PRIM

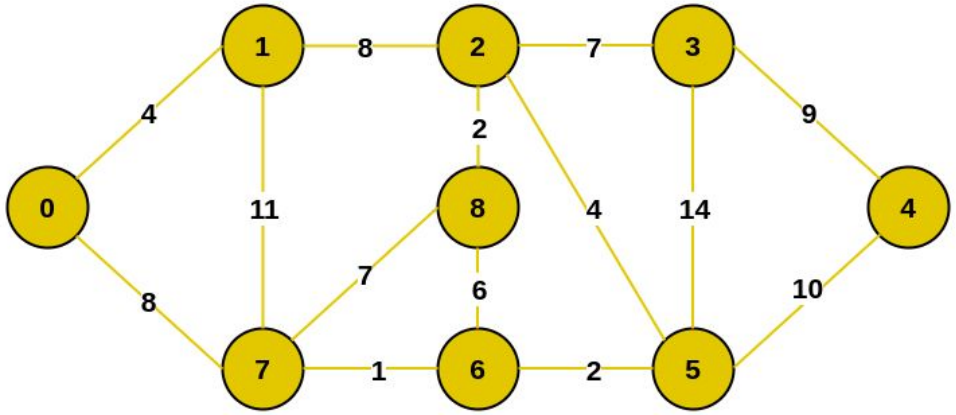
Agustín Gutiérrez

Importante manejarse con los MST!

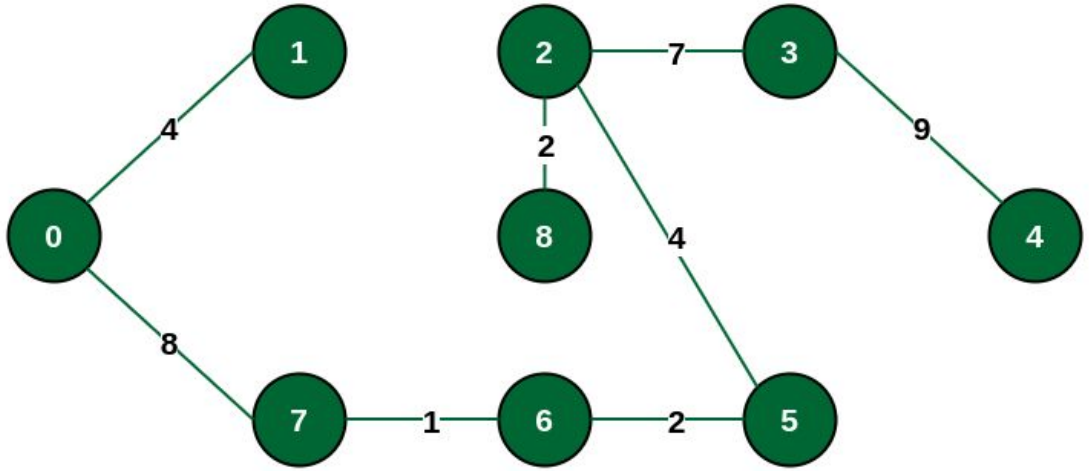
MST = Minimum Spanning Tree

Es un árbol que minimiza el costo total (Suma de los costos de las aristas)

ES EL MÍNIMO, PERO NO NECESARIAMENTE ES ÚNICO



Example of a Graph



The final structure of MST

I3 2017-1

3. Sea $G = (V, E)$ un grafo no dirigido y $T \subseteq E$ el único árbol de cobertura de costo mínimo (MST) de G. Suponga ahora que G' se construye agregando nodos a G y aristas que conectan estos nuevos nodos con nodos de G. Finalmente, sea T' un MST de G'.

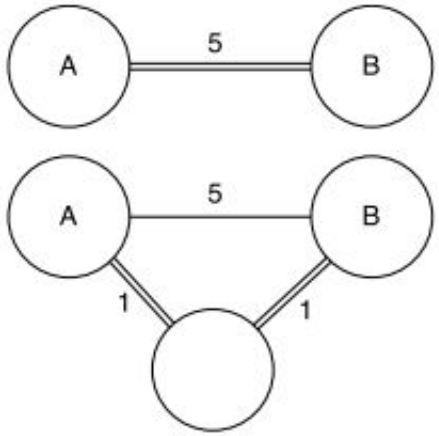
a) (1/3) Demuestre que no necesariamente $T \subseteq T'$.

I3 2017-1

3. Sea $G = (V, E)$ un grafo no dirigido y $T \subseteq E$ el único árbol de cobertura de costo mínimo (MST) de G. Suponga ahora que G' se construye agregando nodos a G y aristas que conectan estos nuevos nodos con nodos de G. Finalmente, sea T' un MST de G'.

a) (1/3) Demuestre que no necesariamente $T \subseteq T'$.

Respuesta: Basta con dar un contraejemplo:



Las aristas dobles muestran el árbol del grafo. En la primera imagen el árbol T es $\{(A, B)\}$, mientras que en el segundo grafo el árbol T' es $\{(A, C), (B, C)\}$. Es evidente que $T \notin T'$

I3 2017-1

3. Sea $G = (V, E)$ un grafo no dirigido y $T \subseteq E$ el único árbol de cobertura de costo mínimo (MST) de G. Suponga ahora que G' se construye agregando nodos a G y aristas que conectan estos nuevos nodos con nodos de G. Finalmente, sea T' un MST de G'.

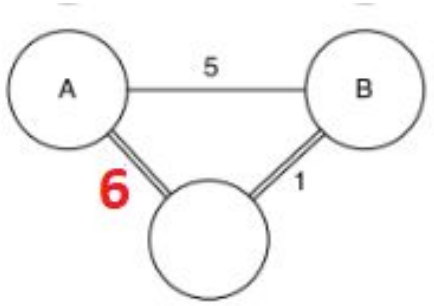
b) (2/3) Dé una condición necesaria y suficiente para garantizar que $T \subseteq T'$. Demuéstrela.

I3 2017-1

3. Sea $G = (V, E)$ un grafo no dirigido y $T \subseteq E$ el único árbol de cobertura de costo mínimo (MST) de G. Suponga ahora que G' se construye agregando nodos a G y aristas que conectan estos nuevos nodos con nodos de G. Finalmente, sea T' un MST de G'.

b) (2/3) Dé una condición necesaria y suficiente para garantizar que $T \subseteq T'$. Demuéstrela.

Respuesta: Para asegurar que $T \in T'$, se debe asegurar para cada arista $(u, v) \in T$ que: Si se genera un camino c nuevo que conecta u con v, entonces al menos existe una arista $a \in c$ tal que el costo de a es mayor al costo de (u, v).



Demostración:
Suficiencia: Dado que el algoritmo de kruscal es correcto, podemos decir lo siguiente:
Dados dos nodos $u, v \in G$ tal que $(u, v) \in T$, se tiene que existe un camino nuevo que conecta u con v en el cual existe una arista a más cara que (u, v). En algun paso de la ejecución del algoritmo de kruscal se tendrá que u y v están en dos grupos no conectados de nodos. Se puede asegurar que el algoritmo de kruscal no conectará los grupos de nodos de u con el de v por la arista a ya que primero se revisan las aristas más bartas, por lo que primero se conectarían a través de (u, v). Por lo tanto, esta propiedad es suficiente.

Necesaria: Si no se cumple esta propiedad entonces ejecutando el algoritmo de kruscal se conectaría u con v a través del nuevo camino, por lo que no se agregaría (u, v) al árbol T'. Por lo tanto, si la propiedad, puede pasar que $T \notin T'$.

Lo PRIMero: el pseudocódigo

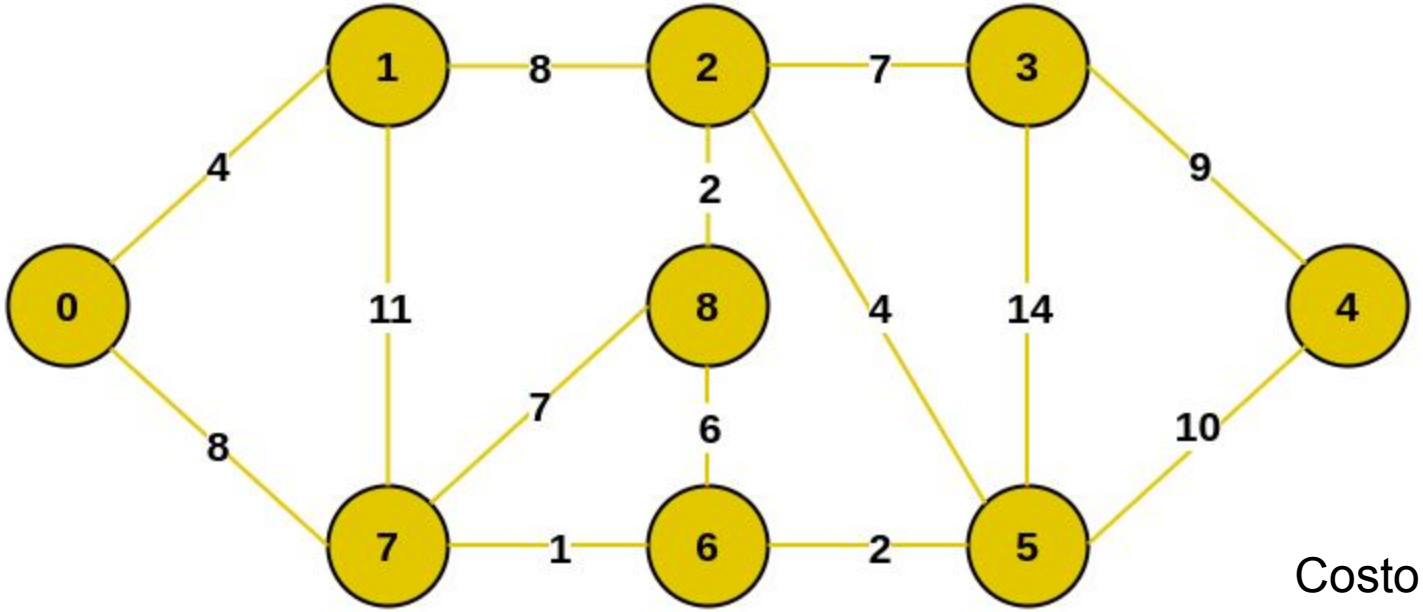
```
Procedure prims
        G – input graph
        U – random vertex
        V – vertices in graph G
begin
        T = Ø;
        U = { 1 };
        while (U ≠ V)
         let (u, v) be the least cost edge such that u ∈ U and v ∈ V - U;
         T = T U {(u, v)}
         U = U U {v}
end procedure
```

T: Es el MST, comienza vacío
U: Un nodo que ha sido visitado
V: Nodos del grafo G
V – U: Los nodos de G que aún no han sido visitados

En este pseudocódigo nos detenemos cuando U = V porque esto significa que ya pasamos por todos los nodos!

