Teoría de Grafos II

MSc. (c) Jhosimar George Arias Figueroa

jariasf03@gmail.com

State University of Campinas
Institute of Computing

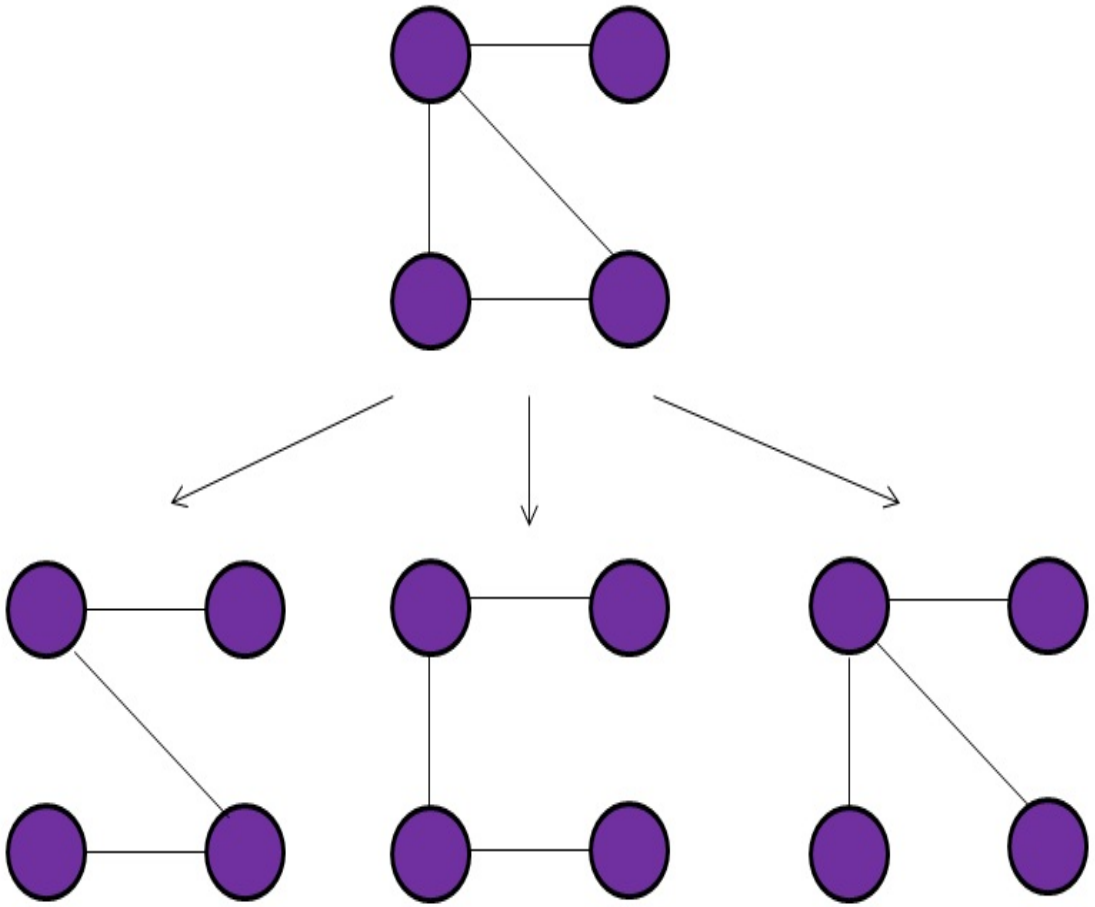
Contenido

Contenido

Árbol de expansión

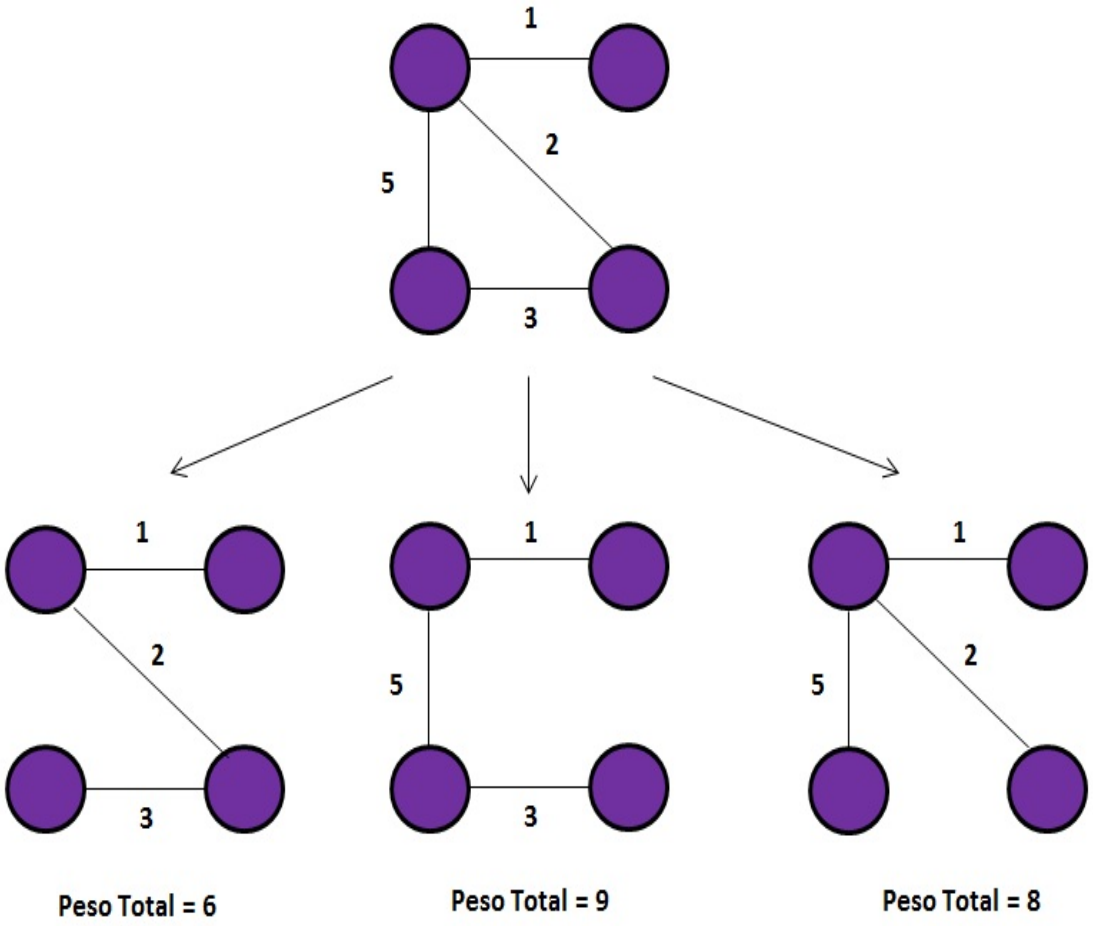
- Un árbol de expansión es un árbol compuesto por todos los vértices y algunas (posiblemente todas) de las aristas.
- Al ser creado un árbol no existirán ciclos, además debe existir una ruta entre cada par de vértices.
- Un grafo puede tener muchos árboles de expansión.

Árbol de expansión



Árbol de expansión mínima

Un árbol de expansión mínima es un árbol compuesto por todos los vértices y cuya suma de sus aristas es la de menor peso.

