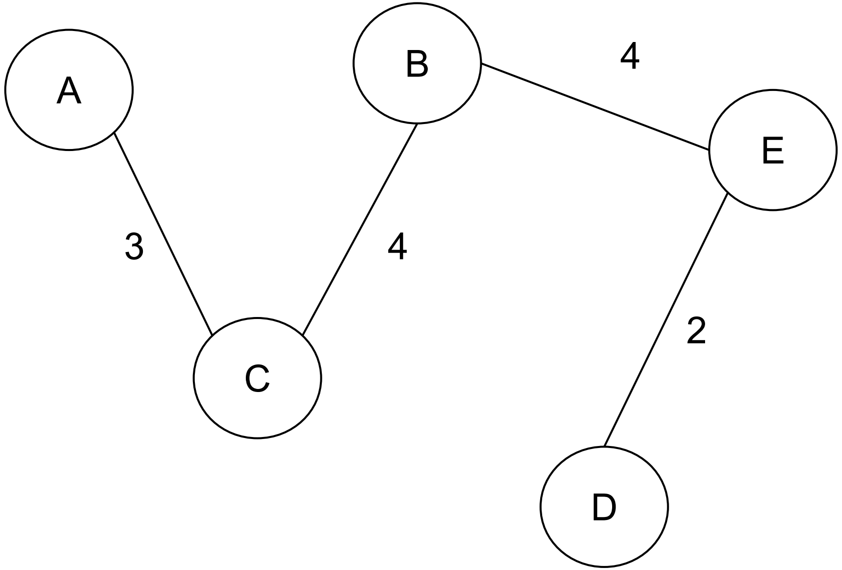
Ejemplos rápidos #2



Ejemplos rápidos #2



MST
Costo = 13



Problema

- Con cada nodo que se agrega se vuelve más complicado encontrar “visualmente” el MST
- Necesitamos un algoritmo que resuelva ese problema por nosotros

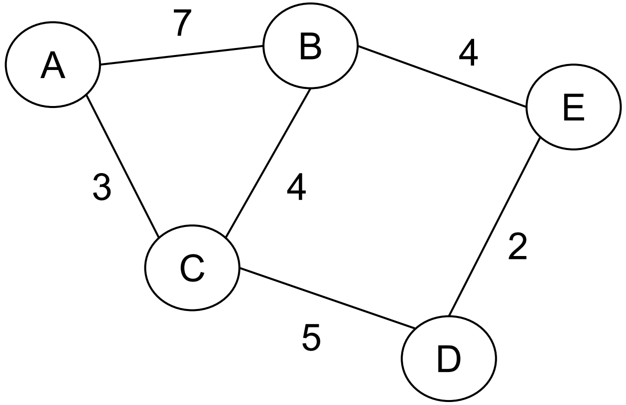
Soluciones

- Algoritmo de Prim
- Algoritmo de Kruskal

Algoritmo de Prim: conceptos previos

- **Corte**: Partición (V_1, V_2) de V(G)
  - V_1 y V_2 no son vacíos
  - La **unión** de V_1 y V_2 = V(G)
  - V_1 y V_2 no comparten vértices

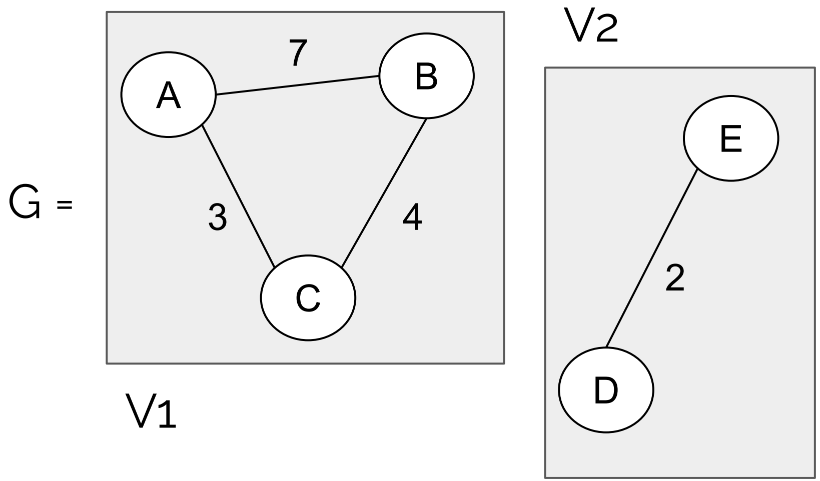
¿Cómo se ve un corte?

G = 

Algoritmo de Prim: conceptos previos

- **Corte**: Partición (V_1, V_2) de V(G)
  - V_1 y V_2 no son vacíos
  - La **unión** de V_1 y V_2 = V(G)
  - V_1 y V_2 no comparten vértices

¿Cómo se ve un corte?

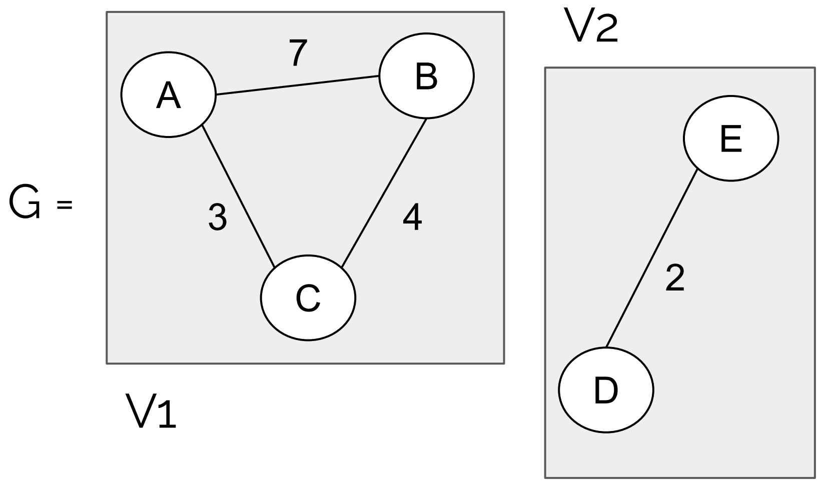


Algoritmo de Prim: conceptos previos

- Arista que **cruza un corte** (V1, V2):

  Arista que comienza en un vértice de V1 y termina en un vértice de V2

¿Cómo se ve?

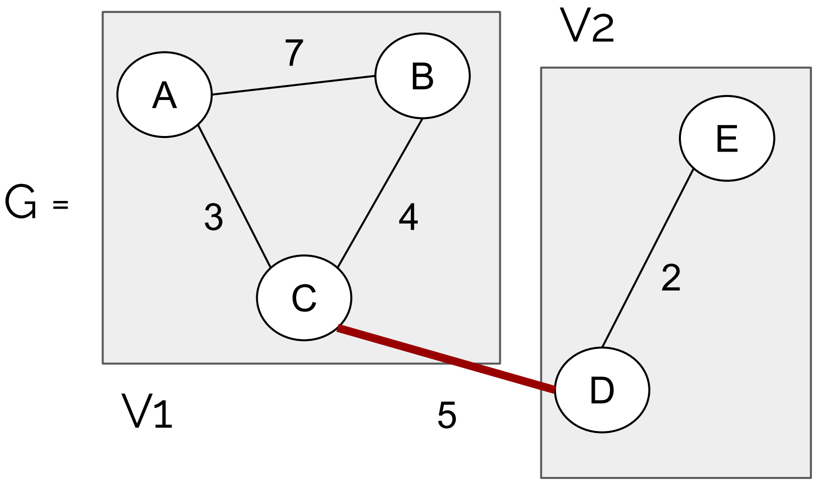


Algoritmo de Prim: conceptos previos

- Arista que **cruza un corte** (V_1, V_2):

  Arista que comienza en un vértice de V_1 y termina en un vértice de V_2

¿Cómo se ve?



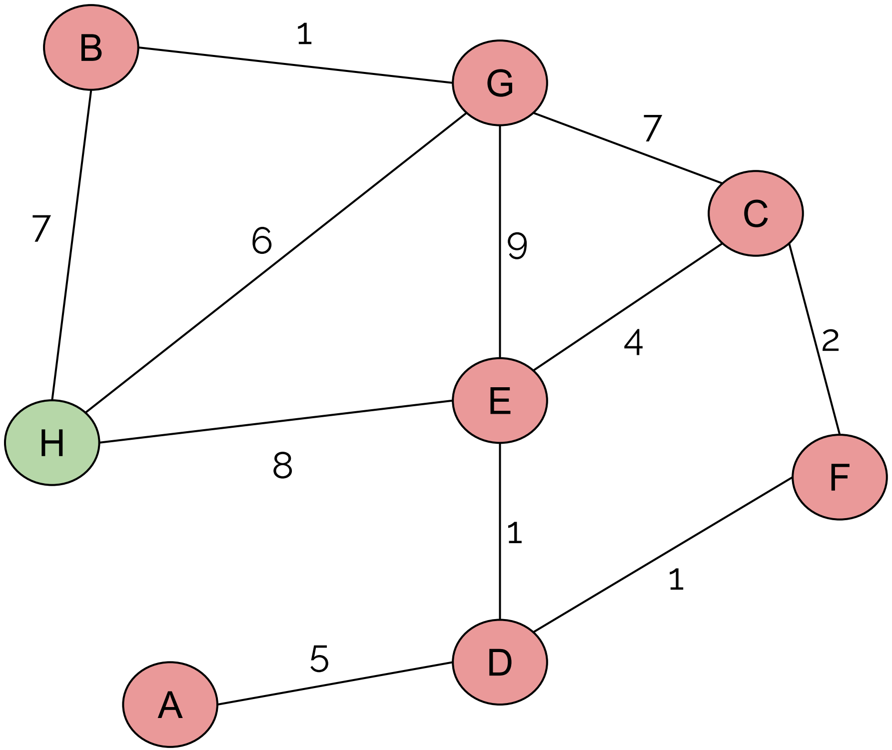
Algoritmo de Prim:

- Idea base: Utilizar aristas que cruzan cortes para guiar su construcción

Sea G = (V, E) grafo no dirigido y v nodo inicial cualquiera.