De esta forma, se construye un MST utilizando aristas que cruzan cortes (Aristas que unen 2 grafos que no están conectados entre sí), donde el primer grafo corresponde a T, y el segundo es el nodo que se une al MST al añadir la arista (u, v)

Ejemplo

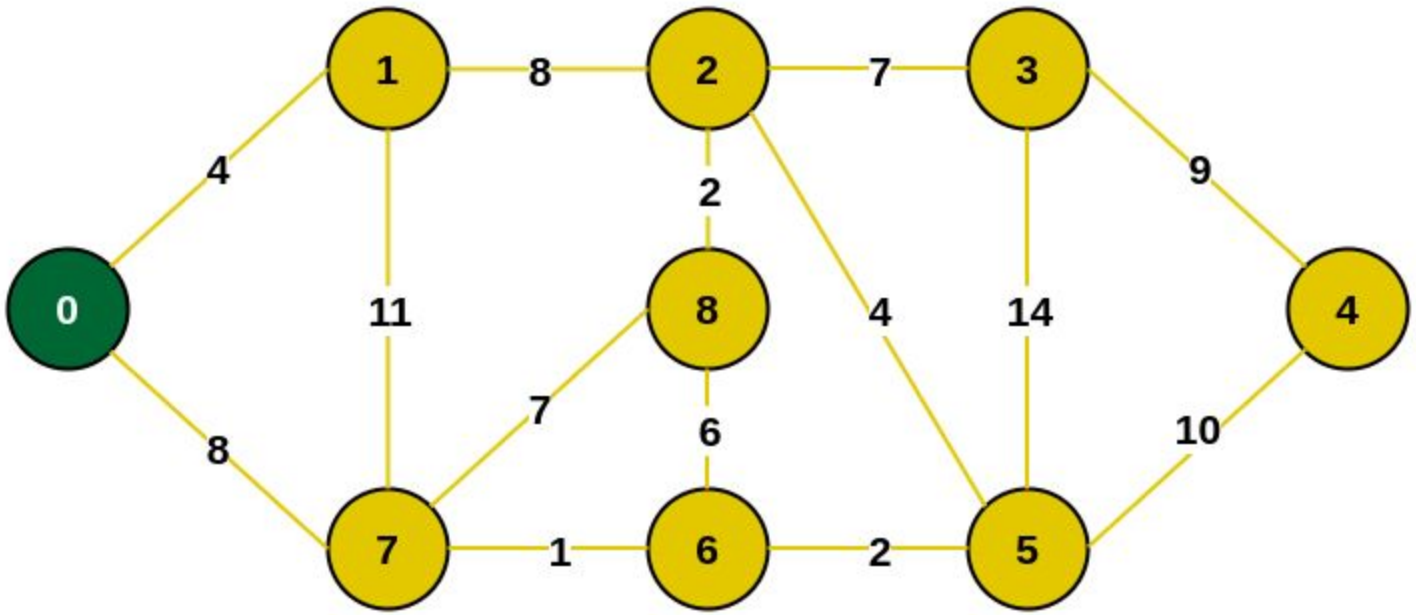


Example of a Graph

Costo total = 93

R = { }
R'= {0, 1, 2, 3, 4, 5, 6, 7, 8}

Fuente: https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/

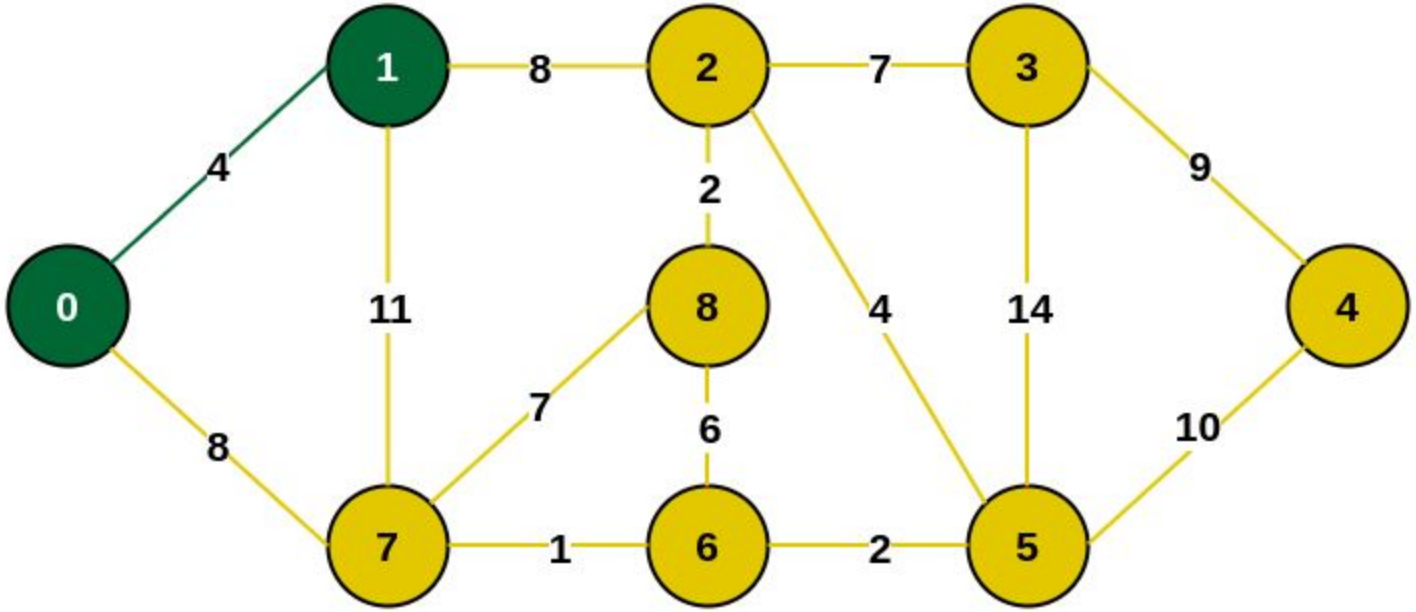


Select an arbitrary starting vertex. Here we have selected 0

R = {0}

R'= {1, 2, 3, 4, 5, 6, 7, 8}