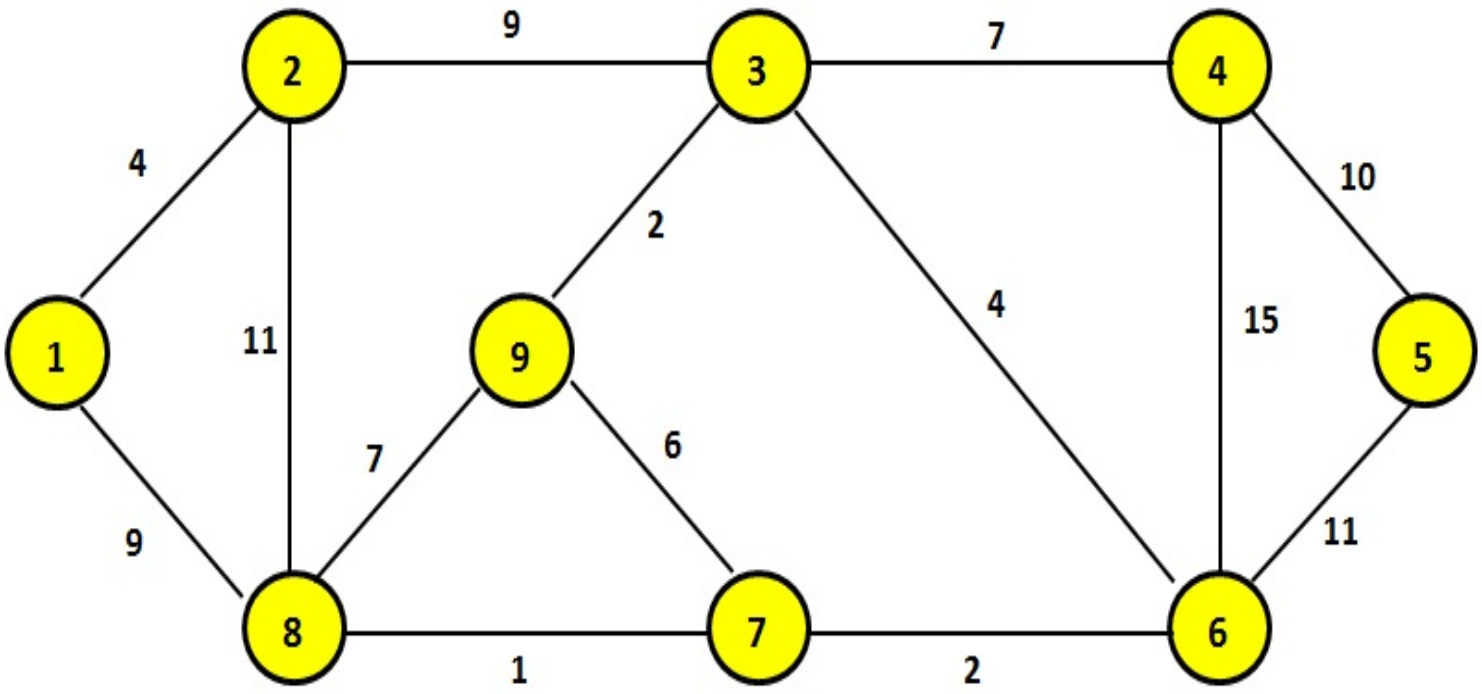


Contenido

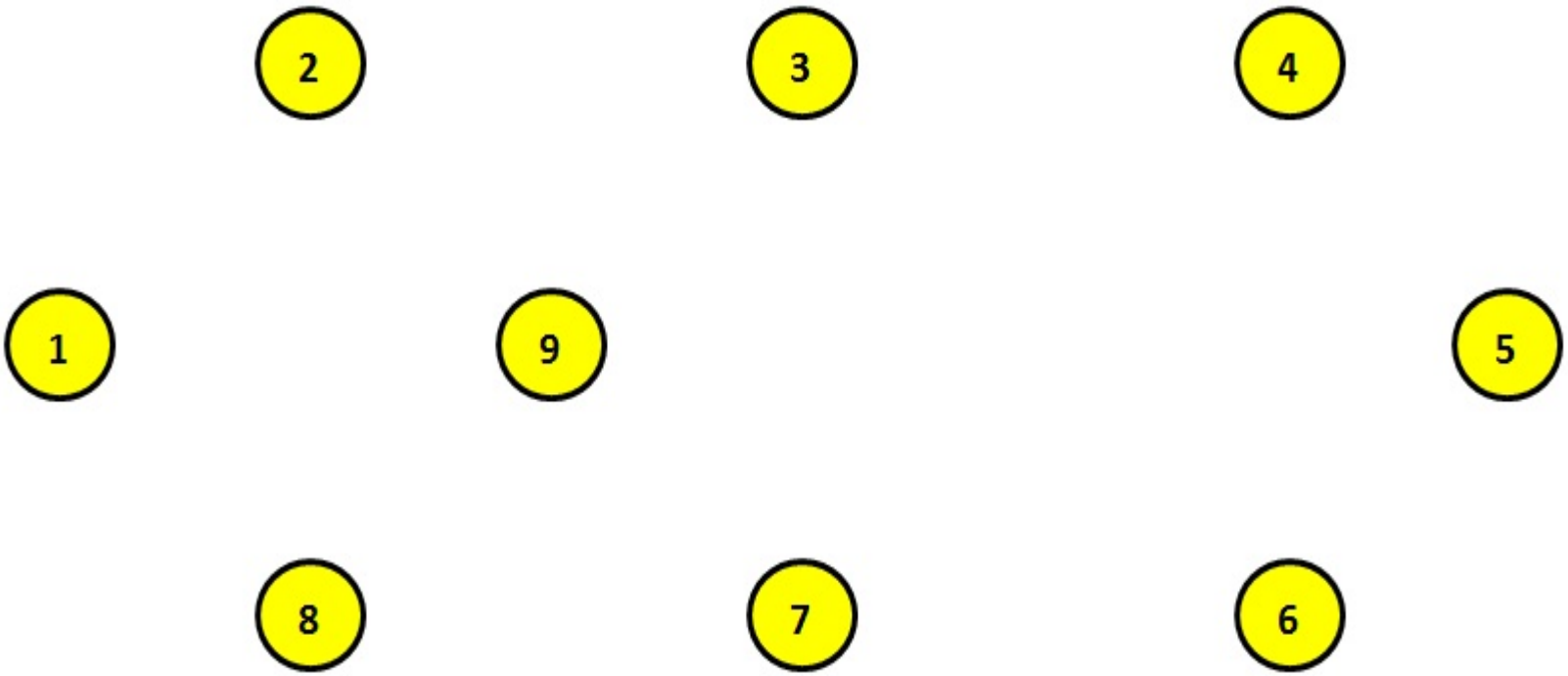
Kruskal

- Es un algoritmo voraz (greedy).
- Recorre todas las aristas del grafo en orden creaciente por peso.
- Vorazmente escoge una arista que no forme un ciclo (Union-Find puede ser usado para poder detectar la presencia de un ciclo).
- Cuando hayamos recorrido todas las aristas, el resultado será el árbol de expansión mínima.
- Complejidad: $O(Elog(E))$