1. R = {v} y R’ = V - R # Corte de v y todos los nodos restantes
2. Sea **e** la arista de menor costo que cruza R a R’ # Usamos arista que cruza corte
3. Sea **u** el nodo de **e** que pertenece a R’
4. Agregar **e** al MST. Eliminar **u** de R’ y agregar a R # Ahora tenemos un nuevo corte
5. Si quedan elementos en R’, volver al paso 2

Ejemplo:

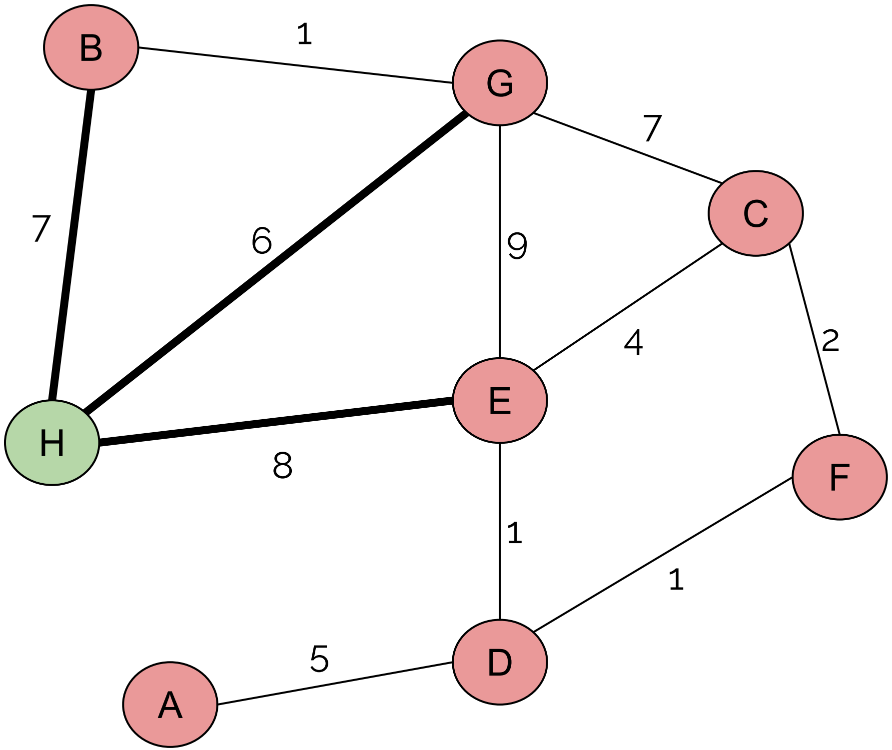


V = H

R = { H }

R' = { A, B, C, D, E, F, G }

Ejemplo:



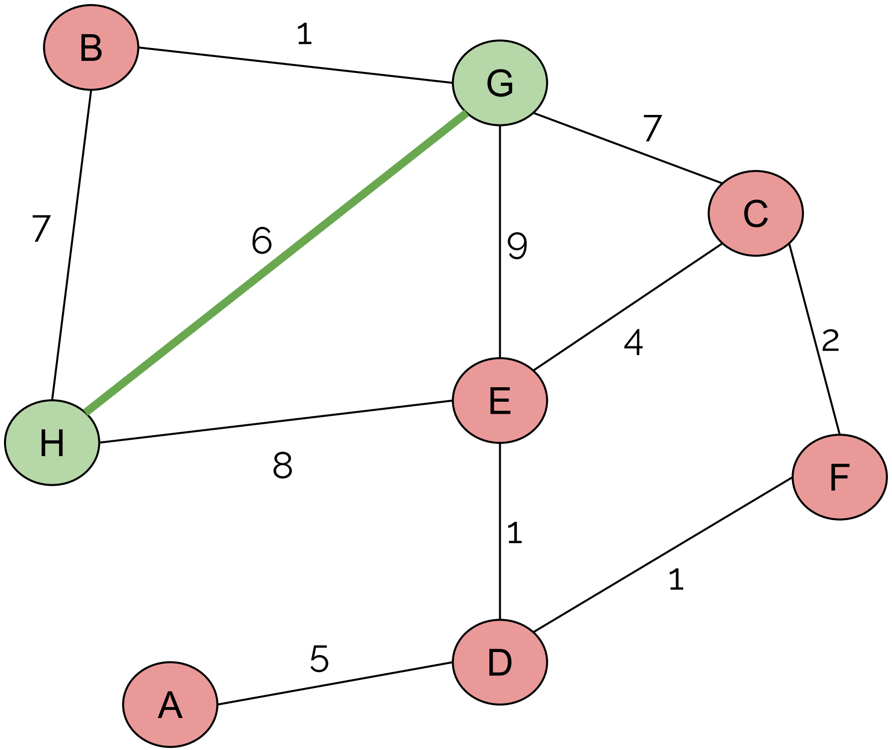
V = H

R = { H }

R' = { A, B, C, D, E, F, G }

Aristas candidatas

Ejemplo:



V = H

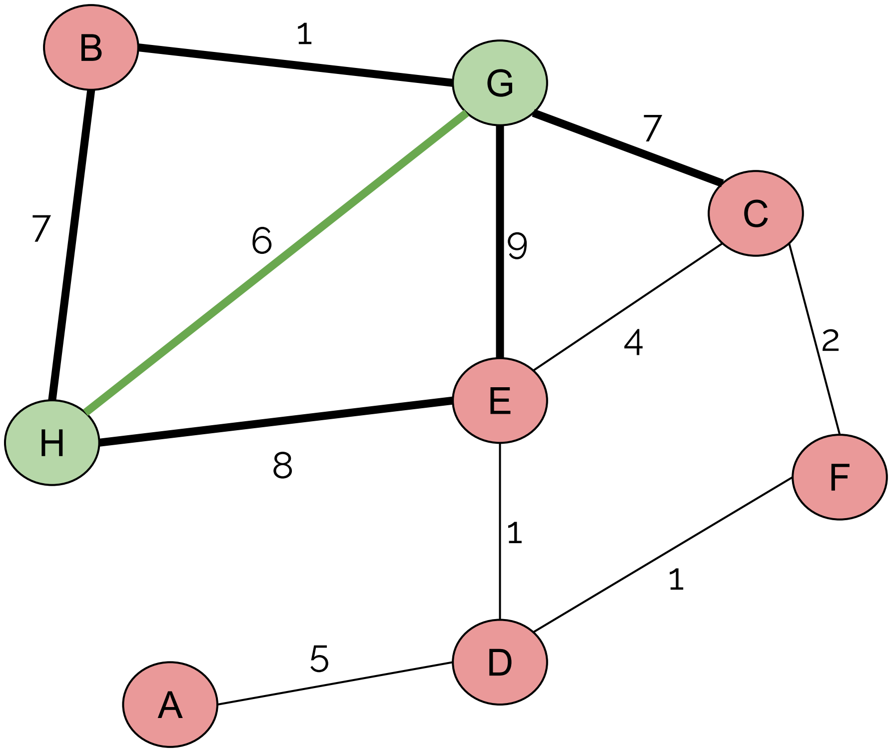
R = { H, G }

R' = { A, B, C, D, E, F}

Aristas candidatas

Aristas en MST

Ejemplo:



V = H

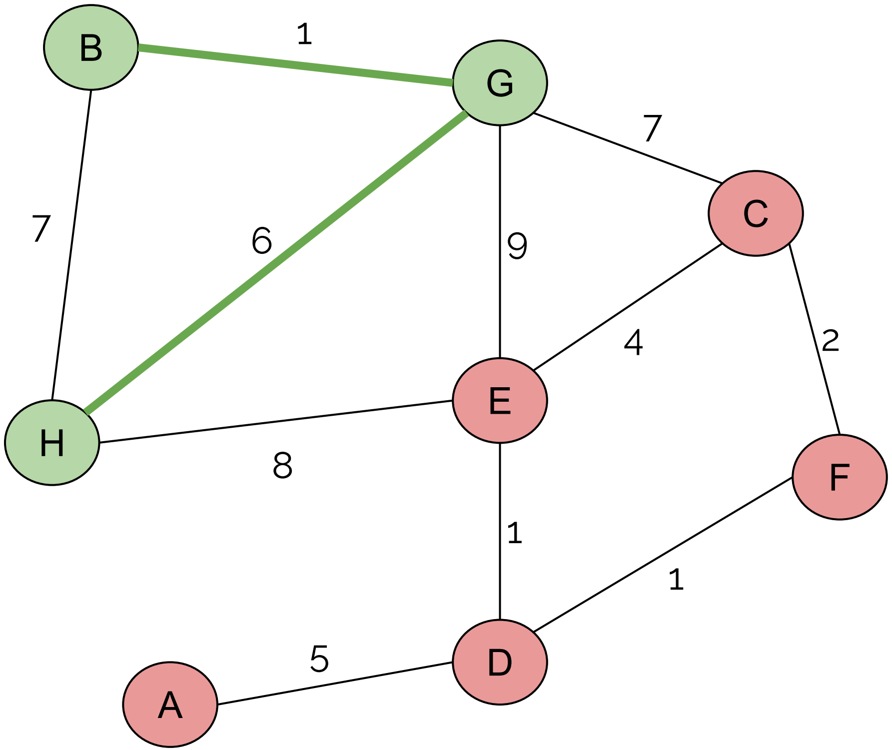
R = { H, G }

R' = { A, B, C, D, E, F}

Aristas candidatas

Aristas en MST

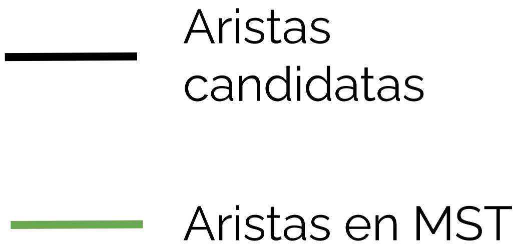
Ejemplo:



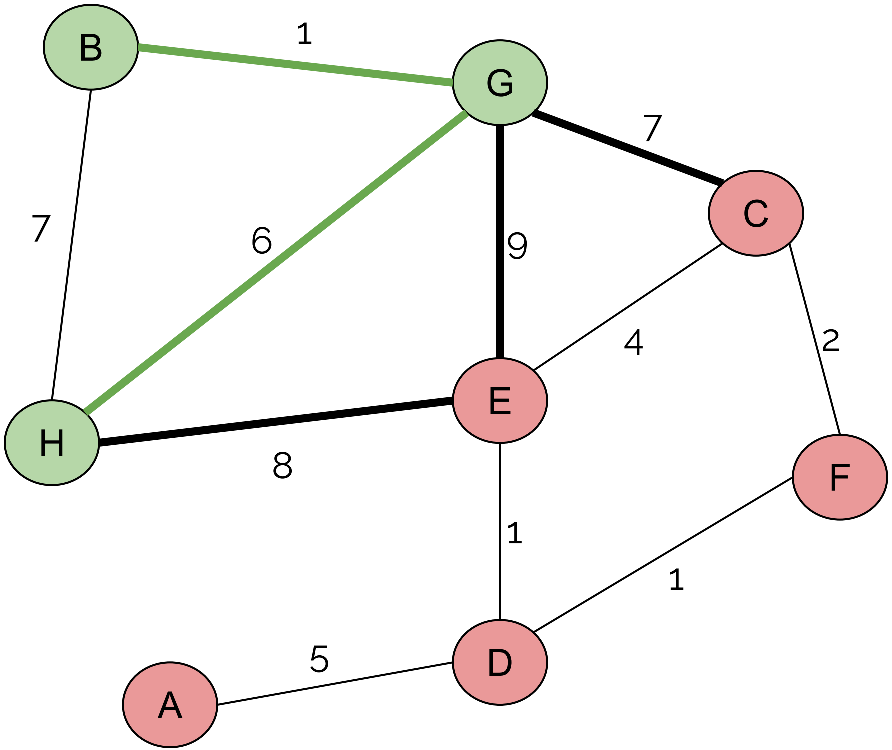
V = H

R = { H, G, B }

R' = { A, C, D, E, F}



Ejemplo:



V = H

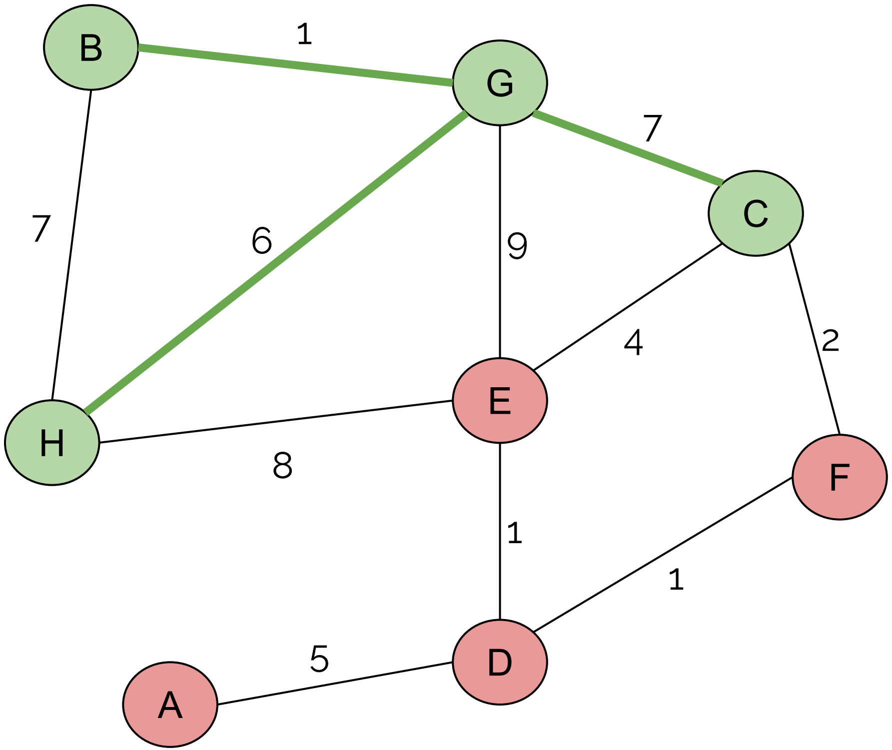
R = { H, G, B }

R' = { A, C, D, E, F}

Aristas candidatas

Aristas en MST

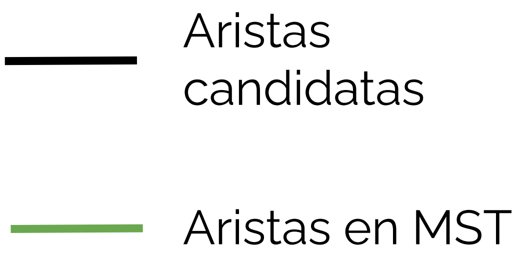
Ejemplo:



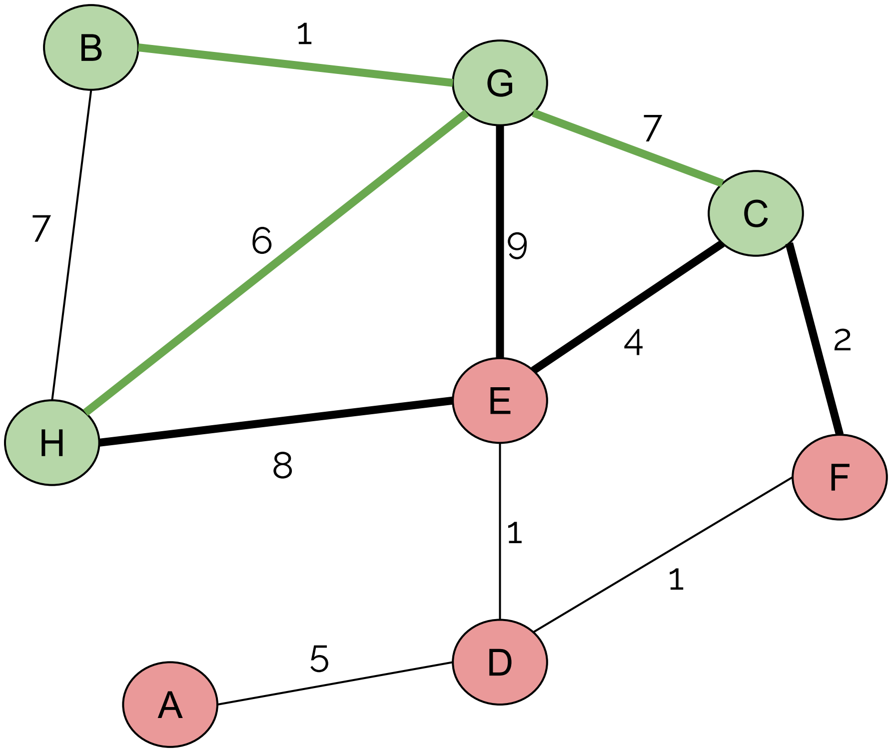
V = H

R = { H, G, B, C}

R' = { A, D, E, F}



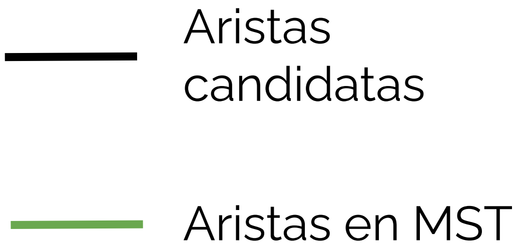
Ejemplo:



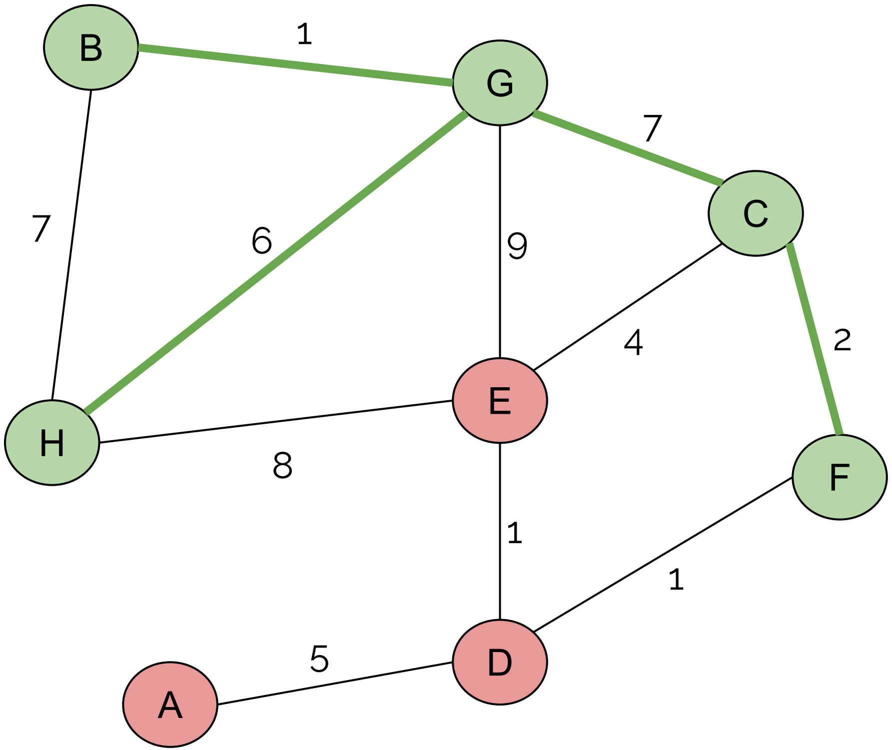
V = H

R = { H, G, B, C}

R' = { A, D, E, F}



Ejemplo:



V = H

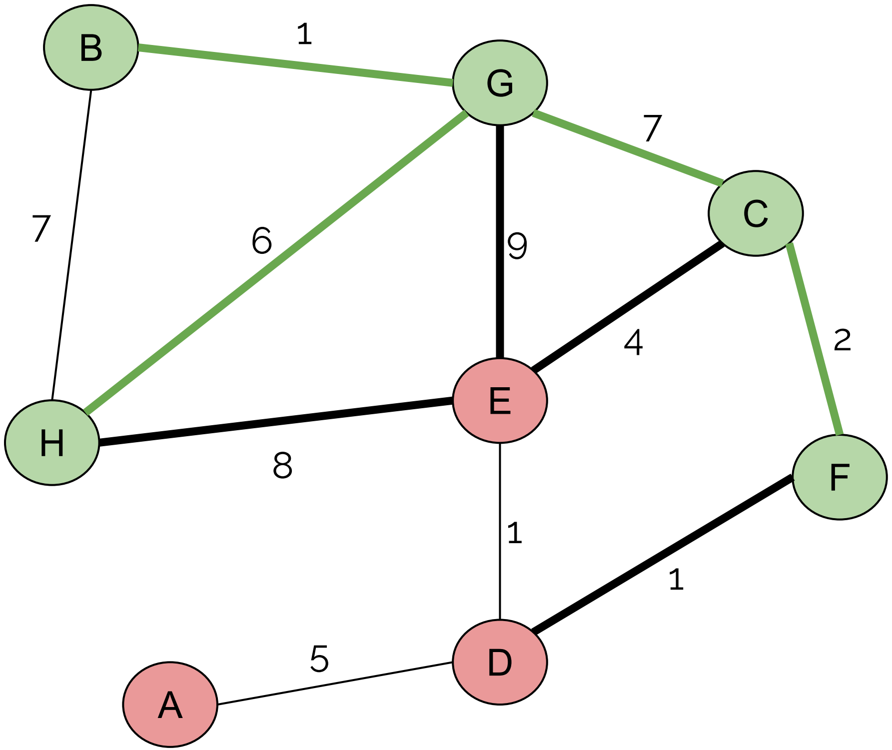
R = { H, G, B, C, F }

R' = { A, D, E }

Aristas candidatas

Aristas en MST

Ejemplo:



V = H

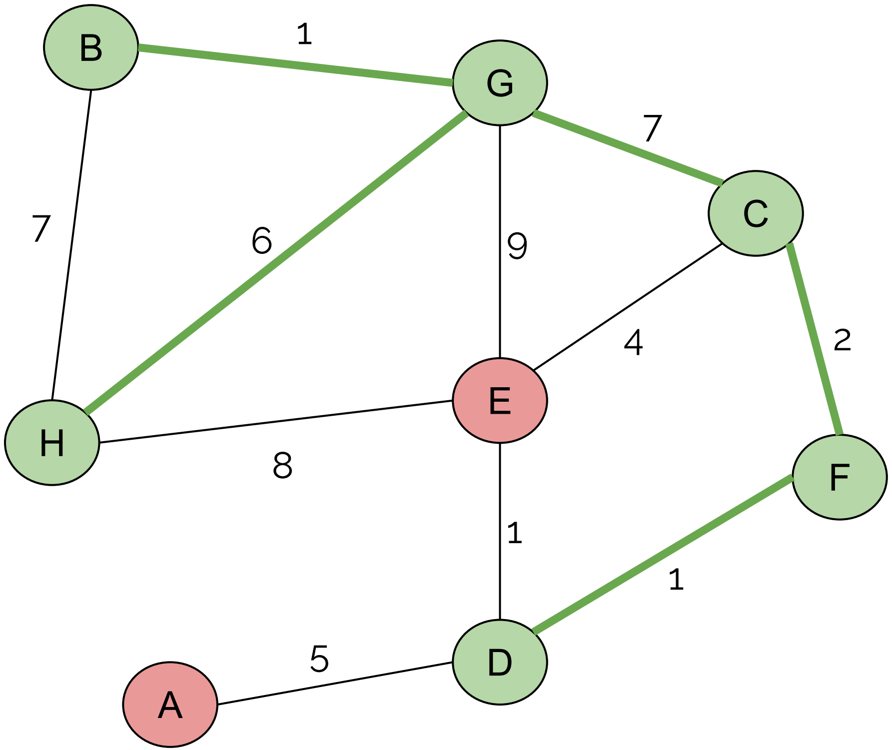
R = { H, G, B, C, F }

R' = { A, D, E }

Aristas candidatas

Aristas en MST

Ejemplo:



V = H

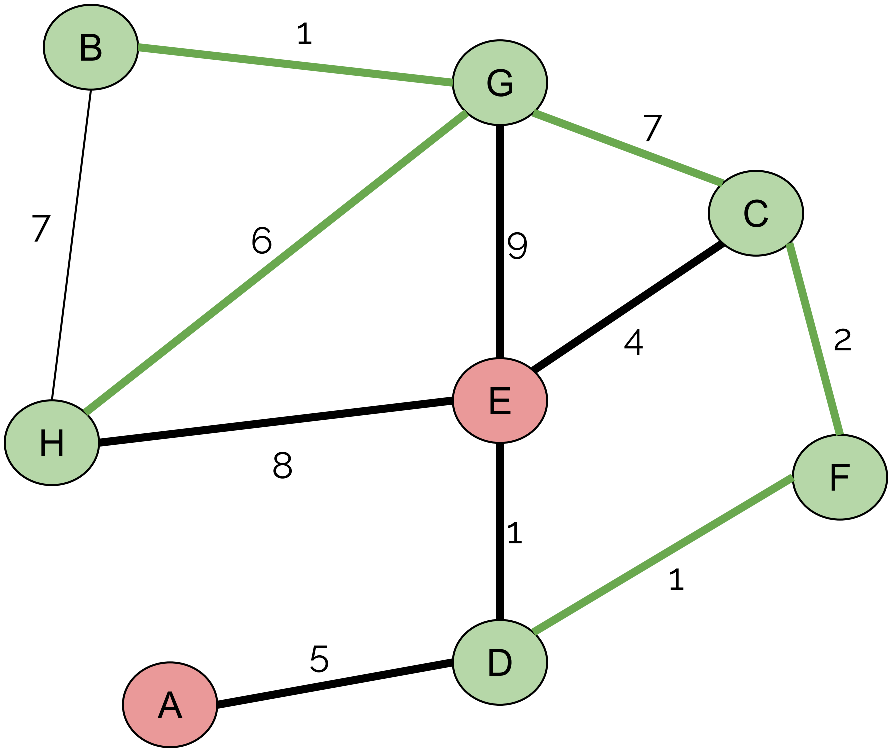
R = { H, G, B, C, F, D }

R' = { A, E }

Aristas candidatas

Aristas en MST

Ejemplo:



V = H

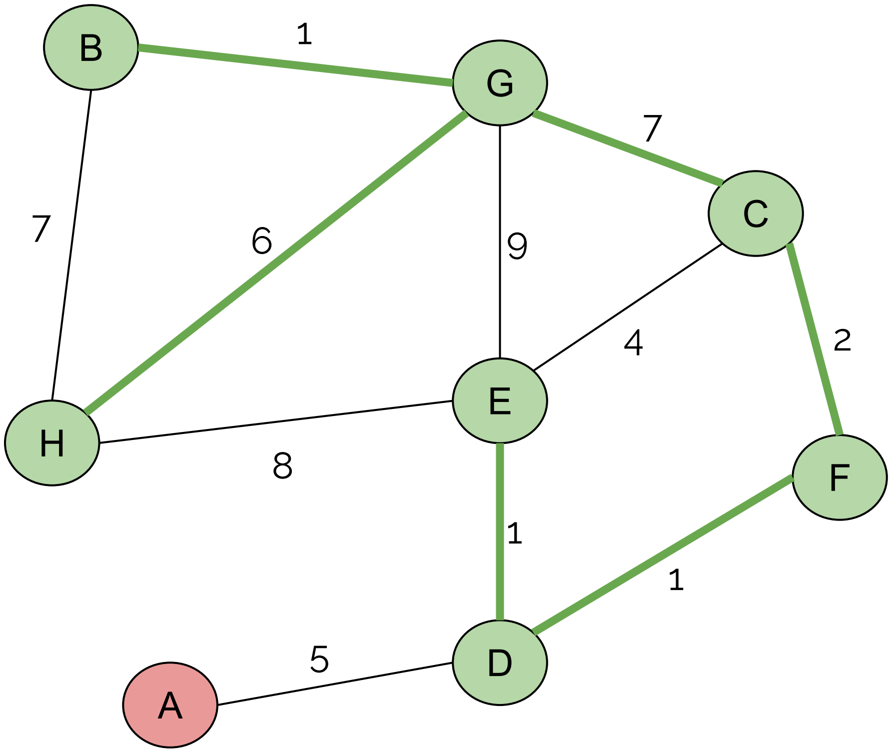
R = { H, G, B, C, F, D }

R' = { A, E }

Aristas candidatas

Aristas en MST

Ejemplo:



V = H

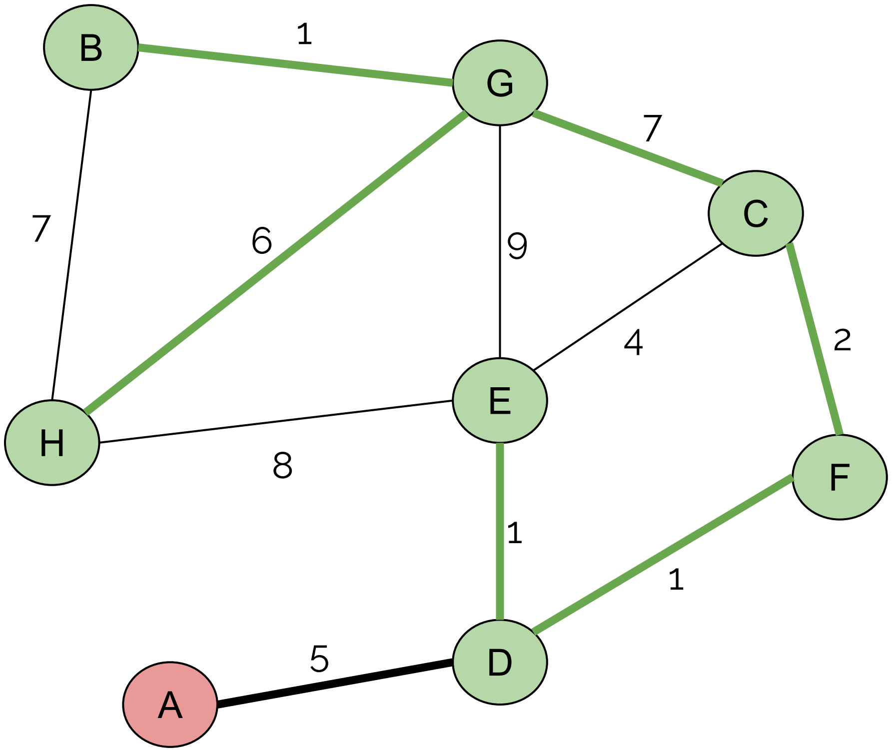
R = { H, G, B, C, F, D, E }

R' = { A }

Aristas candidatas

Aristas en MST

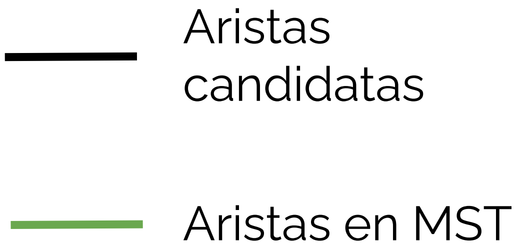
Ejemplo:



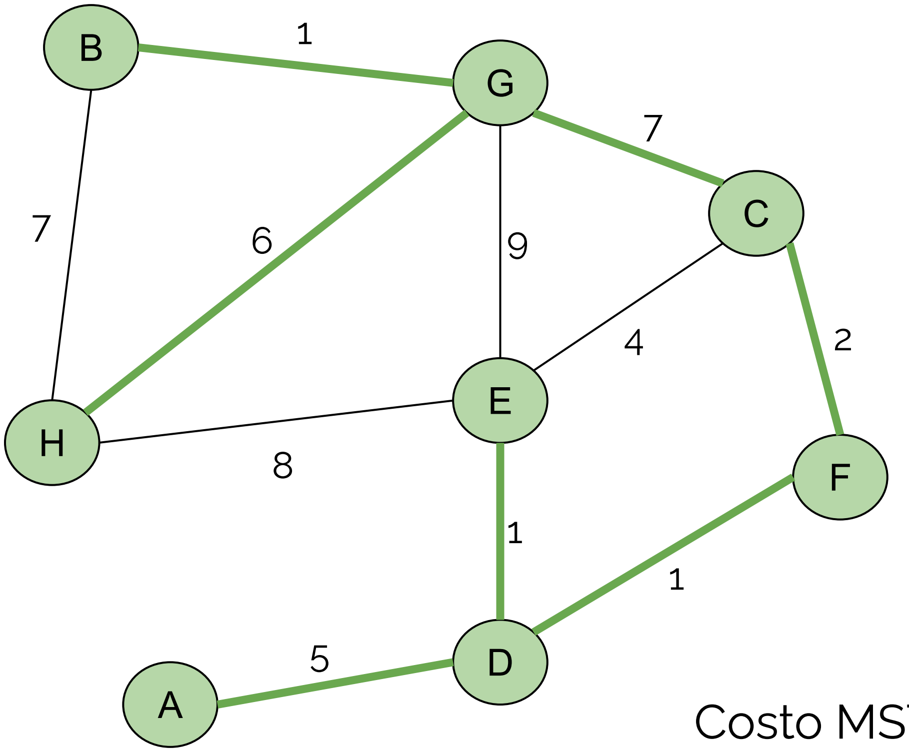
V = H

R = { H, G, B, C, F, D, E }

R' = { A }



Ejemplo:



V = H

R = { H, G, B, C, F, D, E, A }

R' = { }

Aristas candidatas

Aristas en MST

Costo MST = 6 + 1 + 7 + 2 + 1 + 1 + 5 = 23

Prim

```
Prim(Grafo G)
     set Q //de nodos ordenados por C[u]
     //C[u] es infty en un comienzo
     forest F
     while not Q.empty()
          u = min(Q)
          F.add(u)
          if E[u]!=null
               F.add(E[u])
          for w,v in G.adjacent[u]
               C[v] = w
               E[v] = (u,v)
```

Kruskal

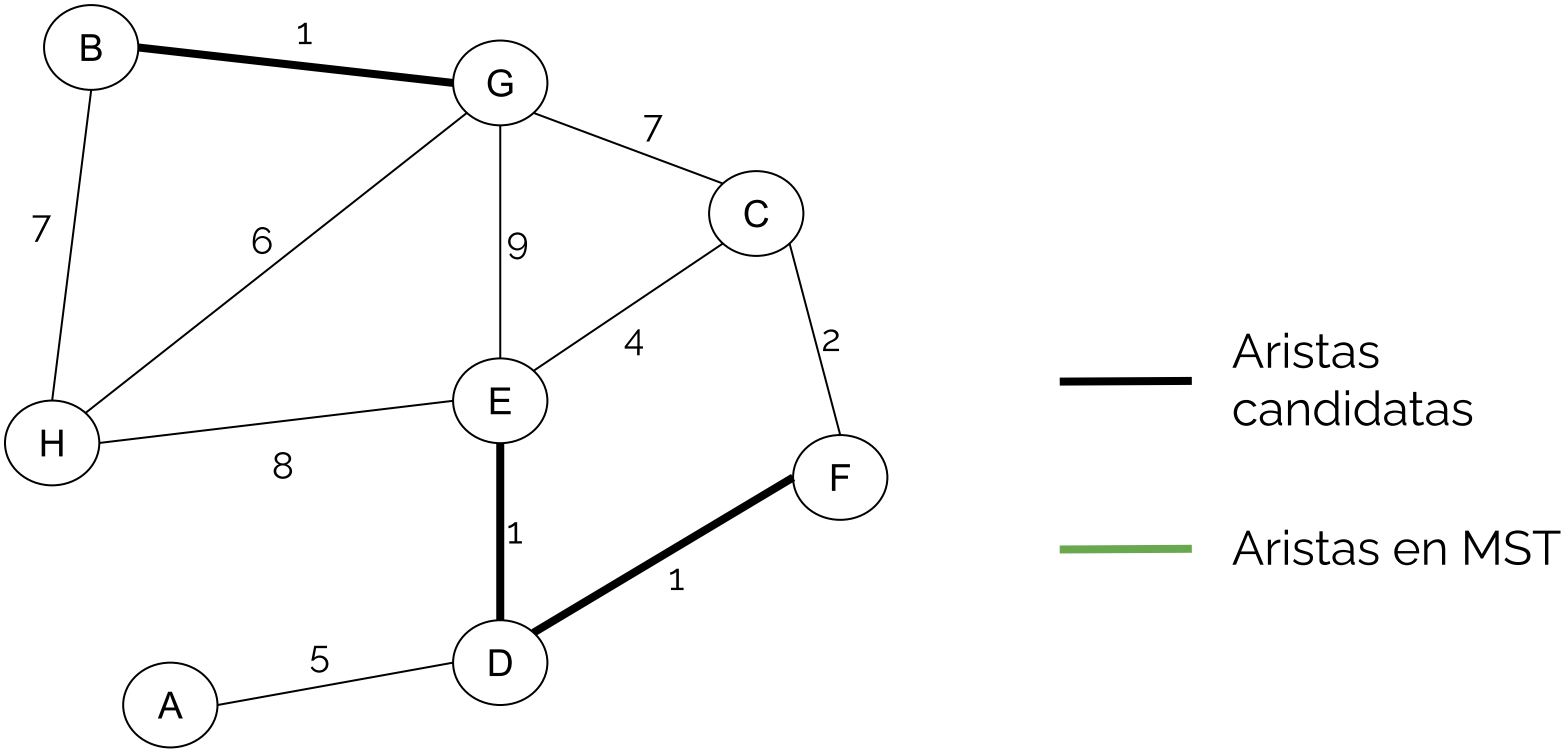
Idea base: Crear un bosque que converge en un único árbol

Sea G = (V, E) grafo no dirigido iteramos sobre las aristas **e** en orden no decreciente de costo