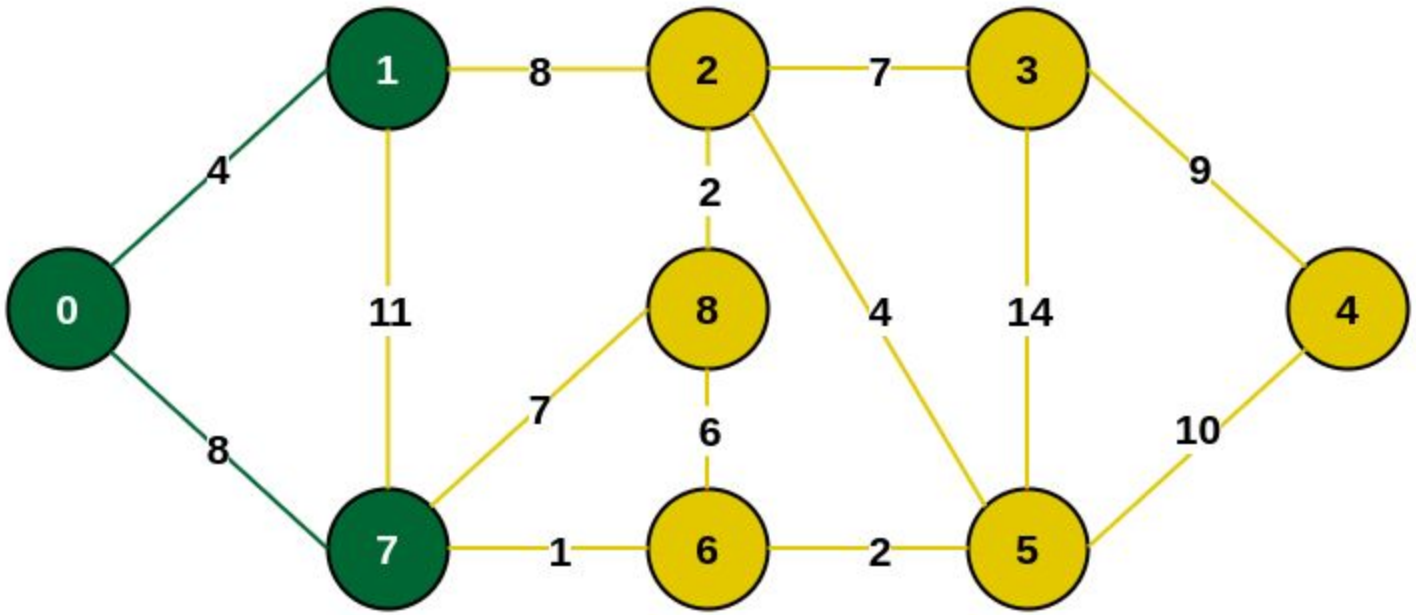
Escogemos arbitrariamente el nodo 0



Minimum weighted edge from MST to other vertices is 0-1 with weight 4

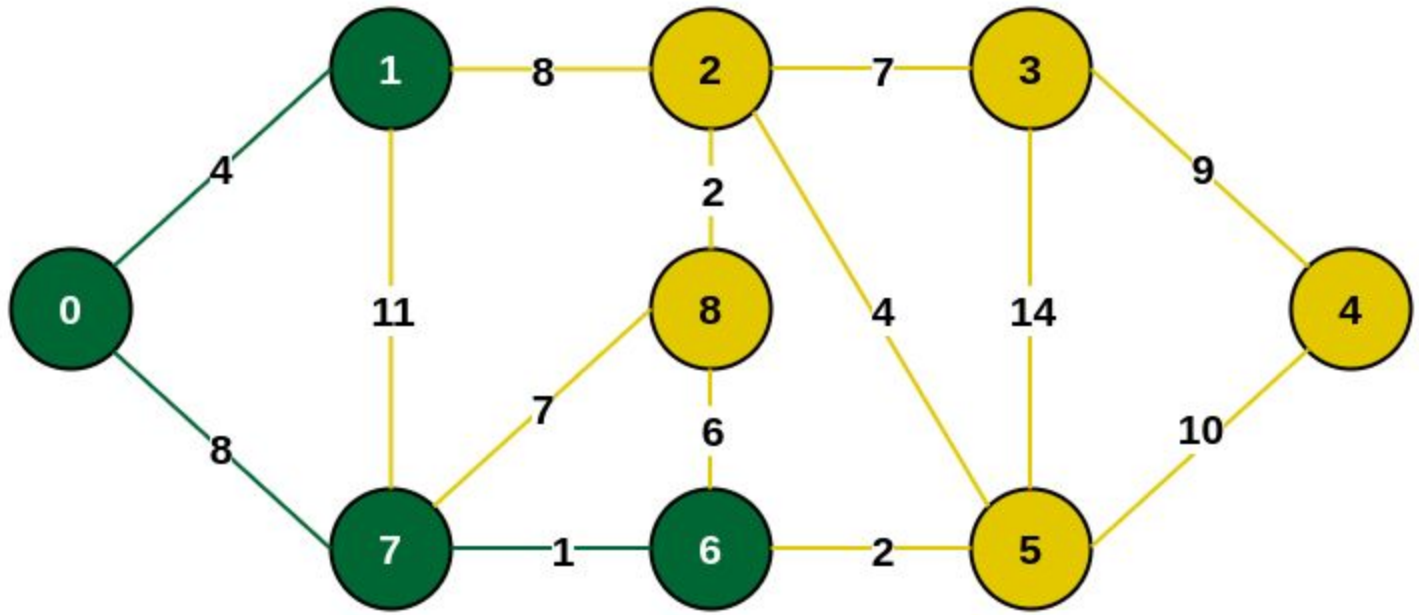
R = {0, 1}
R'= {2, 3, 4, 5, 6, 7, 8}

De todas las aristas disponibles que van desde un nodo visitado a un nodo no visitado, escogemos la de menor costo



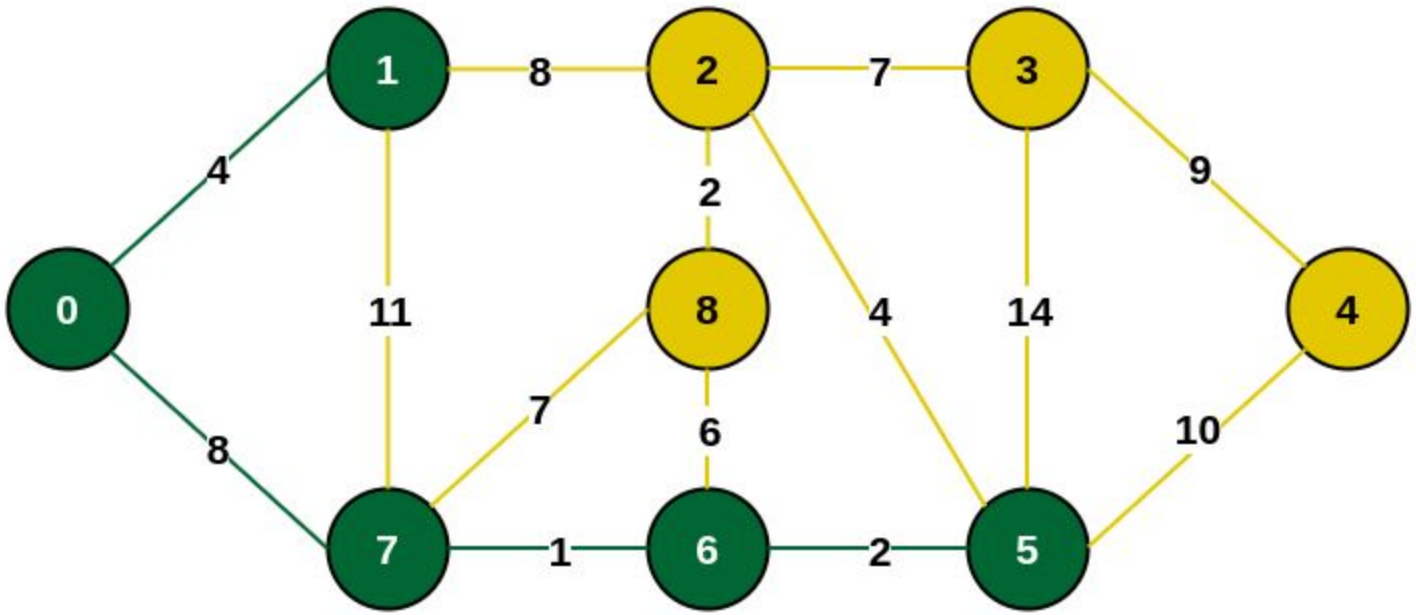
Minimum weighted edge from MST to other vertices is 0-7 with weight 8

R = {0, 1, 7}
R'= {2, 3, 4, 5, 6, 8}



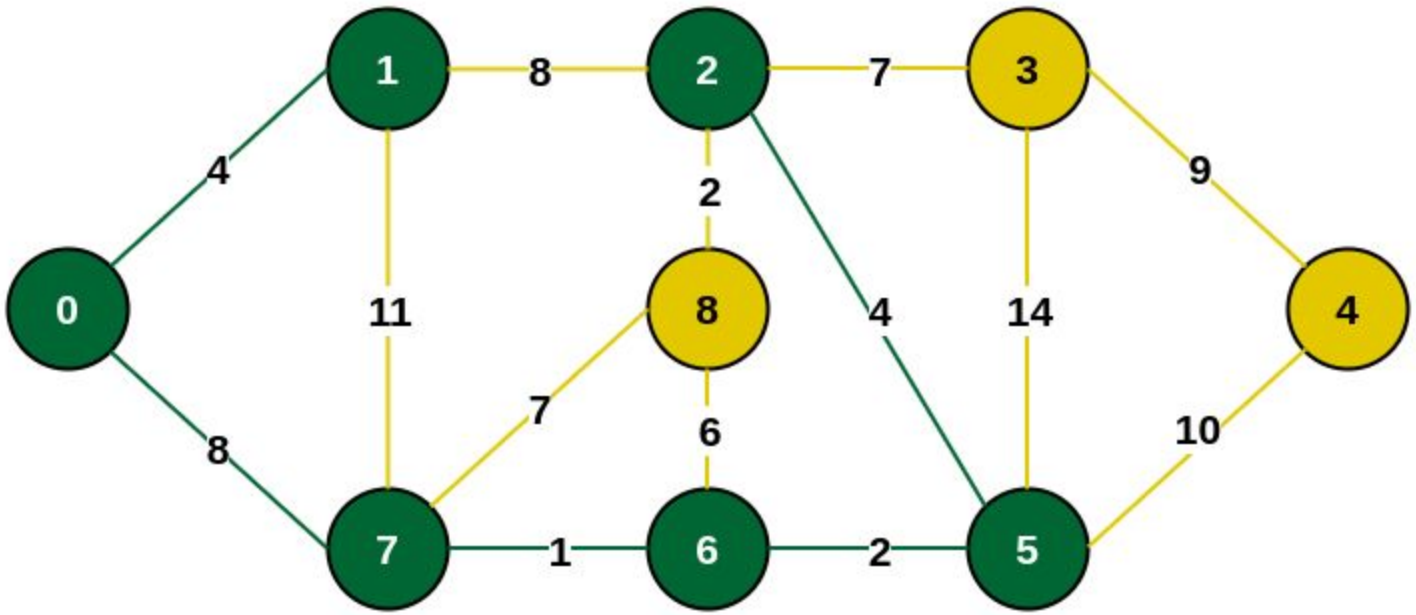
Minimum weighted edge from MST to other vertices is 7-6 with weight 1