Kruskal

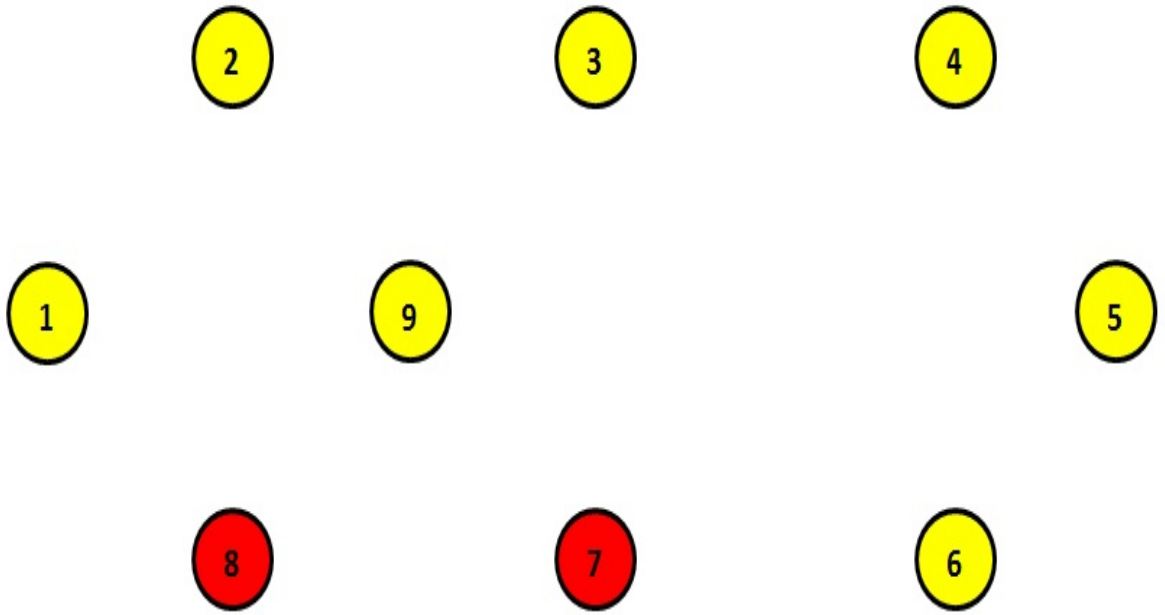


Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	3	4	5	6	7	8	9

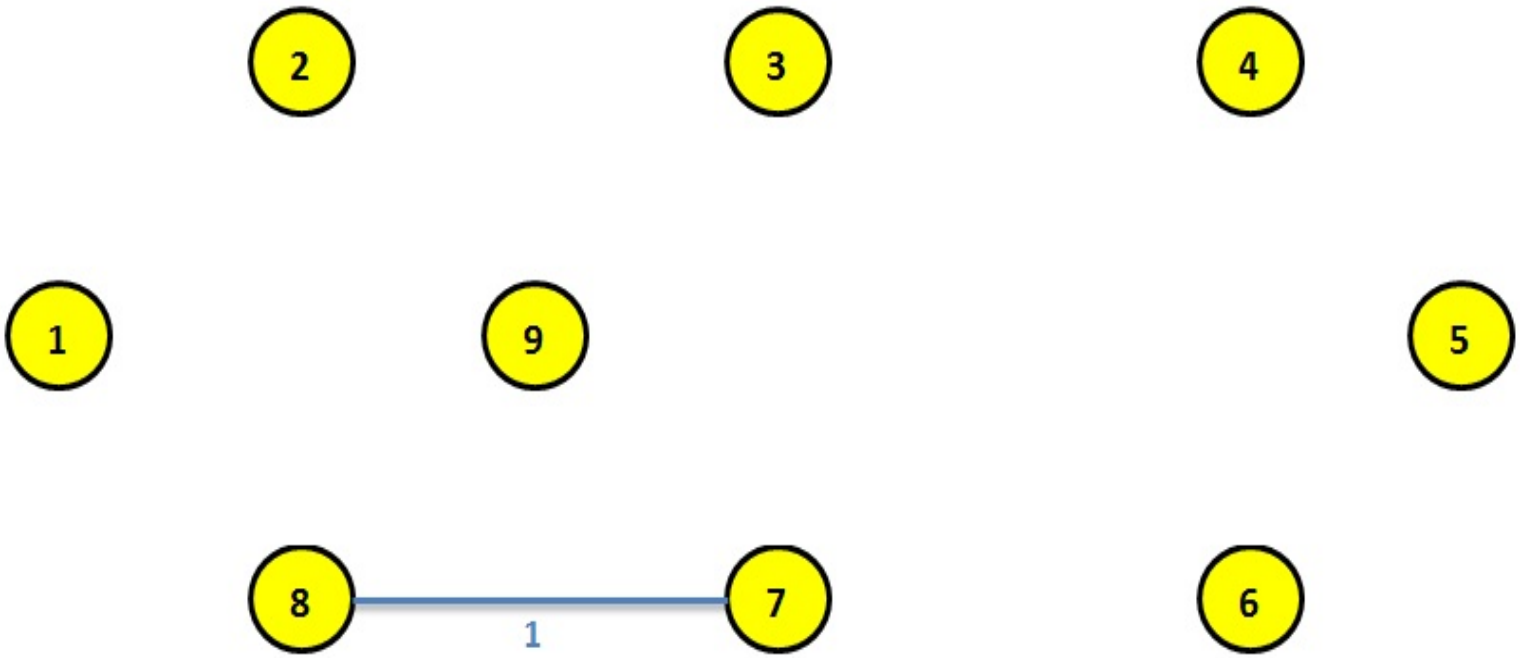
Kruskal



Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

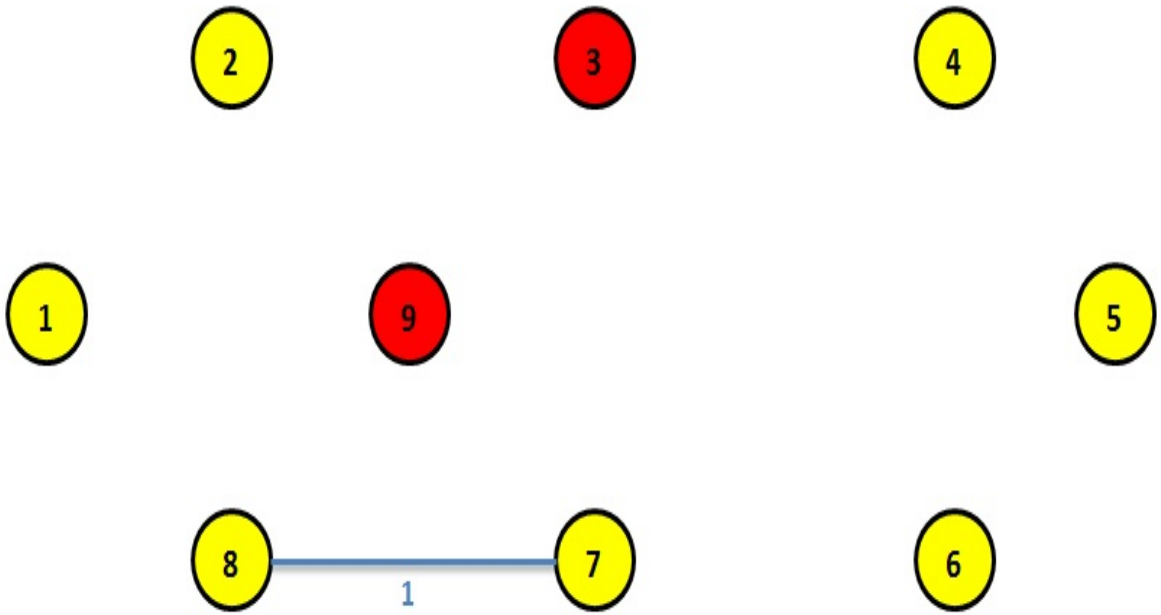
Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	3	4	5	6	7	8	9

Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	3	4	5	6	7	7	9

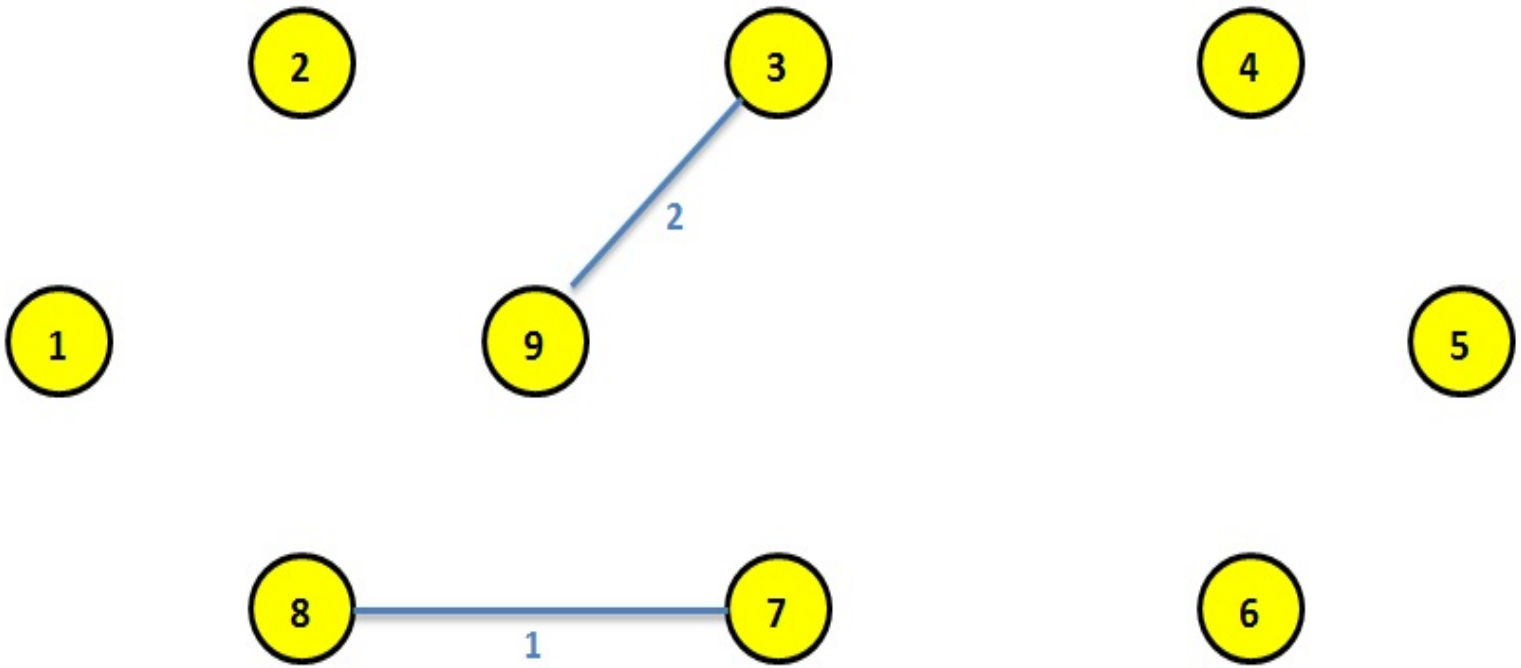
Kruskal



Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	**2**
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

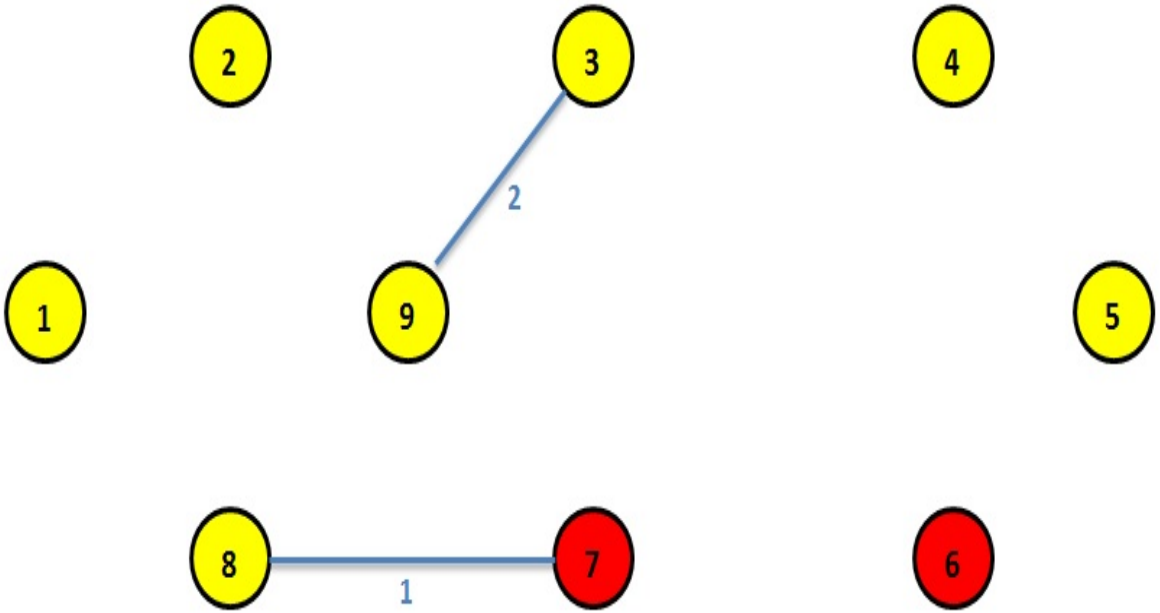
Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	3	4	5	6	7	7	9

Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	9	4	5	6	7	7	9

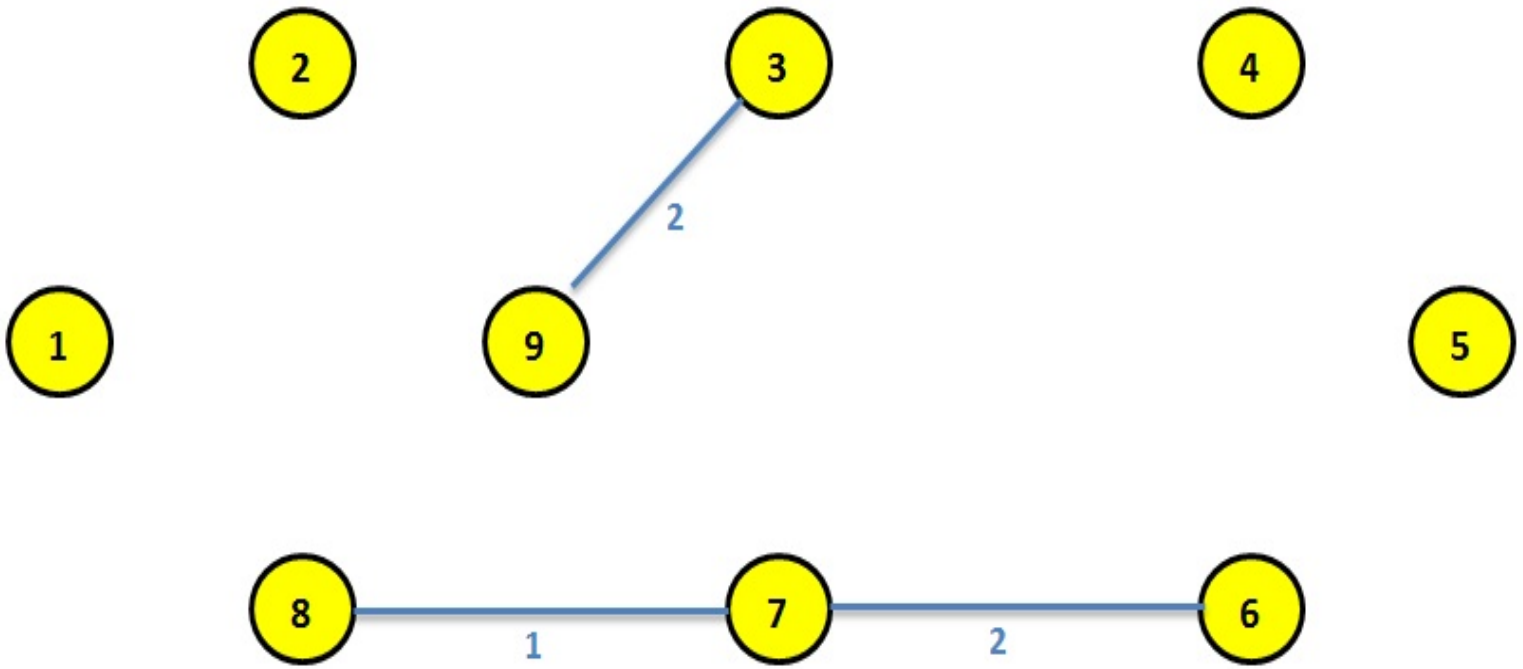
Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	9	4	5	6	7	7	9

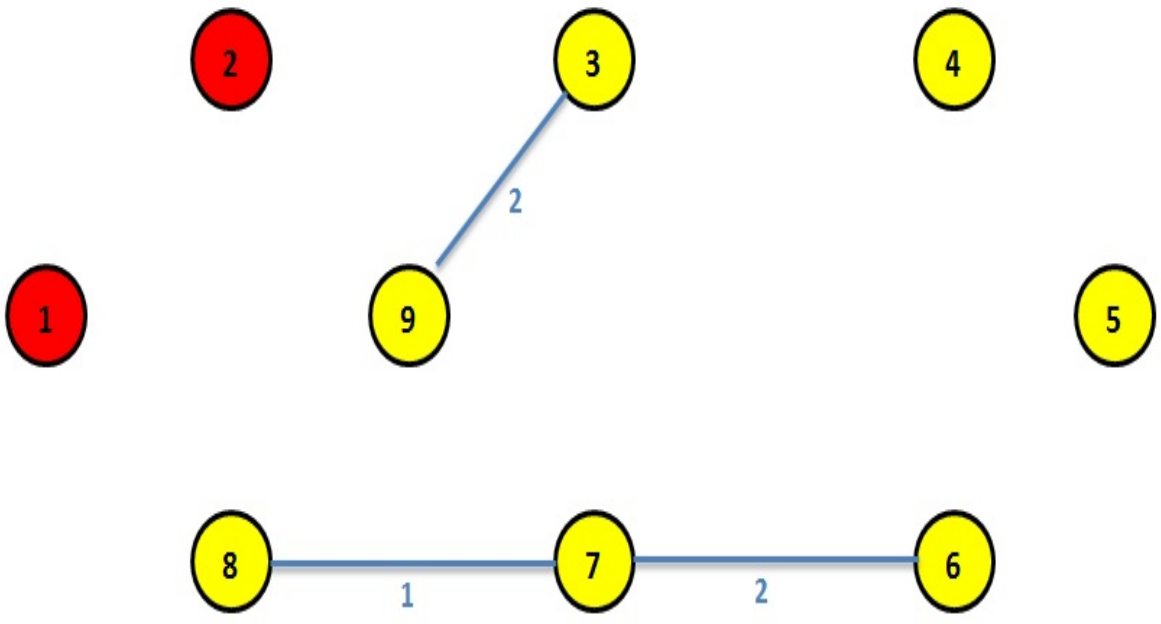
Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	**2**
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	9	4	5	7	7	7	9

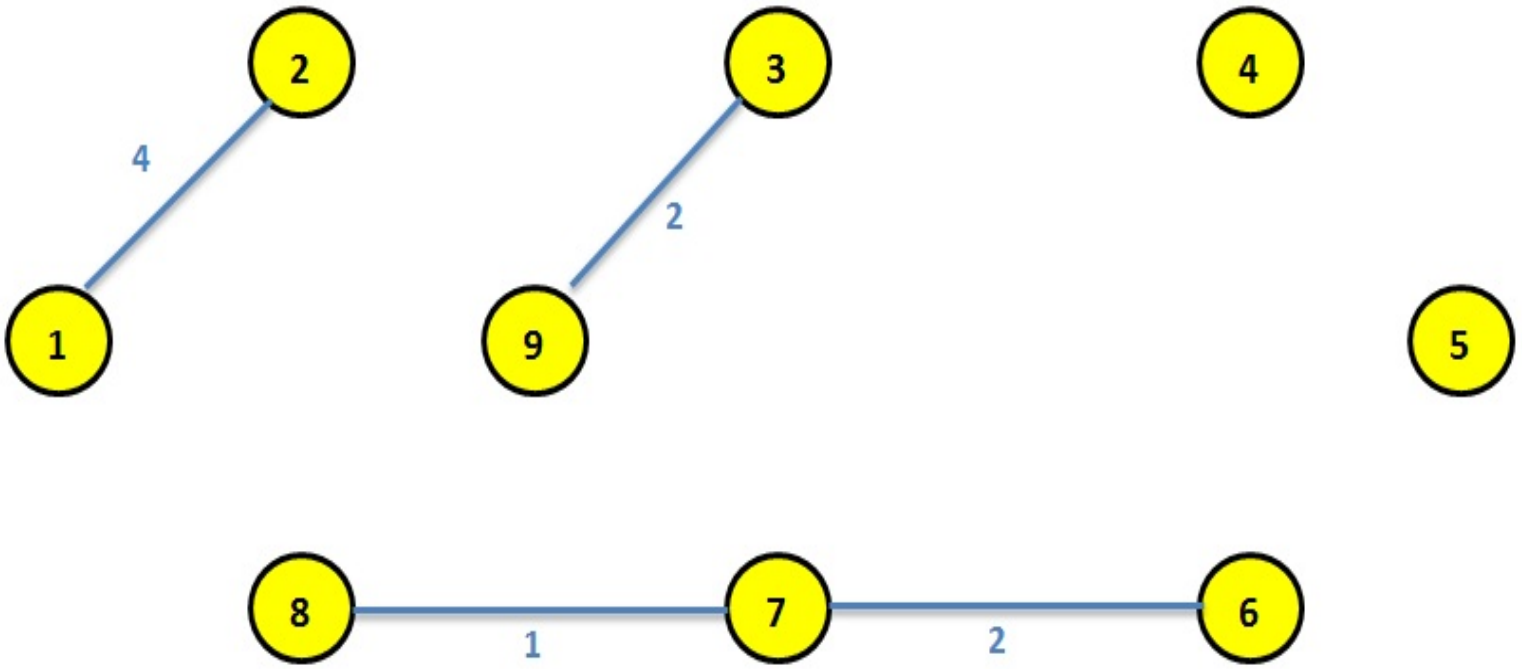
Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	9	4	5	7	7	7	9