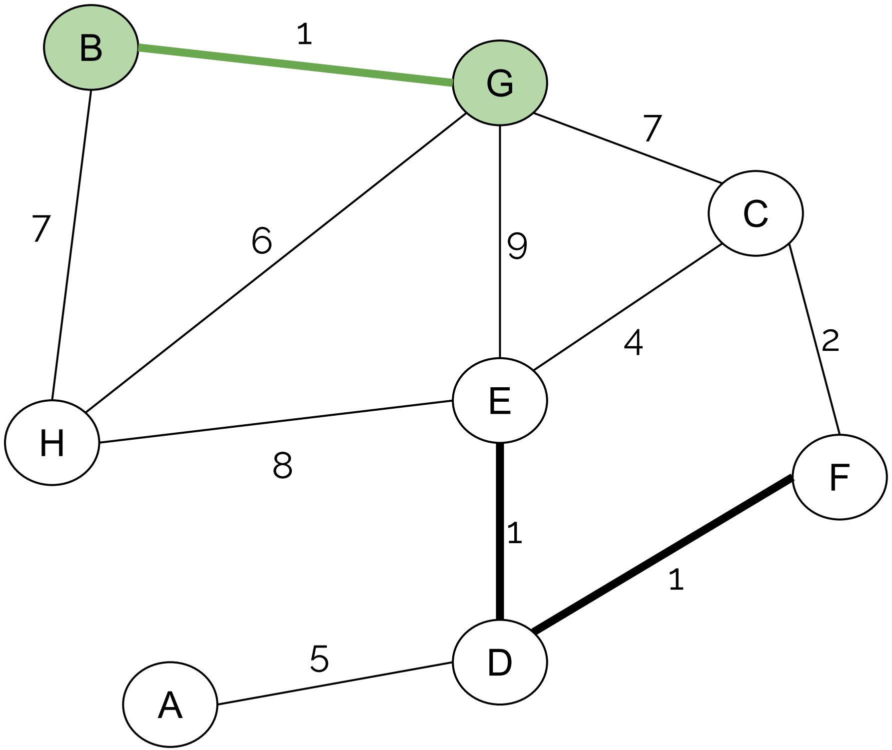
1. Si **e** genera un ciclo al agregarla a **T,** la ignoramos
2. Si no genera ciclo, se agrega

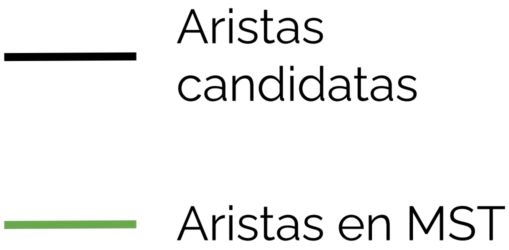
¿Será necesario revisar **todas** las aristas de E?

Ejemplo:

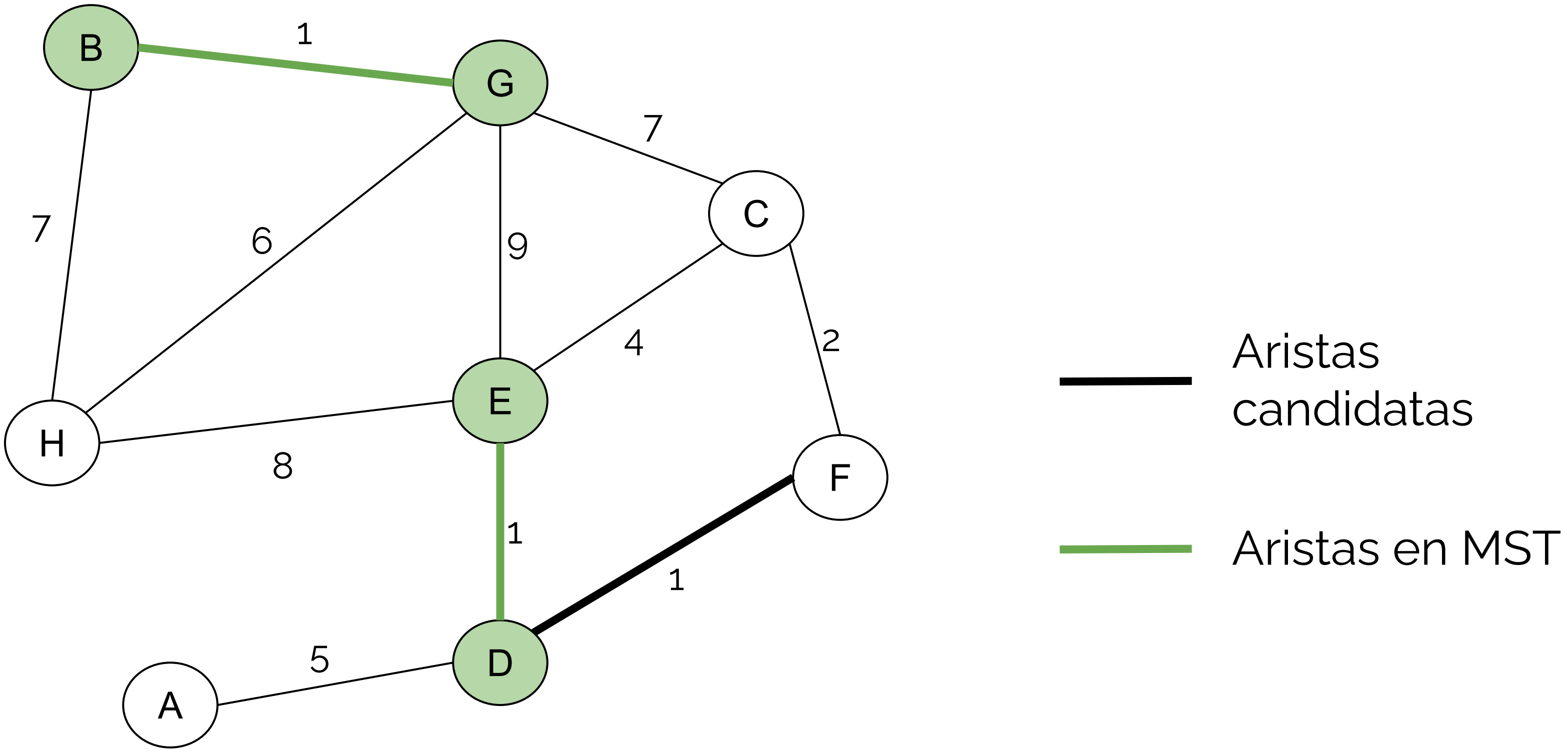


Ejemplo:

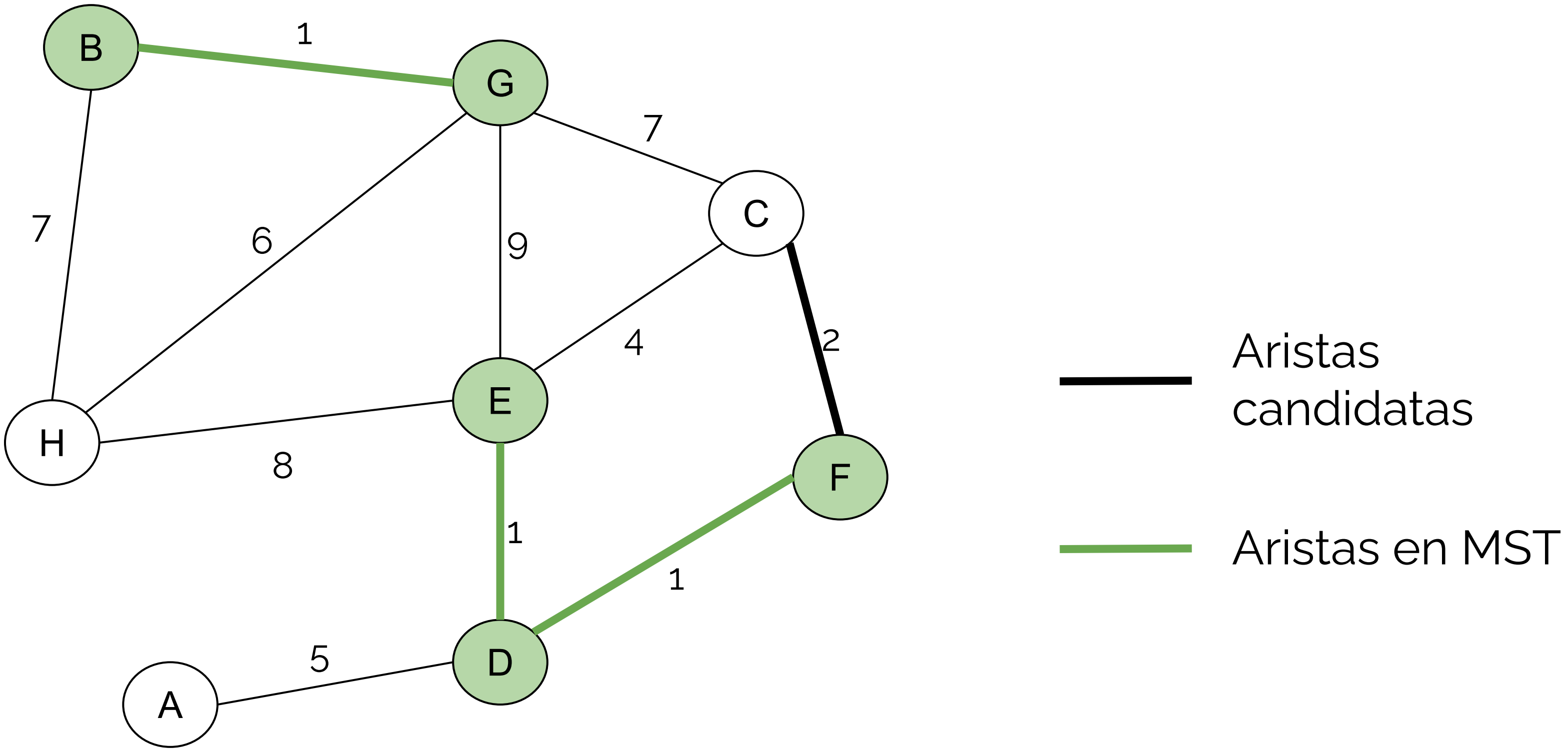




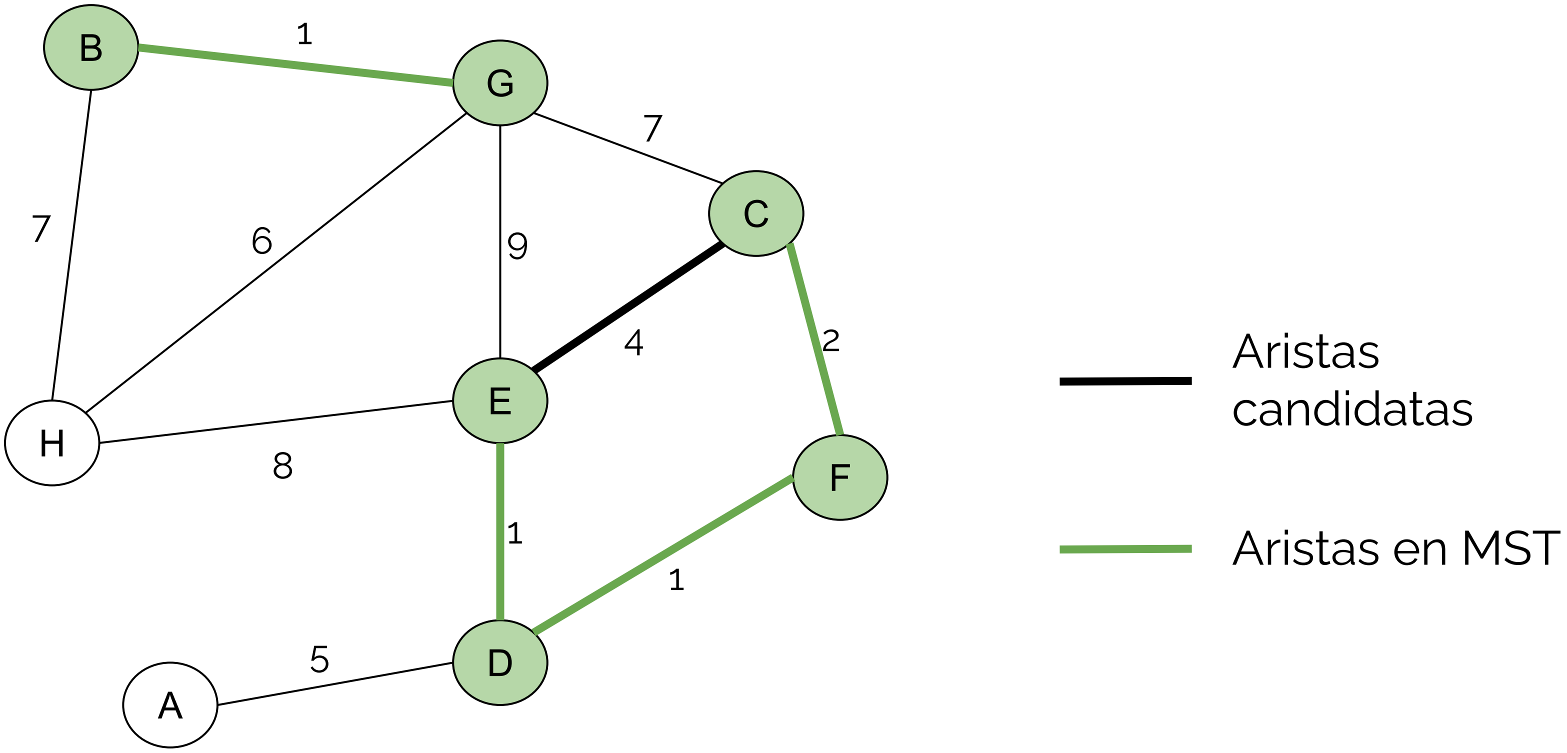
Ejemplo:



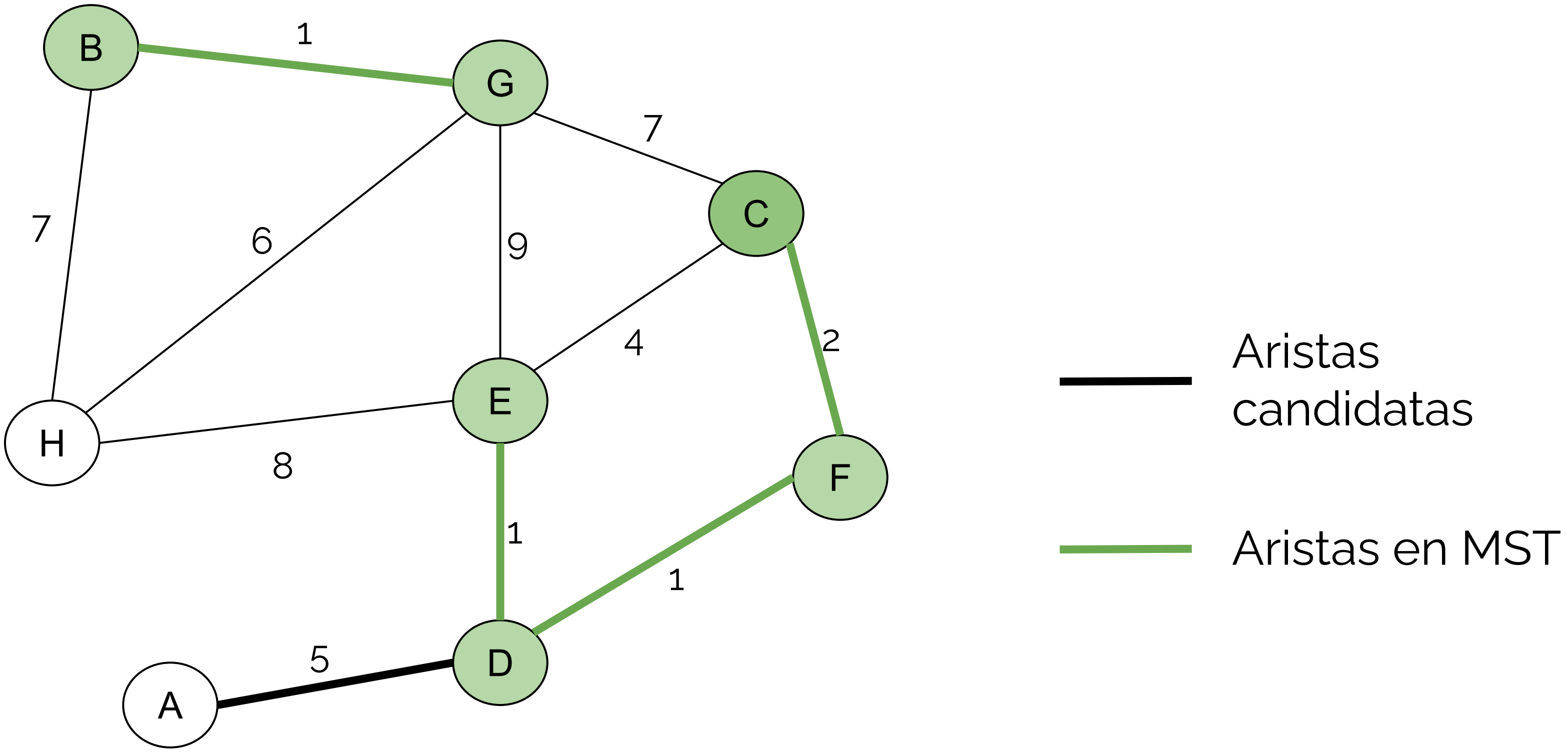
Ejemplo:



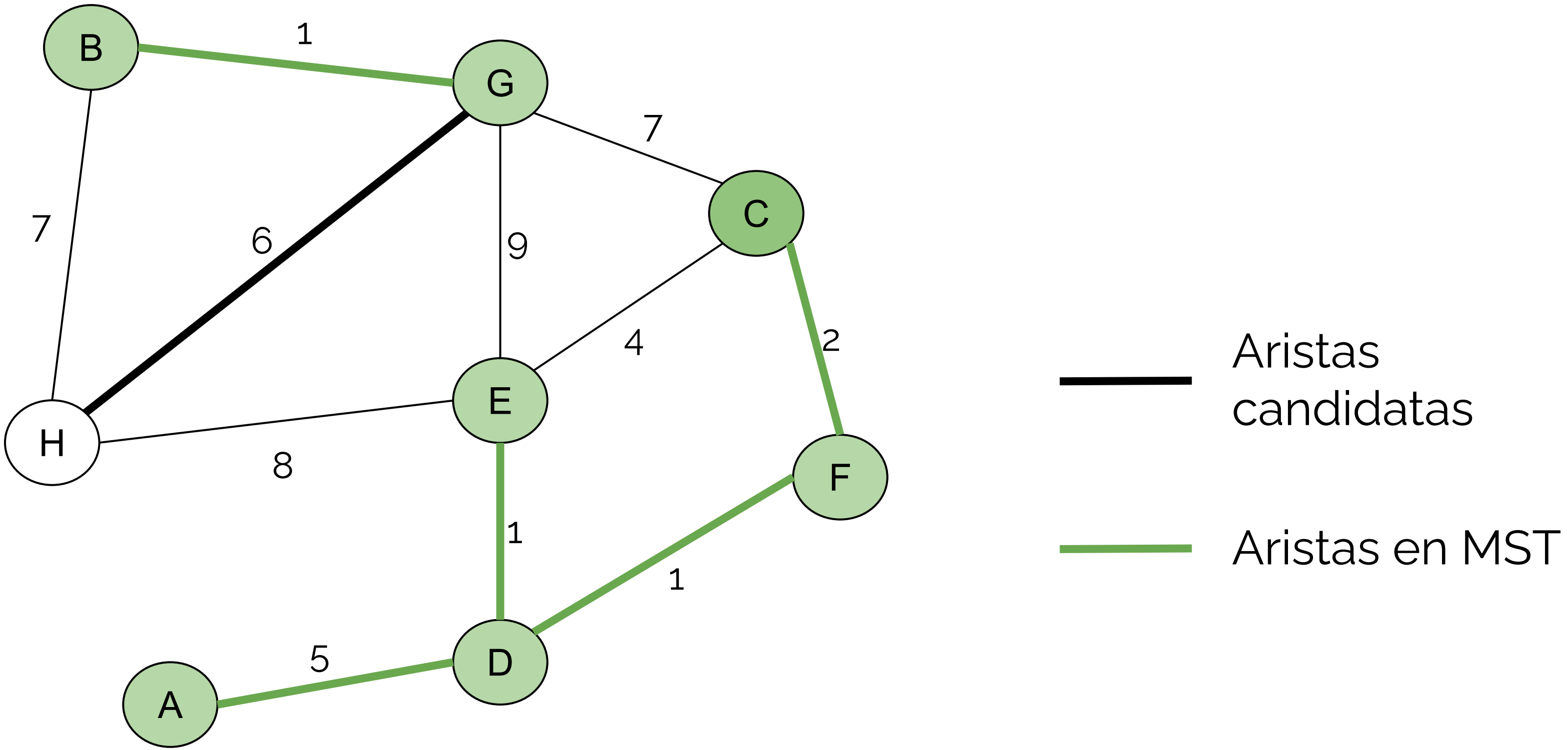
Ejemplo:



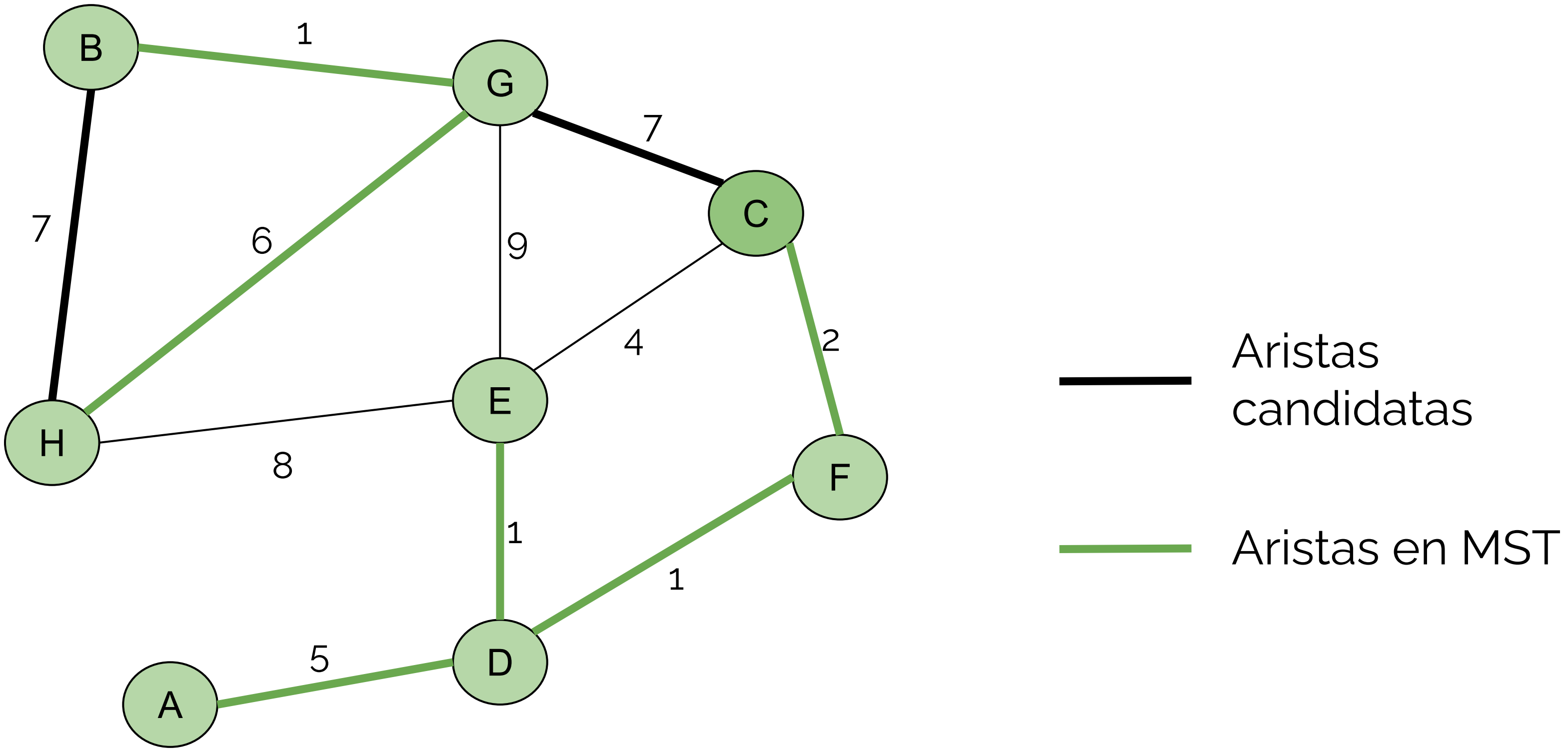
Ejemplo:



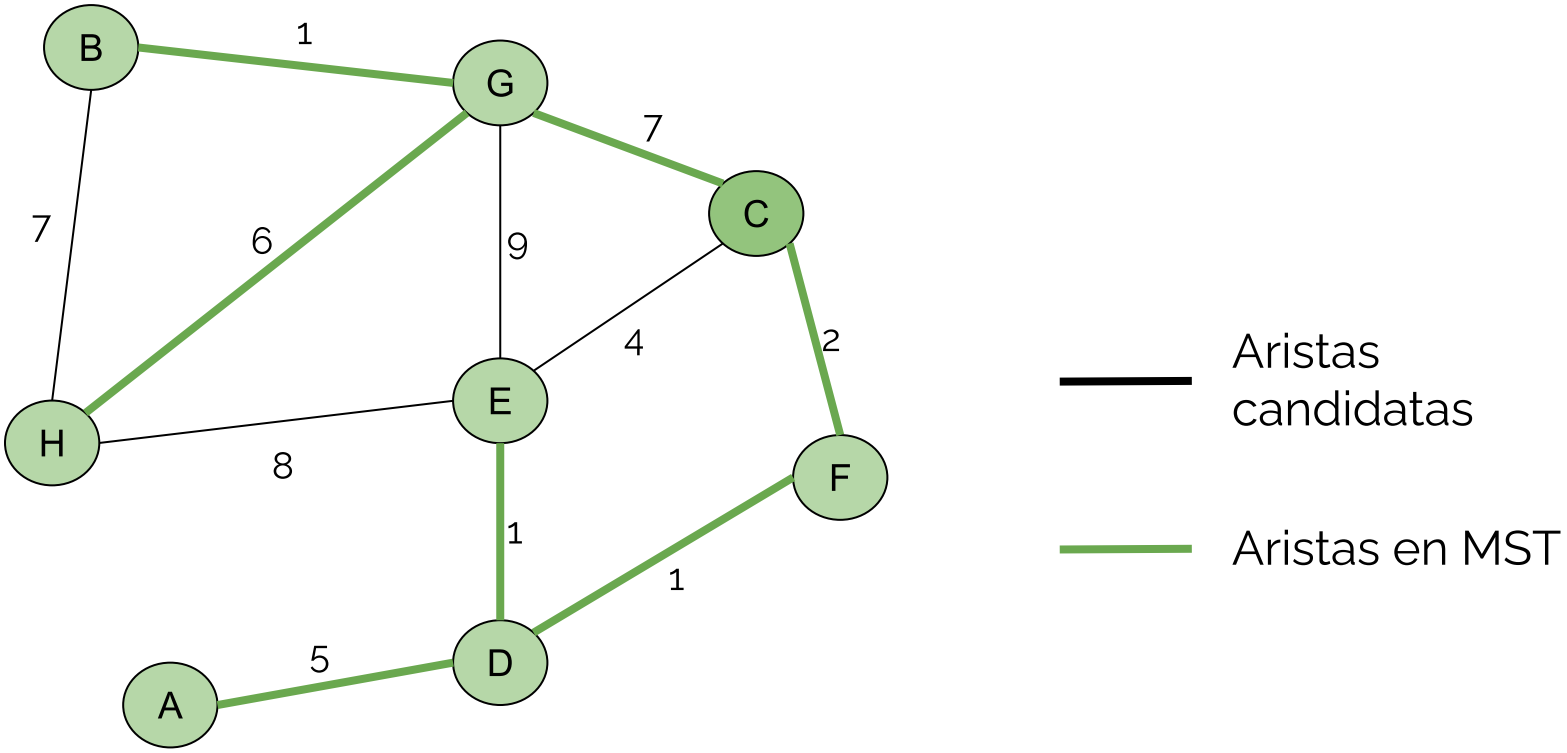
Ejemplo:



Ejemplo:



Ejemplo:



Kruskal

```
kruskal(Grafo G)
	UnionFind T
	E = G.edges
	sort(E) //Por el peso
	for w,u,v in E
		if not T.same_set(u,v)
			T.join(u,v)
			//añadir arista u,v
```

Fin!