R = {0, 1, 7, 6}
R'= {2, 3, 4, 5, 8}



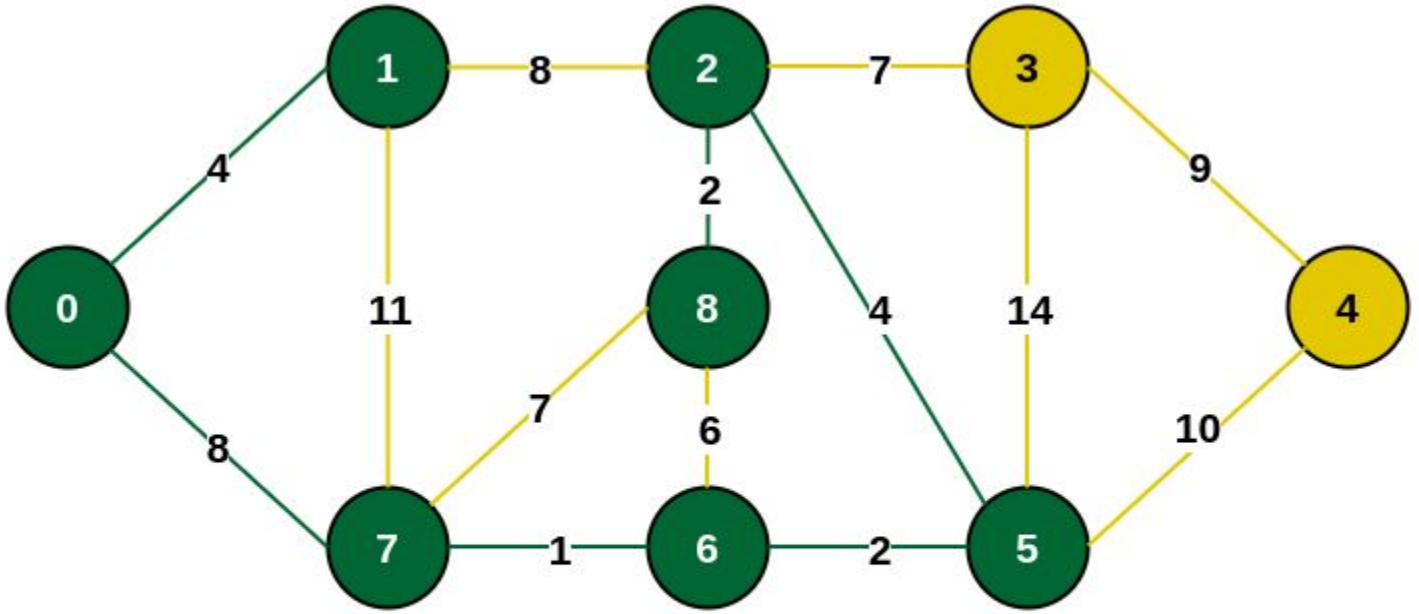
Minimum weighted edge from MST to other vertices is 6-5 with weight 2

R = {0, 1, 7, 6, 5}
R'= {2, 3, 4, 8}



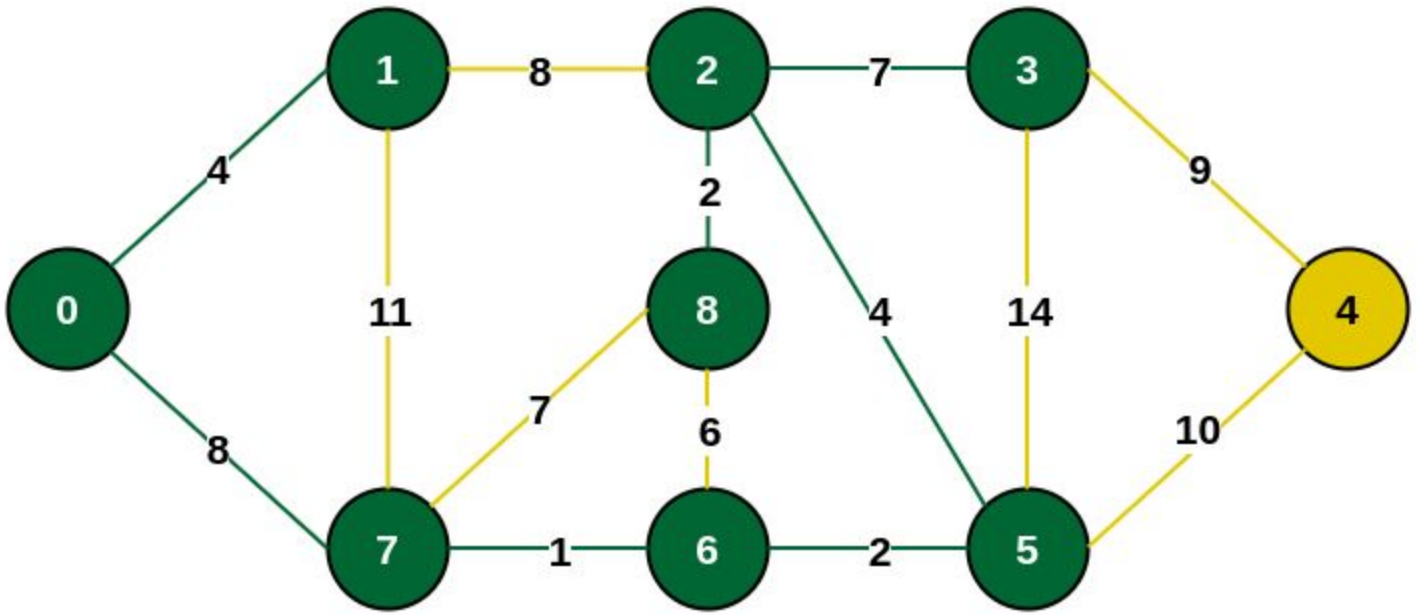
Minimum weighted edge from MST to other vertices is 5-2 with weight 4

R = {0, 1, 7, 6, 5, 2}
R'= {3, 4, 8}



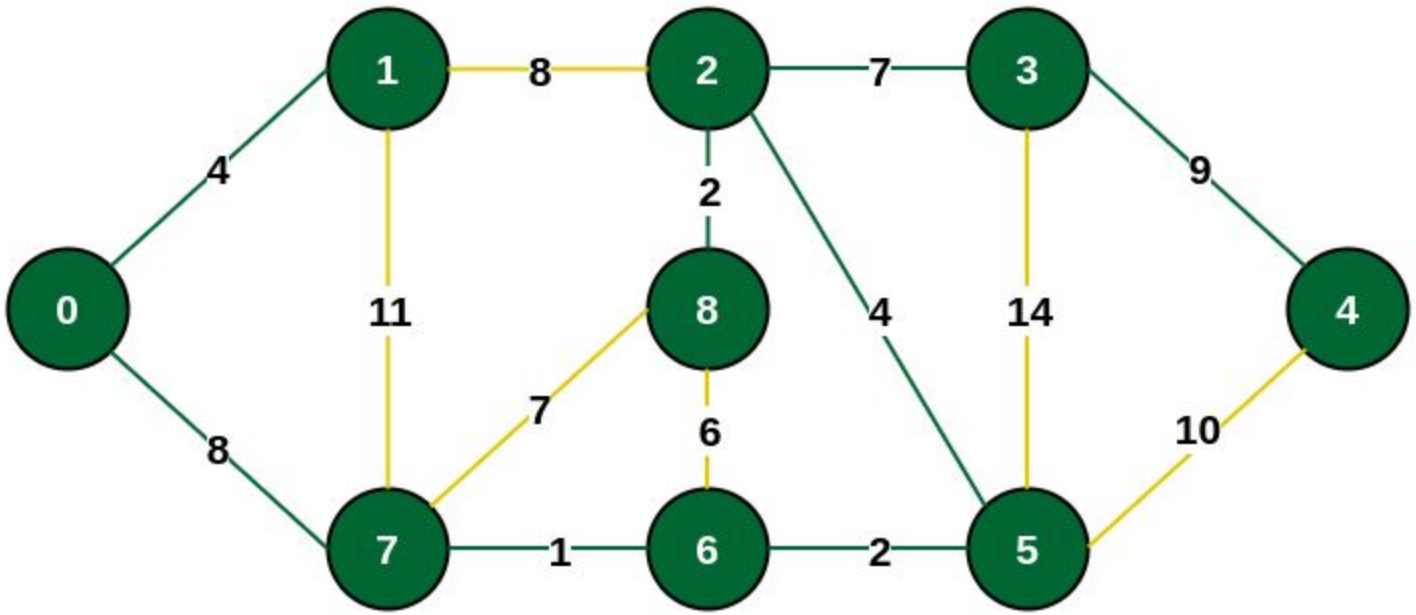
Minimum weighted edge from MST to other vertices is 2-8 with weight 2

R = {0, 1, 7, 6, 5, 2, 8}
R'= {3, 4}



Minimum weighted edge from MST to other vertices is 2-3 with weight 7

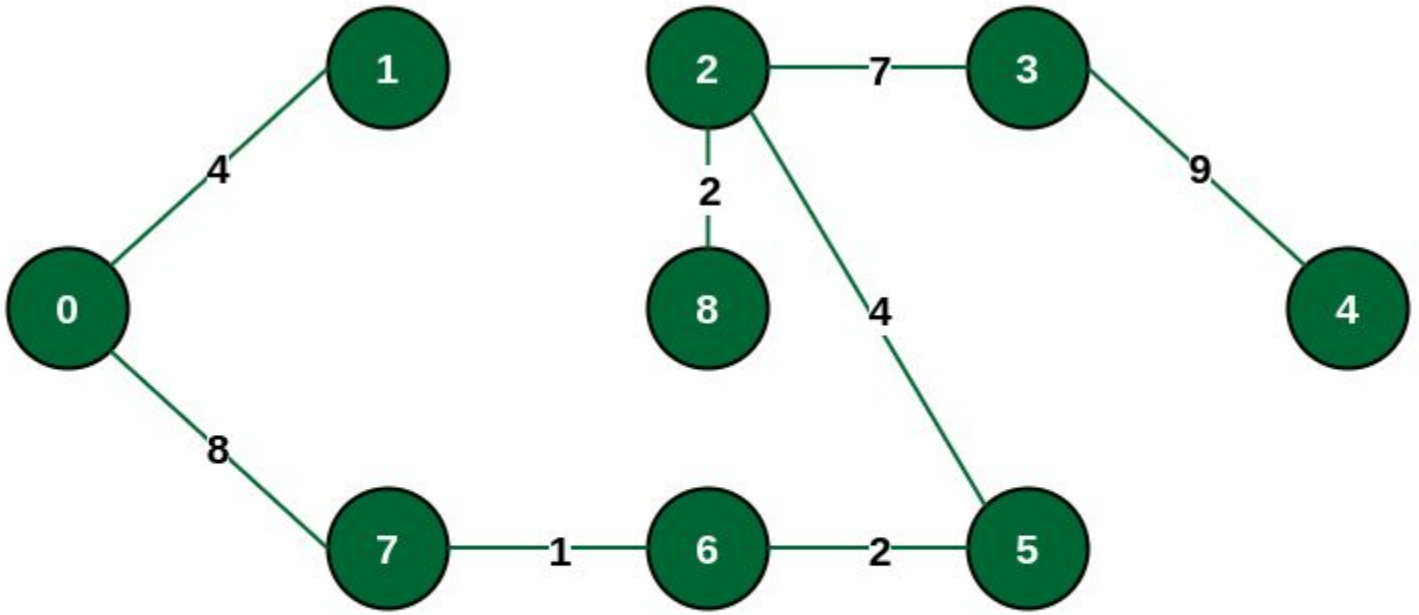
R = {0, 1, 7, 6, 5, 2, 8, 3}
R'= {4}



Minimum weighted edge from MST to other vertices is 3-4 with weight 9

R = {0, 1, 7, 6, 5, 2, 8, 3, 4}
R'= { }

MST, hemos minimizado los costos totales!