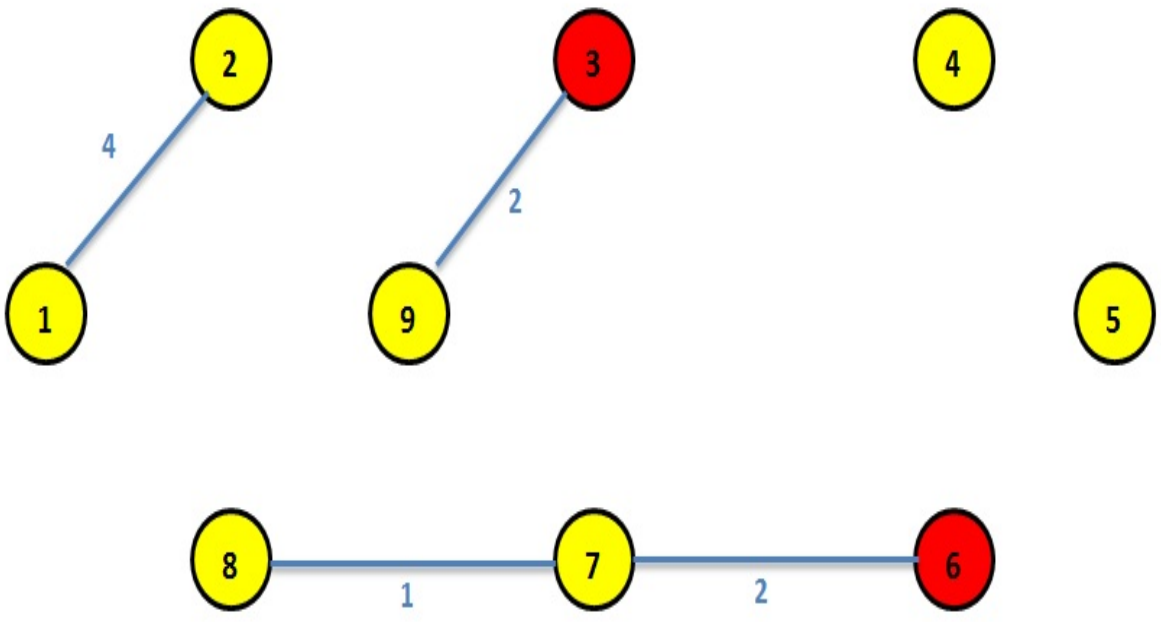
Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	**4**
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	9	4	5	7	7	7	9

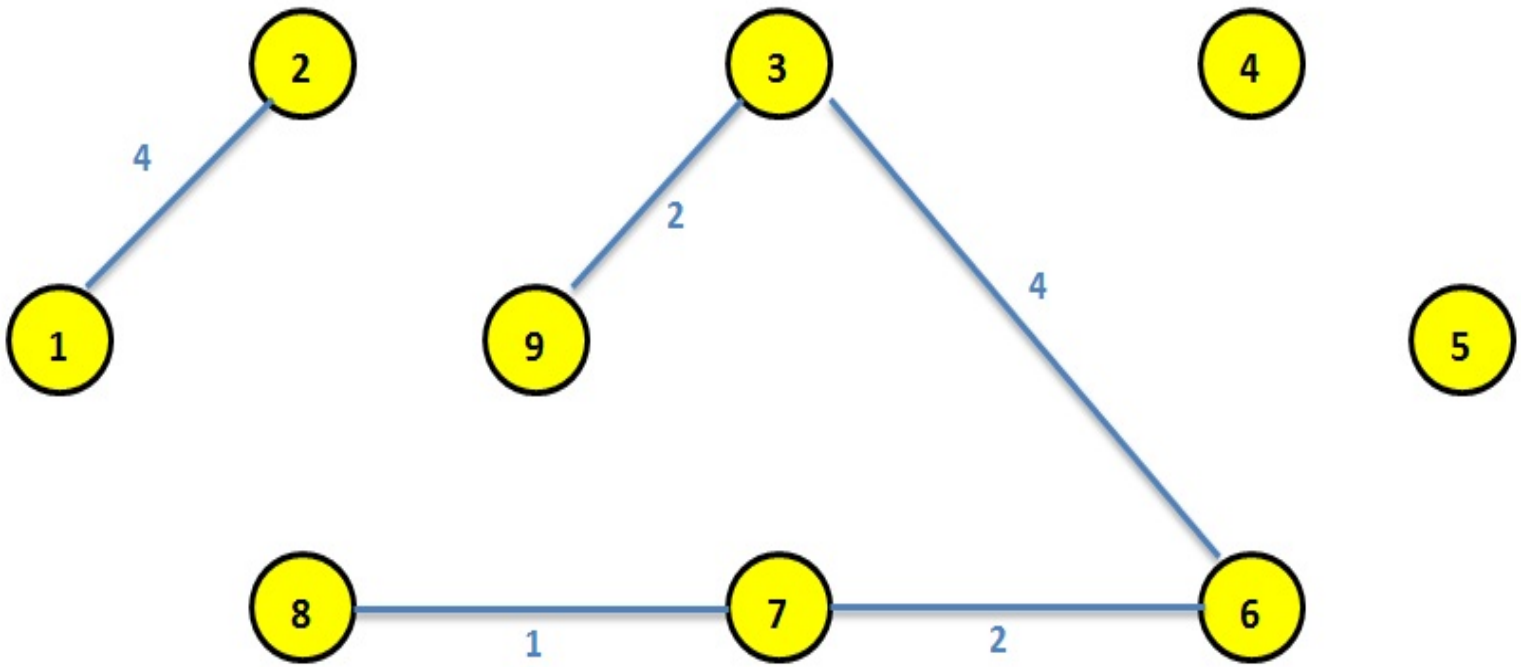
Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	9	4	5	7	7	7	9

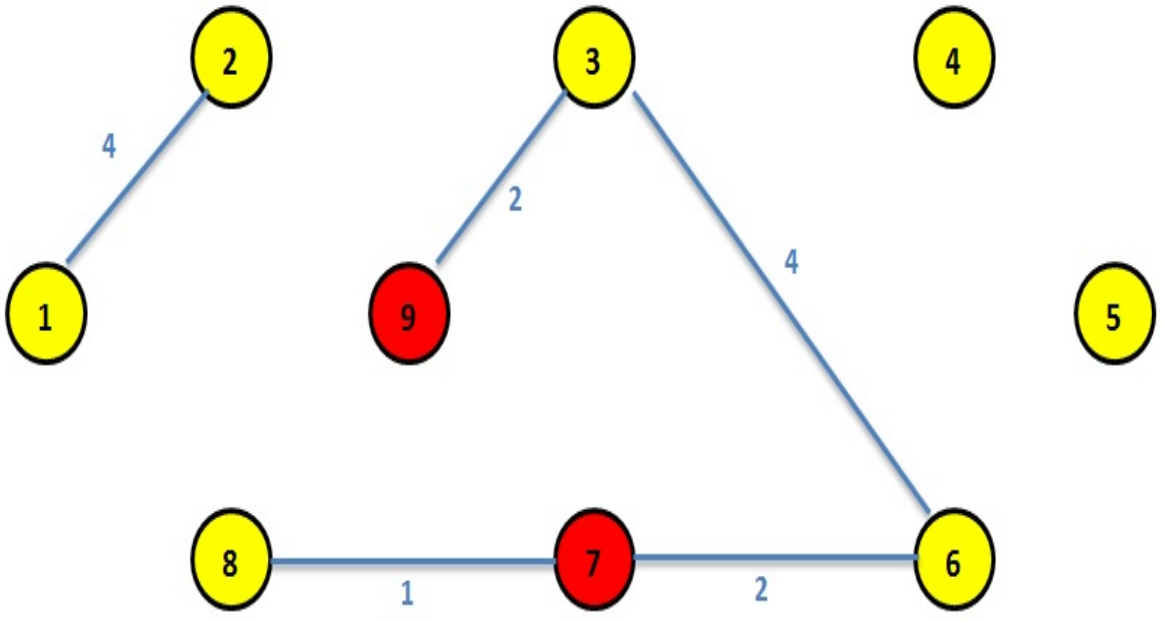
Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	**4**
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	7	4	5	7	7	7	7

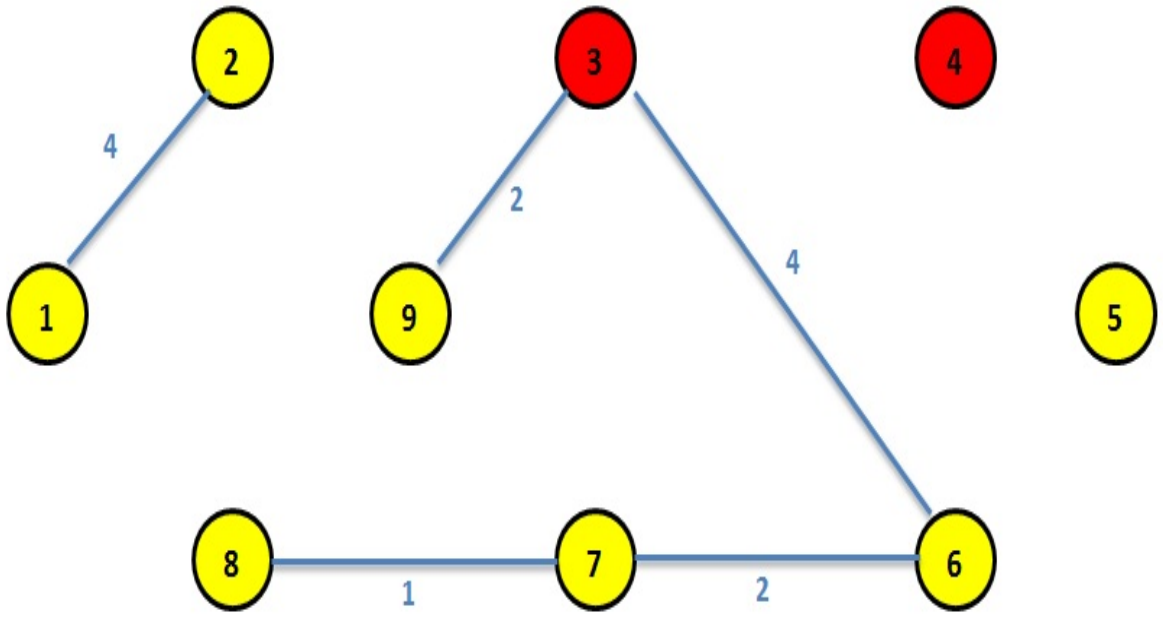
Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	7	4	5	7	7	7	7

Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	**6**
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

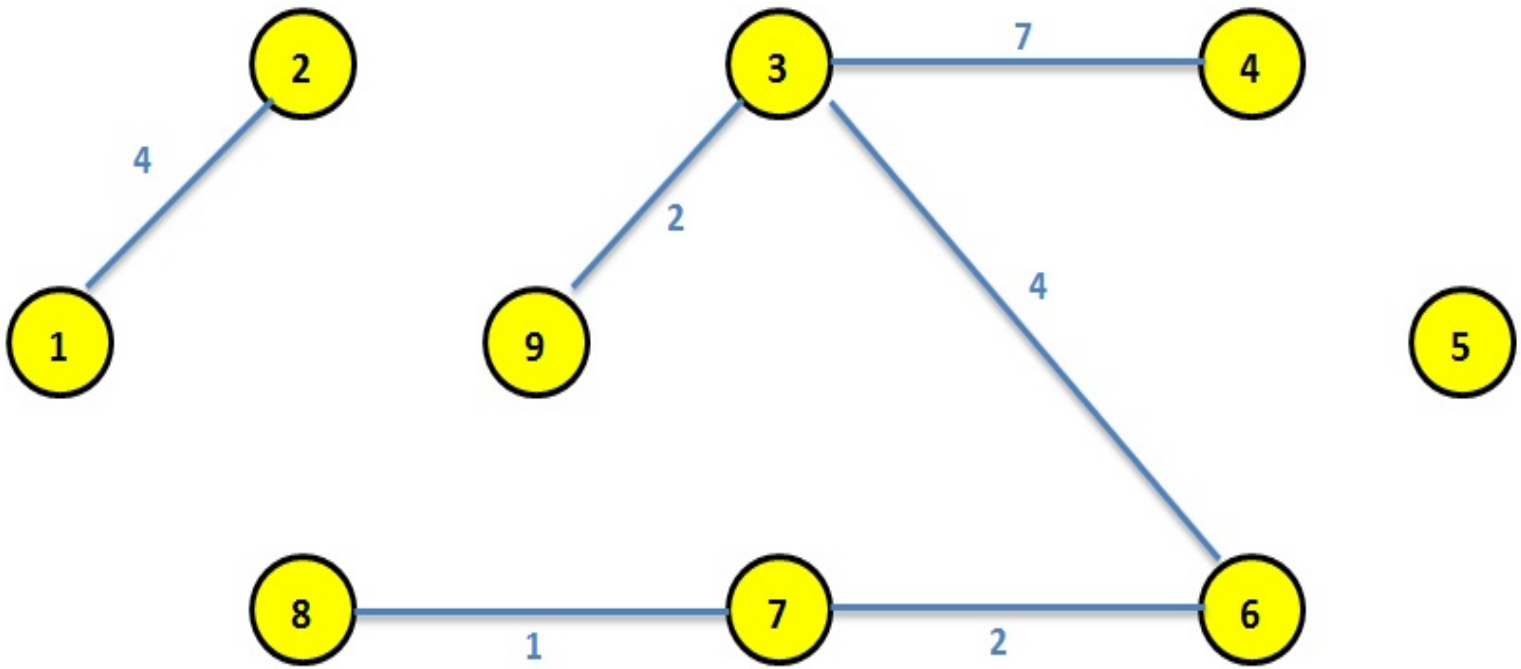
Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	7	4	5	7	7	7	7

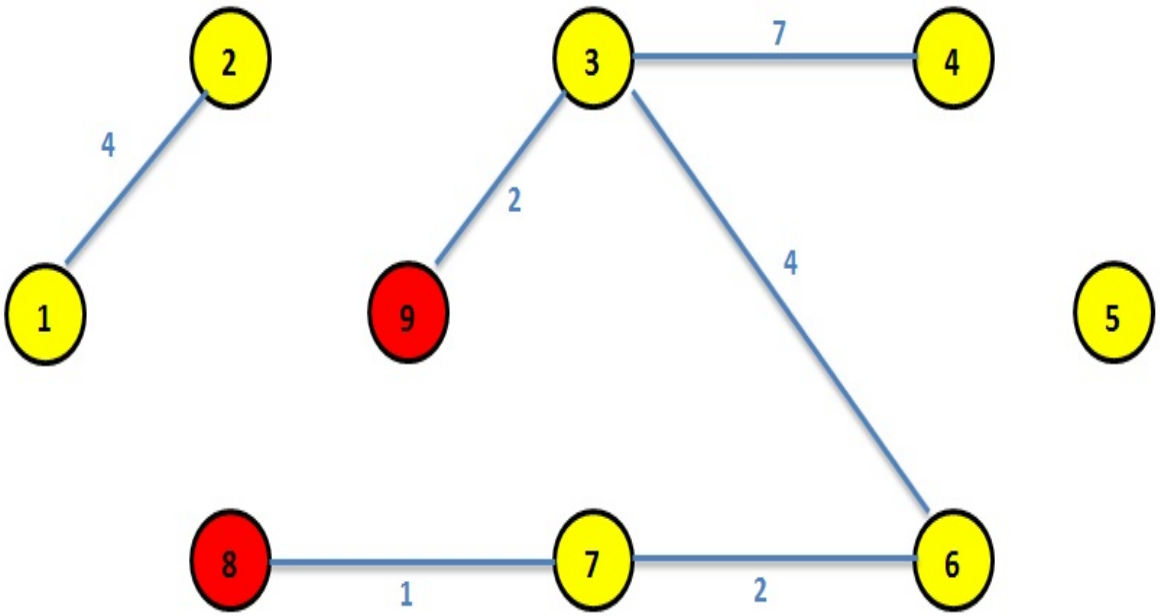
Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	**7**
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	7	7	5	7	7	7	7