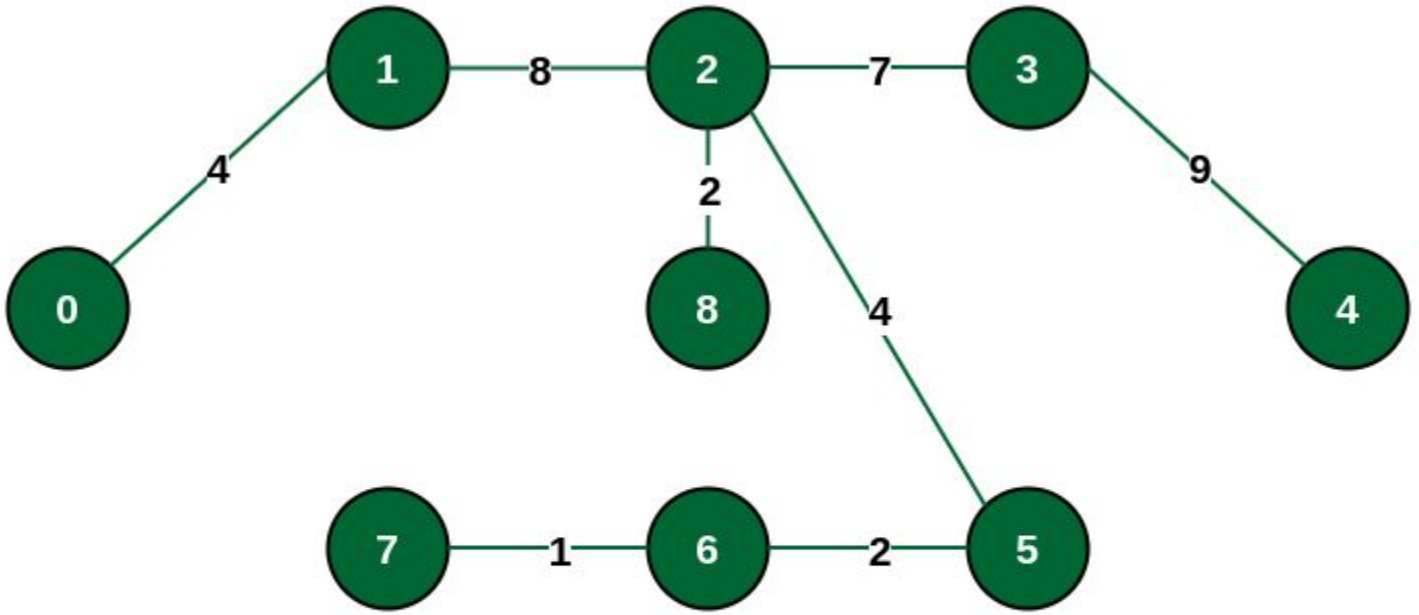


The final structure of MST

Nuevo Costo total = 37

Costo total anterior = 93

MST Alternativo



Alternative MST structure

Costo total = 37

Por definición de MST, no puede existir otro árbol con menor costo total, pero si puede existir otro con el mismo costo total!

2018-2

3. MSTs

Considera el siguiente grafo no direccional con costos, representado matricialmente:

	a	*b*	*c*	*d*	*e*	*f*	*g*
a		2	4	1			
b	2			3	10		
c	4			2		5	
d	1	3	2		7	8	4
e		10		7			6
f			5	8			1
g				4	6	1	

	¿sol?	*dist*	*padre*
a	*F*	0	–
b	*F*	∞	–
c	*F*	∞	–
d	*F*	∞	–
e	*F*	∞	–
f	*F*	∞	–
g	*F*	∞	–

Ejecuta paso a paso el algoritmo de Prim para determinar un árbol de cobertura de costo mínimo, tomando ***a*** como vértice de partida. En cada paso muestra la versión actualizada de la tabla a la derecha, en que *¿sol?* indica si el vértice ya está en la solución: *V* o *F*.

HUH?

	0		
	¿sol?	dist	padre
a	*F*	0	–
b	*F*	∞	–
c	*F*	∞	–
d	*F*	∞	–
e	*F*	∞	–
f	*F*	∞	–
g	*F*	∞	–

	1		
	¿sol?	dist	padre
a	*V*	0	–
b	*F*	2	a
c	*F*	4	a
d	*F*	1	a
e	*F*	∞	–
f	*F*	∞	–
g	*F*	∞	–

	2		
	¿sol?	dist	padre
a	*V*	0	–
b	*F*	2	a
c	*F*	2	d
d	*V*	1	a
e	*F*	7	d
f	*F*	8	d
g	*F*	4	d

	3		
	¿sol?	dist	padre
a	*V*	0	–
b	*V*	2	a
c	*F*	2	d
d	*V*	1	a
e	*F*	7	d
f	*F*	8	d
g	*F*	4	d

2018-2 Solución

Lo PRIMero que tienen que hacer:

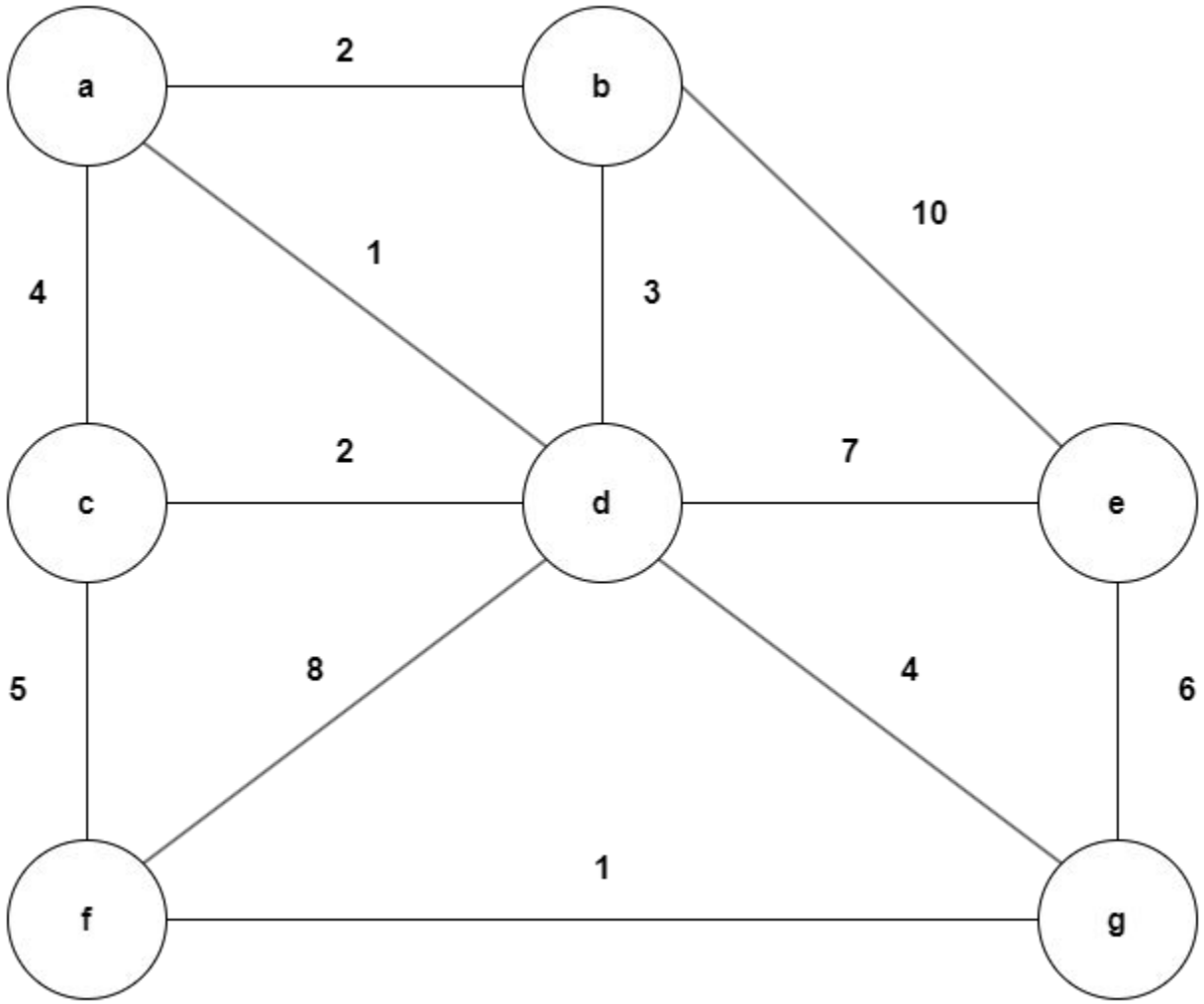
-A partir de la matriz armar el dibujo del grafo (Es fácil marearse y equivocarse mirando la matriz).