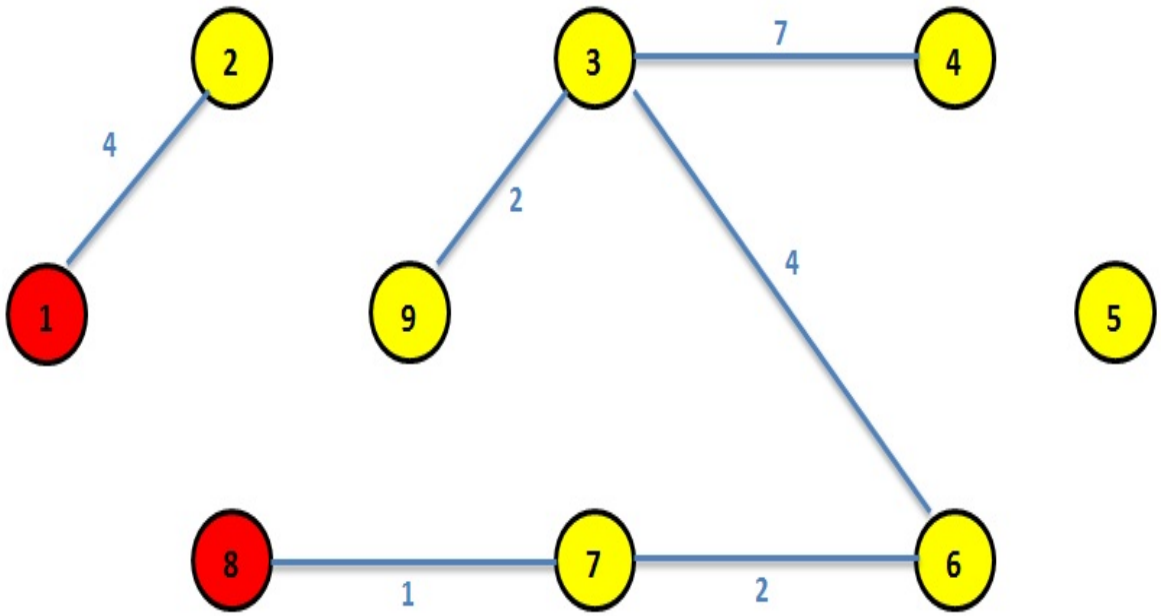
Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	7	7	5	7	7	7	7

Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	**7**
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

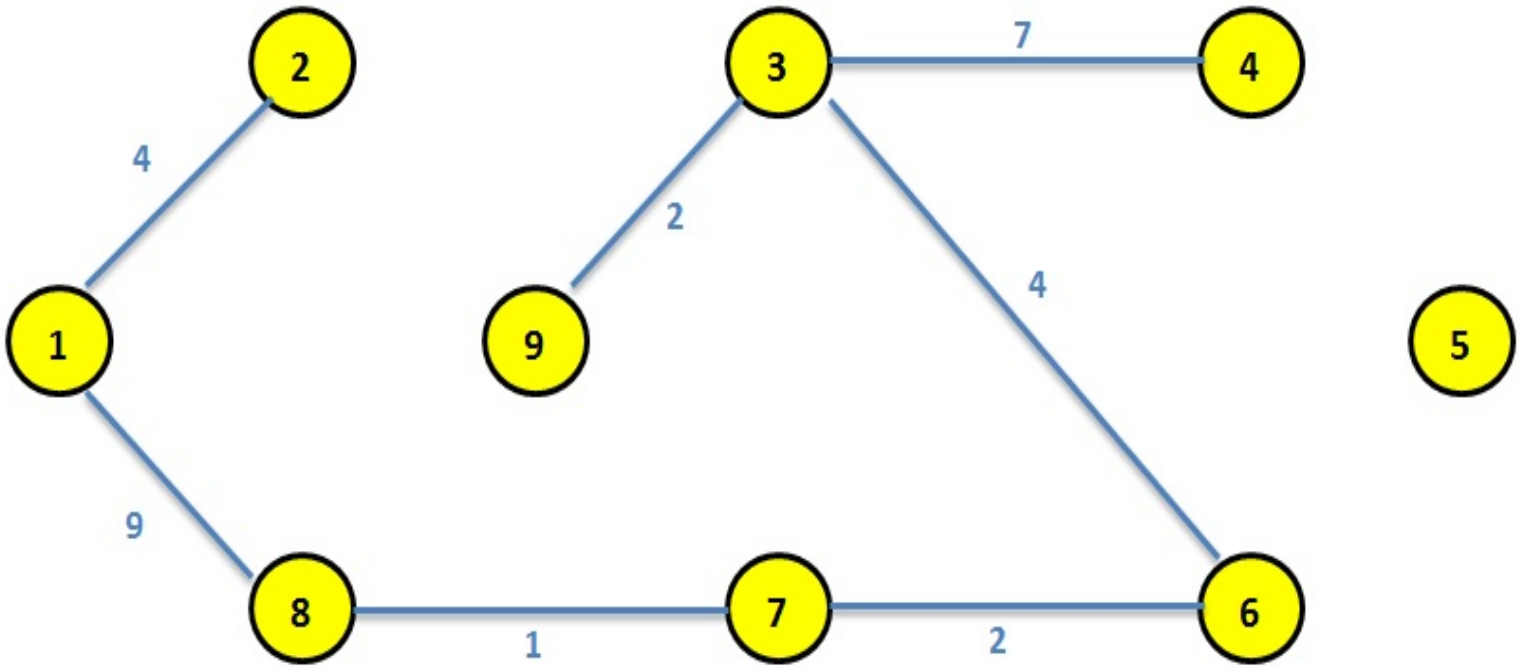
Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	7	7	5	7	7	7	7

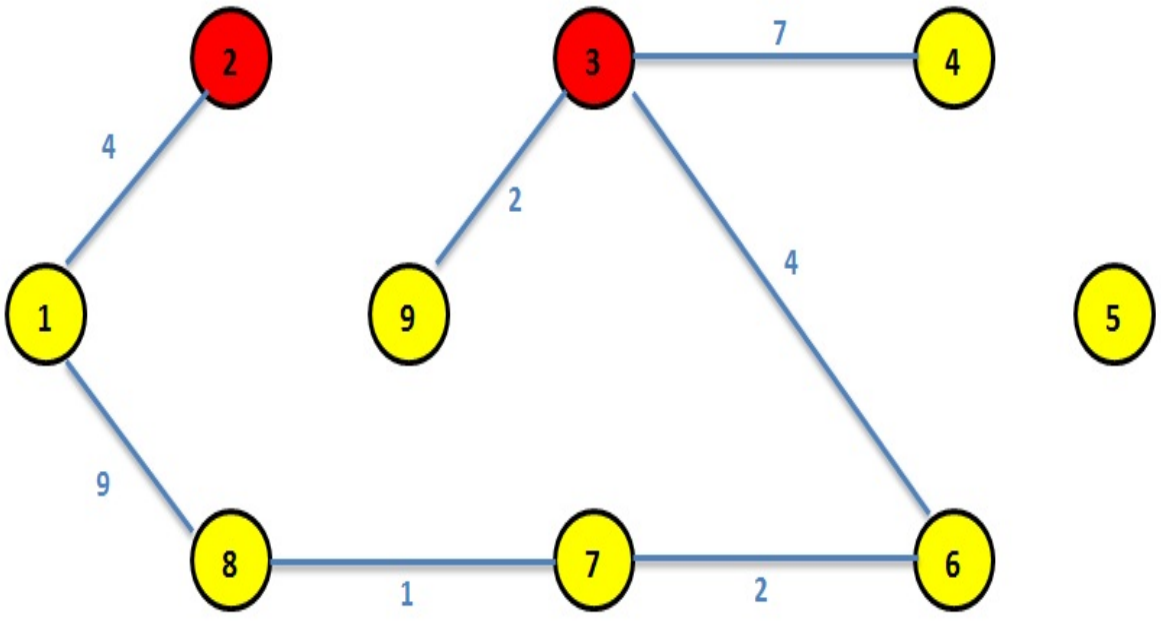
Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	**9**
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	7	7	7	7	5	7	7	7	7

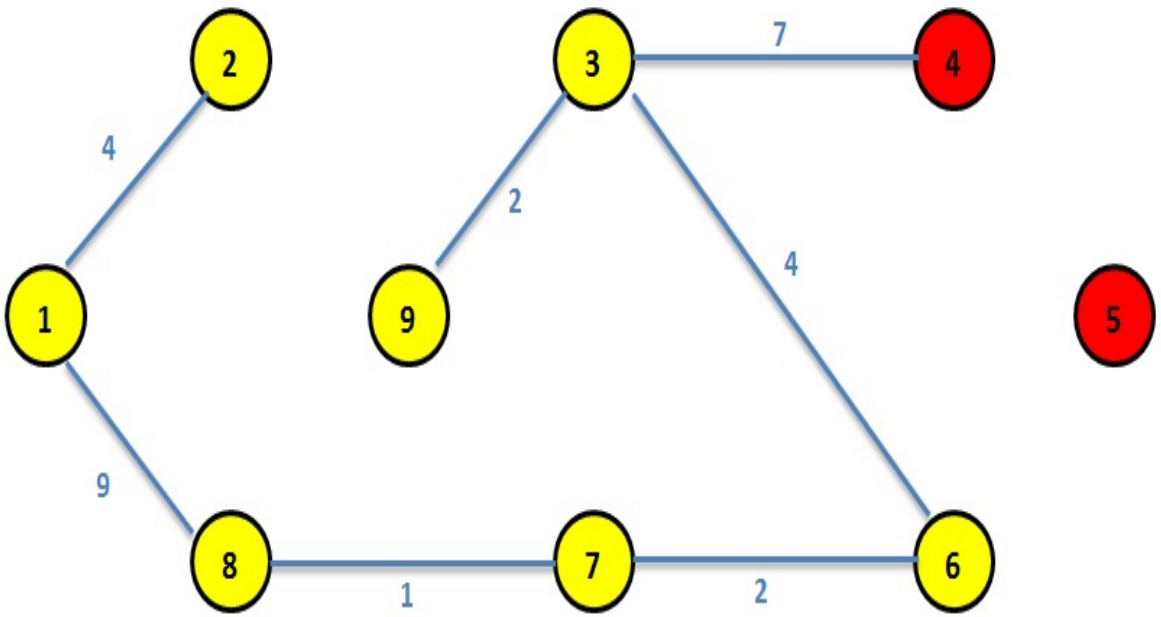
Kruskal



Vértices	1	**2**	**3**	4	5	6	7	**8**	9
Raíz	7	**7**	**7**	7	5	7	7	**7**	7

Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	**9**
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