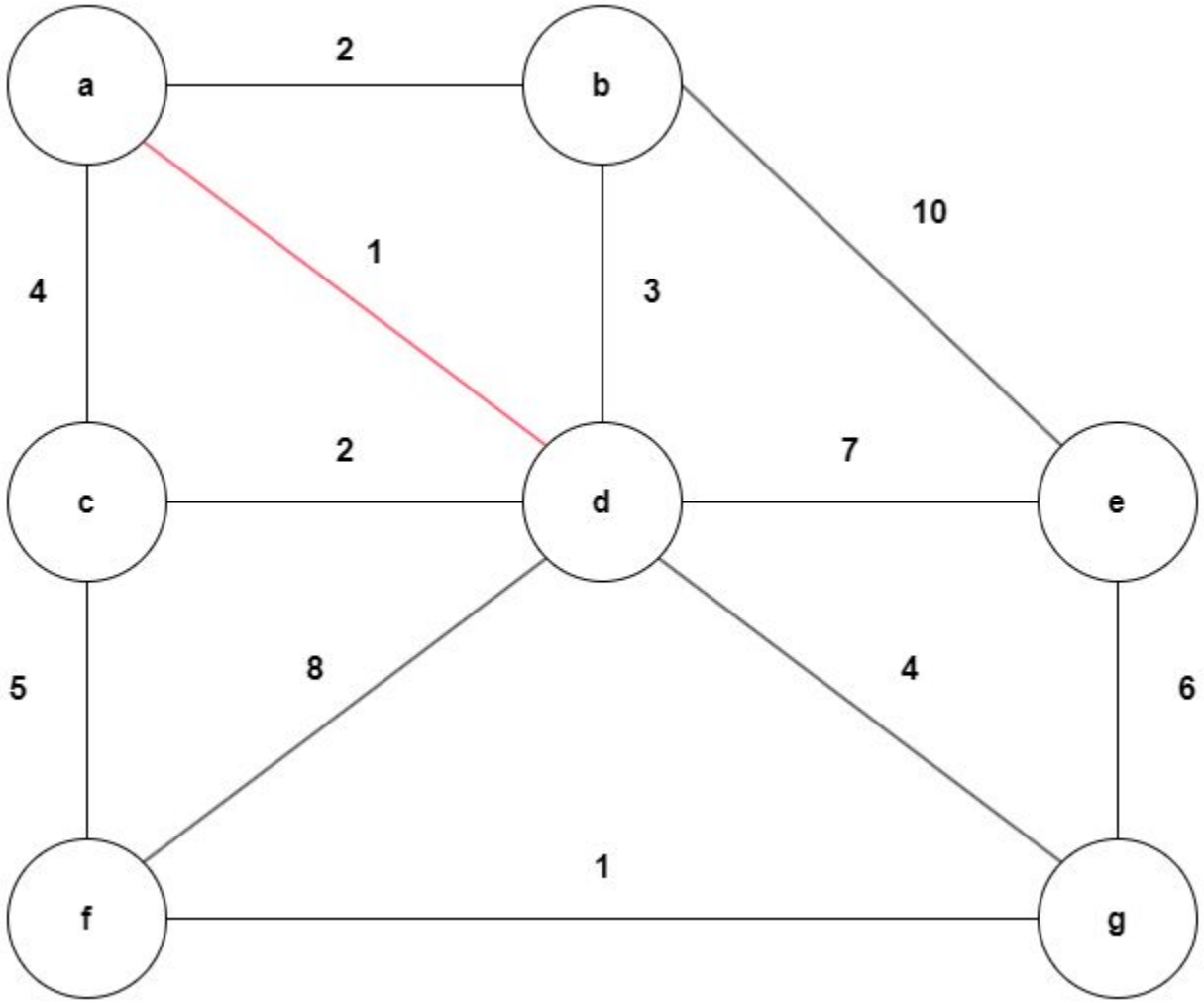
-Darse cuenta de que es V y que es F.

-**Padre** = nodo del MST que “da acceso con el menor costo a este nodo que aún no está en el MST”

-**Dist** = Distancia al nodo padre.

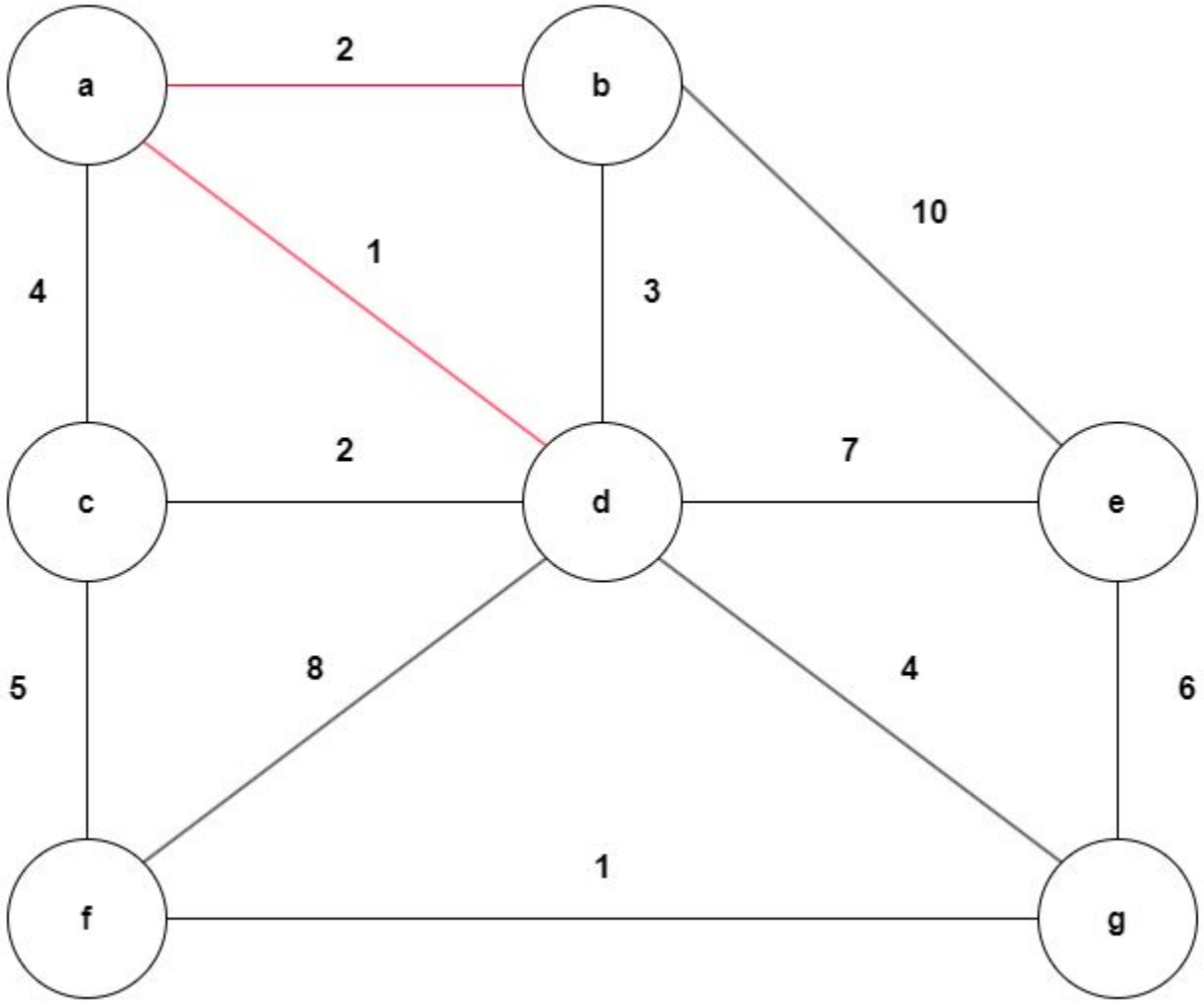
Ahora sipue



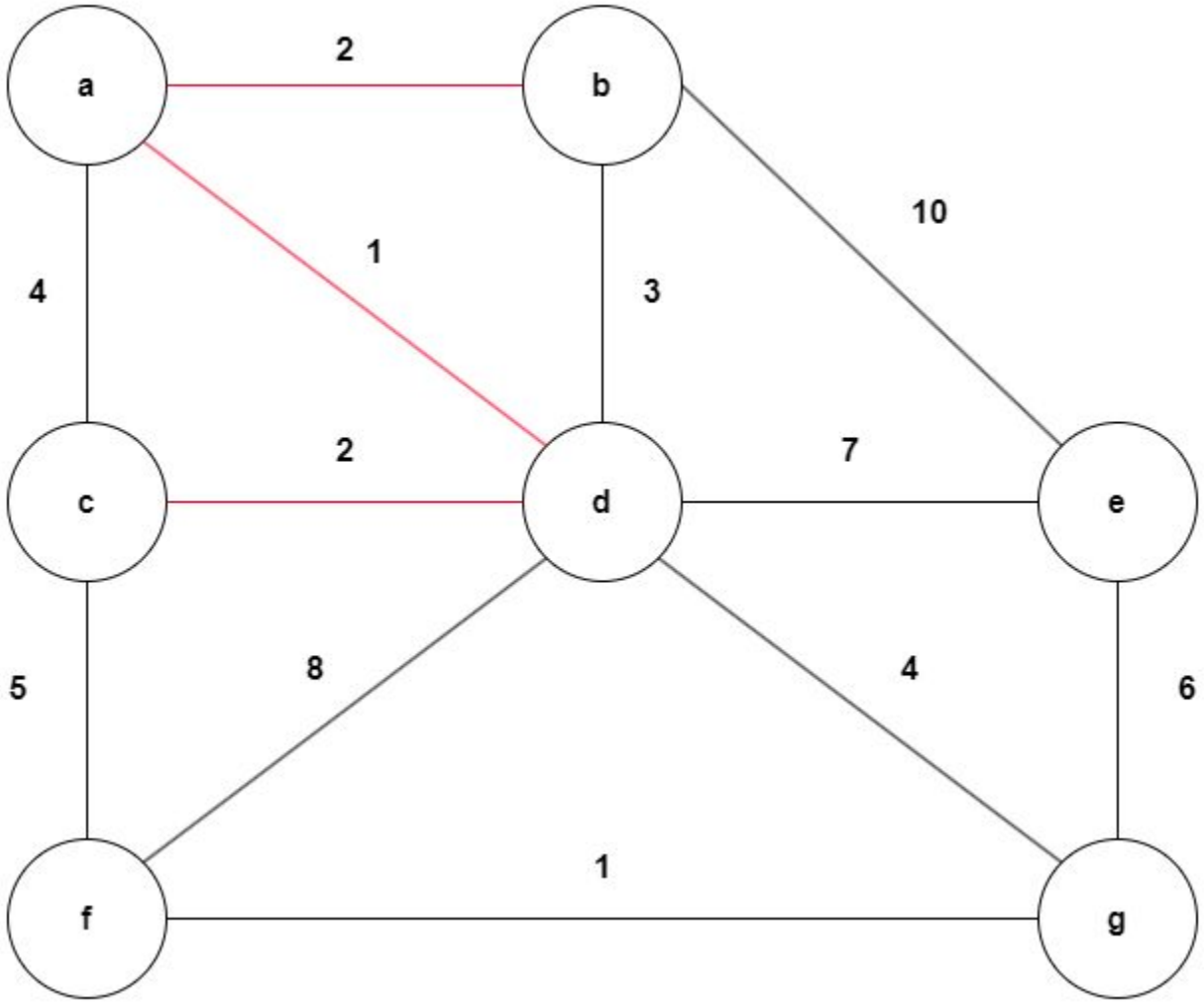


	1		
	¿sol?	dist	padre
a	*V*	0	–
b	*F*	2	a
c	*F*	4	a
d	*F*	1	a
e	*F*	∞	–
f	*F*	∞	–
g	*F*	∞	–

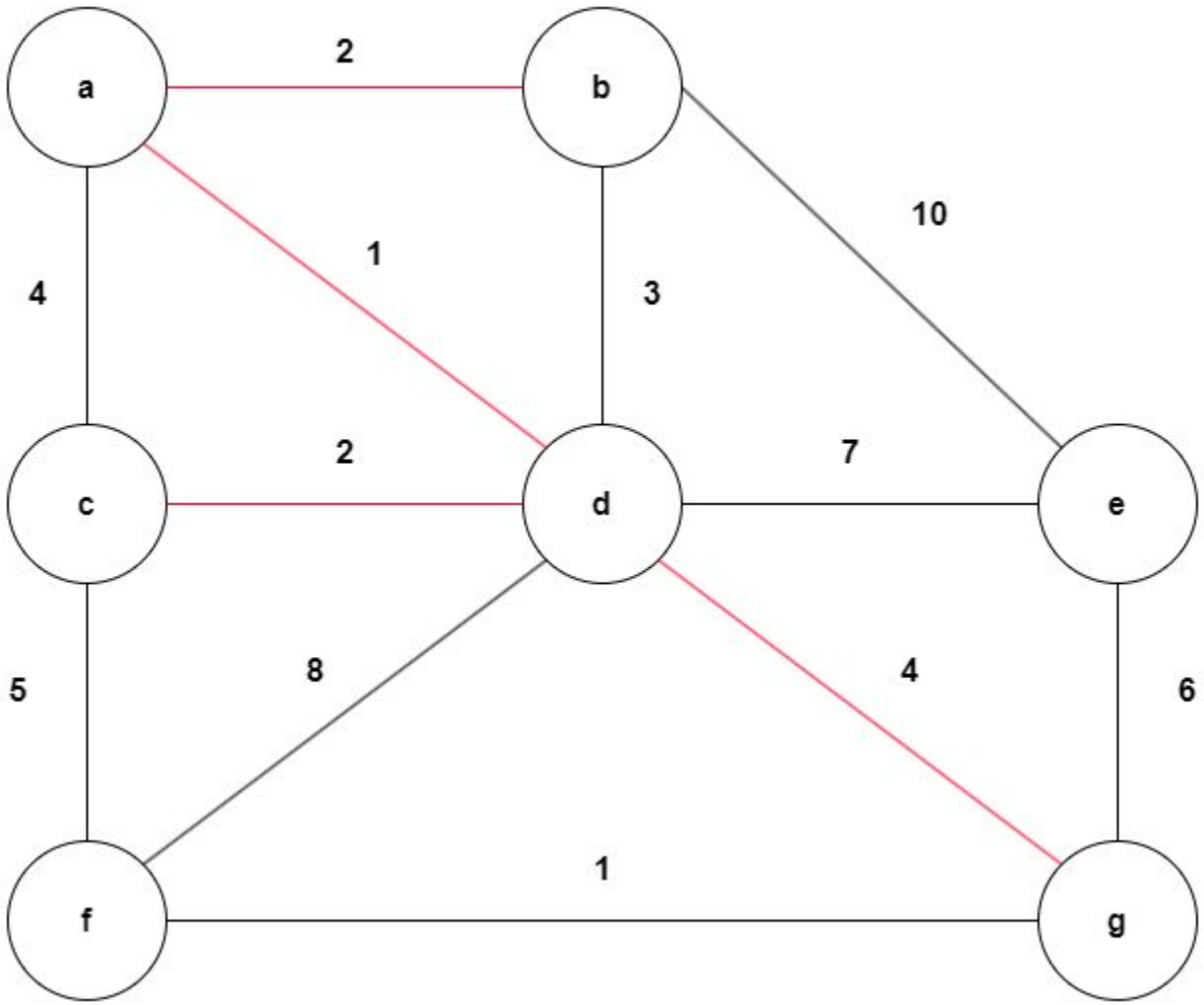
	2		
	¿sol?	dist	padre
a	*V*	0	–
b	*F*	2	a
c	*F*	2	d
d	*V*	1	a
e	*F*	7	d
f	*F*	8	d
g	*F*	4	d



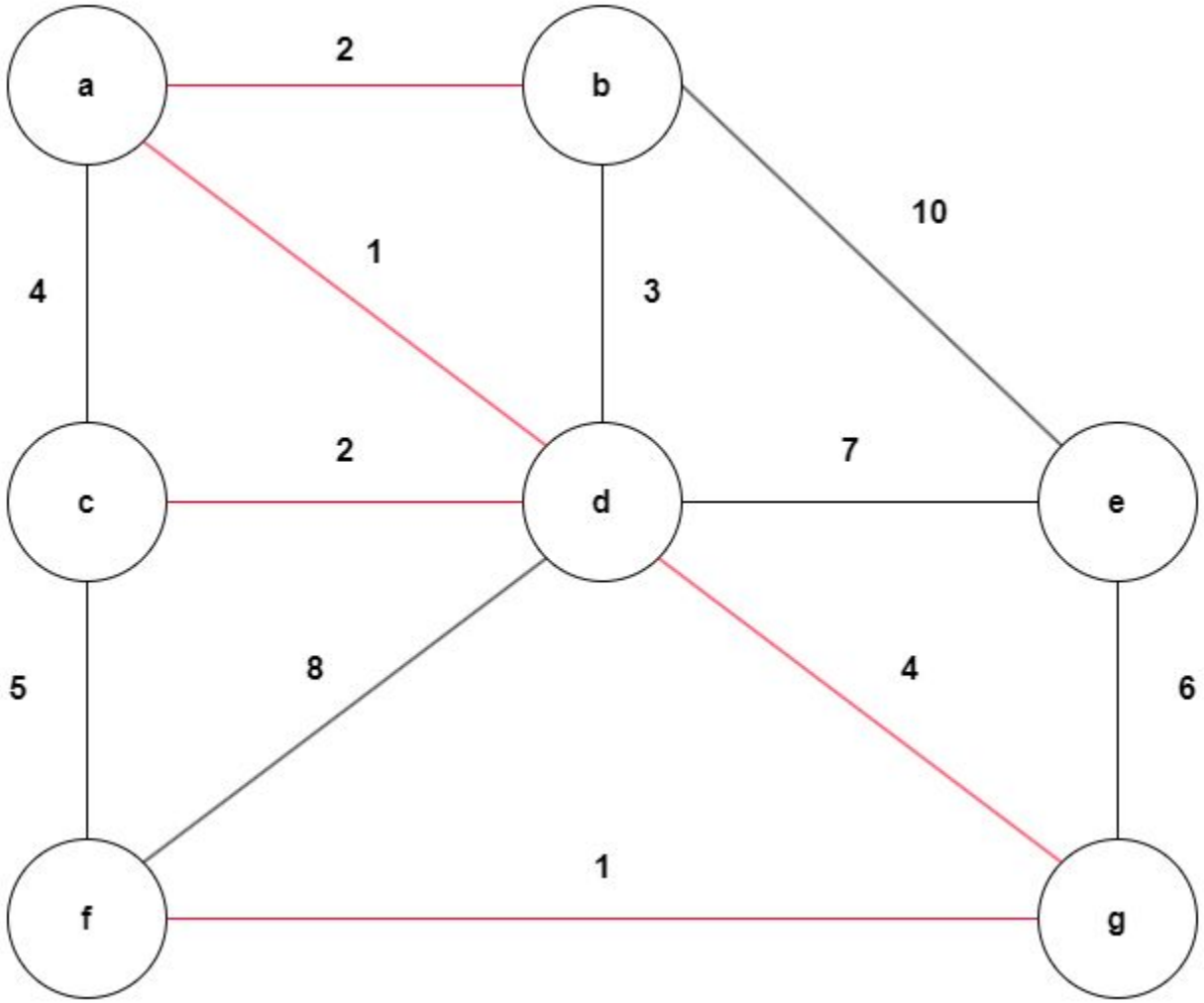
	3		
	¿sol ?	dist	padre
a	*V*	0	–
b	*V*	2	a
c	*F*	2	d
d	*V*	1	a
e	*F*	7	d
f	*F*	8	d
g	*F*	4	d