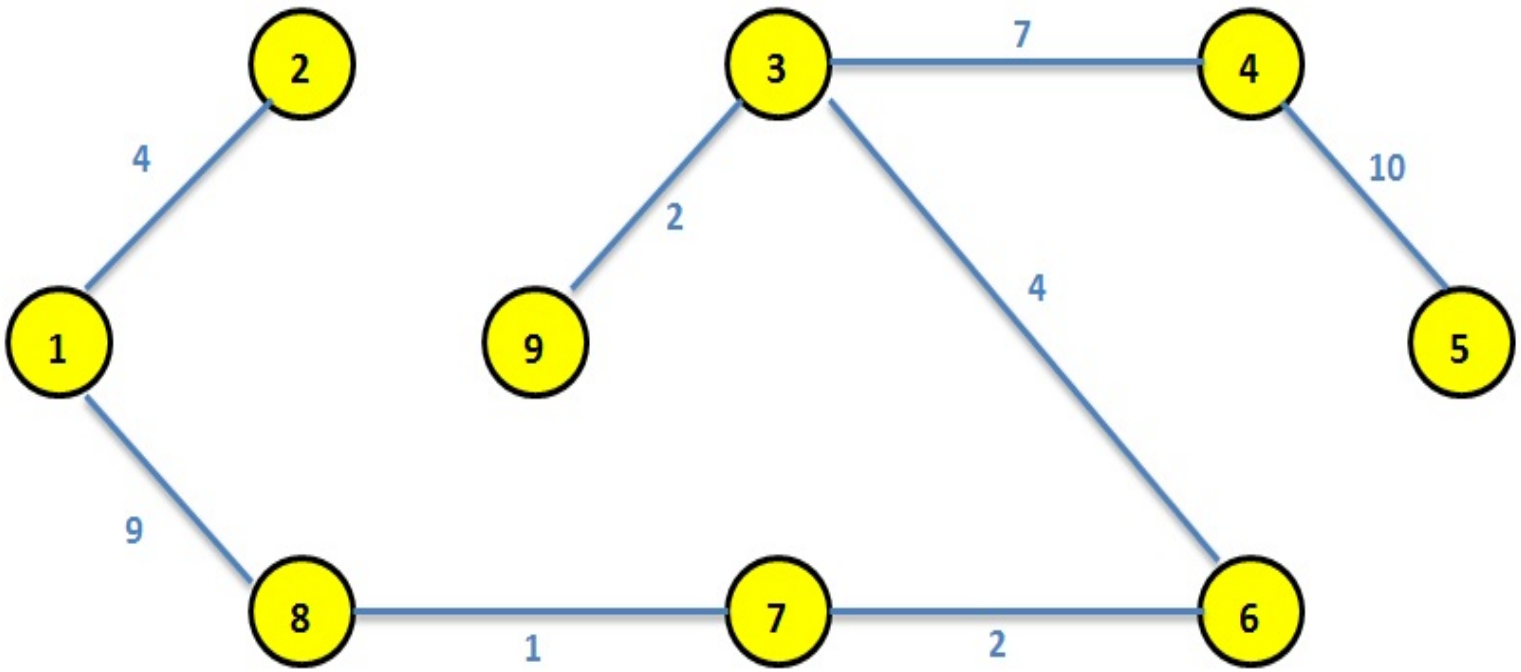
Kruskal



Vértices	1	2	3	4	5	6	7	8	9
Raíz	7	7	7	7	5	7	7	7	7

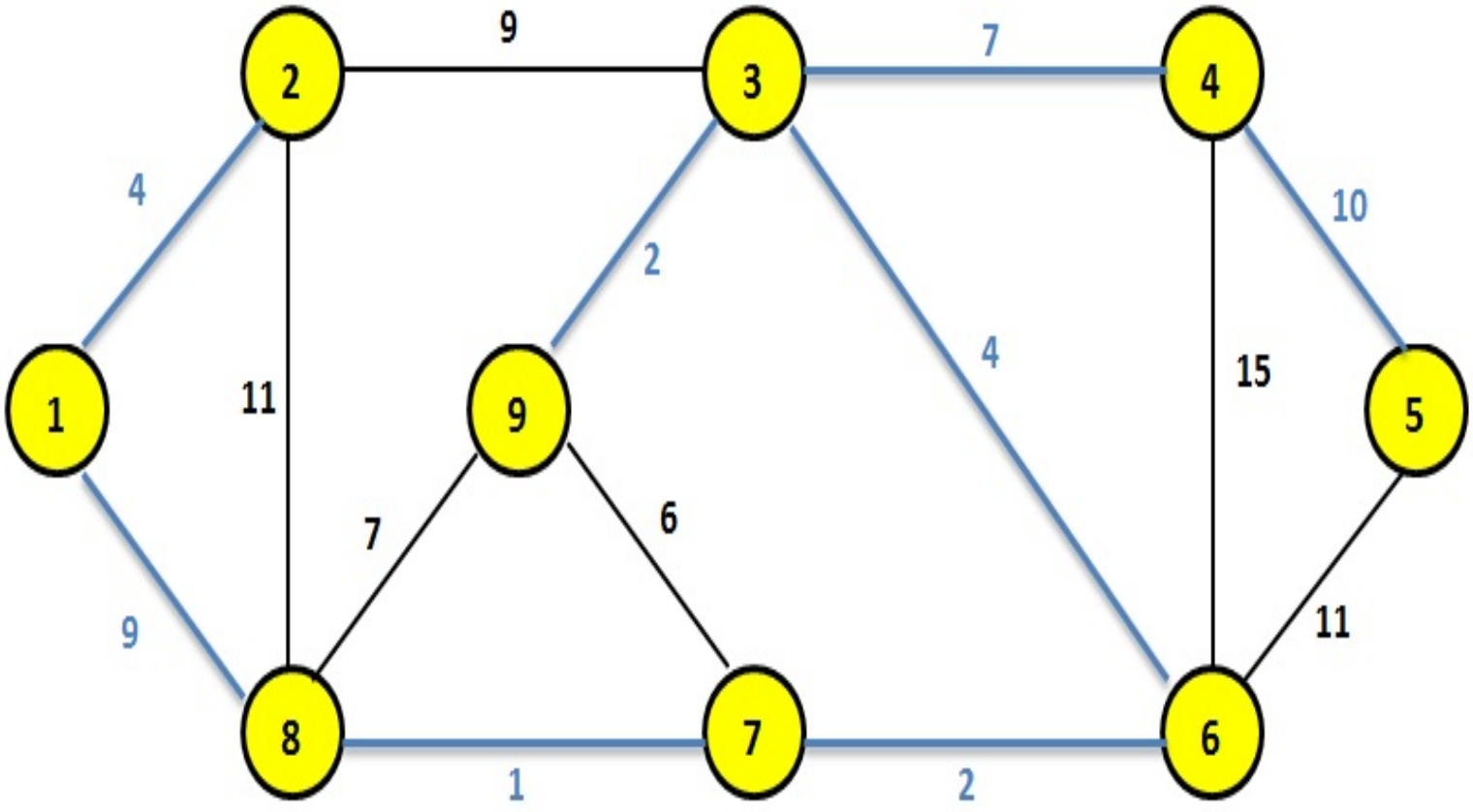
Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	**10**
2 – 8	11
5 – 6	11
4 – 6	15

Kruskal



Vértices	1	2	3	4	5	6	7	**8**	9
Raíz	7	7	7	7	7	7	7	**7**	7

Kruskal



Implementación

```
struct Edge{
    int origen;     //Vertice origen
    int destino;    //Vertice destino
    int peso;       //Peso entre el vertice origen y destino
    //Comparador por peso
    bool operator<( const Edge &e ) const {
        return peso < e.peso;
    }
}arista[ MAX ];      //Arreglo de aristas para el uso en kruskal

void Kruskal(){
    int origen , destino , peso;
    int total = 0;          //Peso total del MST

    MakeSet( V );           //Inicializamos cada componente
    sort( arista , arista + E );    //Ordenamos las aristas

    for( int i = 0 ; i < E ; ++i ){
        origen = arista[ i ].origen;
        destino = arista[ i ].destino;
        peso = arista[ i ].peso;
        //Verificamos si estan o no en la misma componente conexa
        if( !sameComponent( origen , destino ) ){  //Evito ciclos
            total += peso;              //Incremento el peso total del MST
            Union( origen , destino );  //Union de ambas componentes
        }
    }

    printf("El costo minimo de todas las aristas del MST es : %d\n" , total );
}
```

Verificación de MST

- Para que sea un MST válido el número de aristas debe ser igual al número de vértices – 1.
- Esto se cumple debido a que el MST debe poseer todos los vértices del grafo ingresado y además no deben existir ciclos.