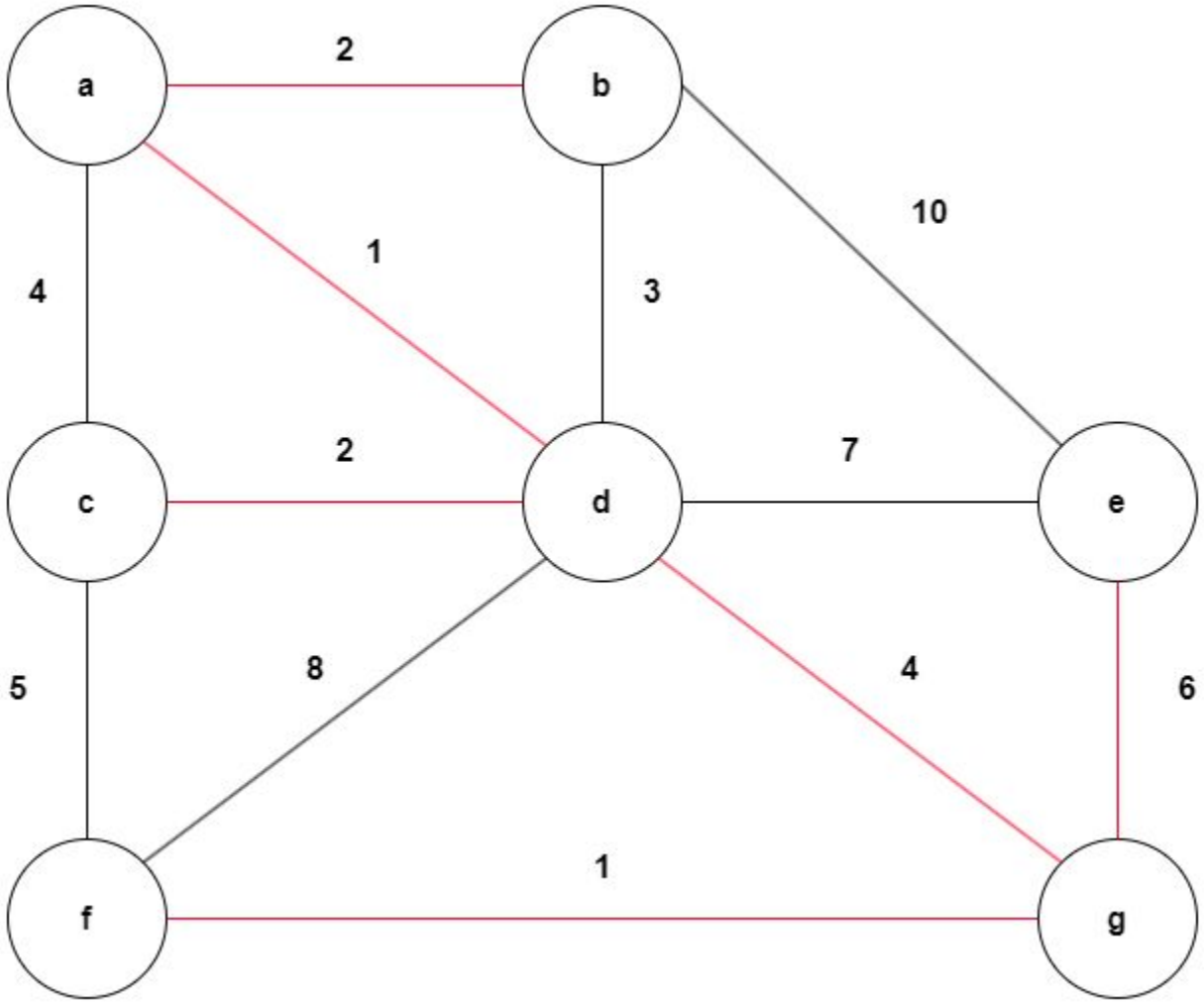
	4		
	¿sol?	dist	padre
a	*V*	0	–
b	*V*	2	a
c	*V*	2	d
d	*V*	1	a
e	*F*	7	d
f	*F*	5	c
g	*F*	4	d



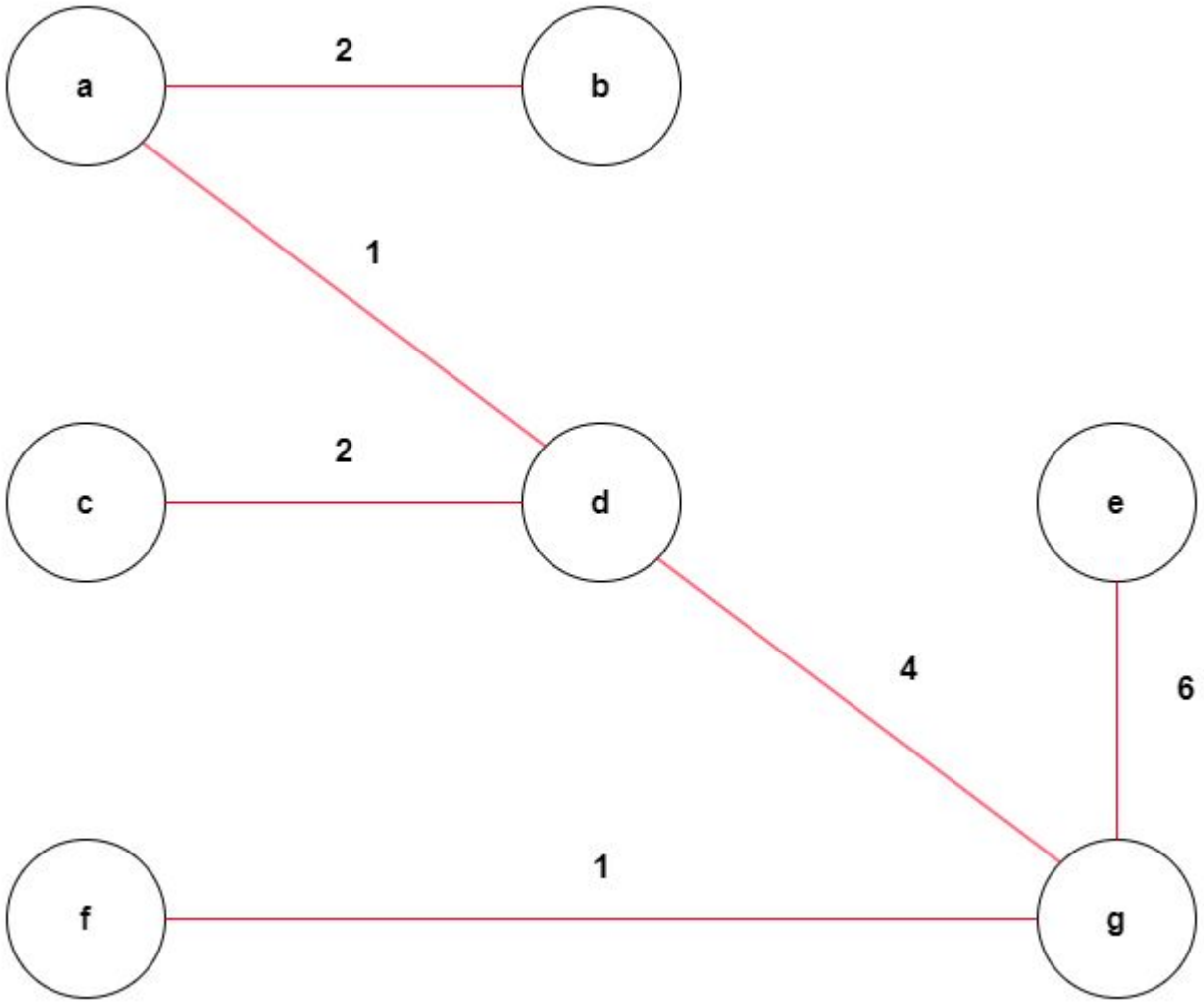
	5		
	¿sol?	dist	padre
a	*V*	0	–
b	*V*	2	a
c	*V*	2	d
d	*V*	1	a
e	*F*	6	g
f	*F*	1	g
g	*V*	4	d



	6		
	¿sol ?	dist	pad re
a	*V*	0	–
b	*V*	2	a
c	*V*	2	d
d	*V*	1	a
e	*F*	6	g
f	*V*	1	g
g	*V*	4	d



	7		
	¿sol?	dist	padre
a	*V*	0	–
b	*V*	2	a
c	*V*	2	d
d	*V*	1	a
e	*V*	6	g
f	*V*	1	g
g	*V*	4	d


a
b
c
d
e
f
g
2
1
2
4
6
1


Holy Moly!

Una pregunta:

¿Cuando es Prim Mejor que Kruskal?

La diferencia PRIMordial:

Kruskal -> O(E * Log V)

Prim -> O(E * Log V) (Para el caso que le entreguemos los nodos como un heap binario o una LISTA de adyacencia)

PERO, hay una implementación de Prim (Heap Fibonacci) que resulta en

O(E + VLogV) (No es necesario que se aprendan como, esto es más conceptual)

Prim es mejor en grafos de alta densidad (Muchas aristas), dado que Prim recorre los nodos, mientras que Kruskal recorre las Aristas.