Union-find : (Disjoint set union)

If we have a set of N elements which are partitioned into further subsets, and if we have to keep track of connectivity of each element in a particular subset or connectivity of subsets with each other, we use Union-find.

Union(A,B) - Connect two elements A and B
Find (A,B) - find, if there is any path connecting two elements A and B.

Example:

Arr	0	1	2	3	4	5	6	7	8	9
	0	1	2	3	4	5	6	7	8	9

y x
Union(2,1)

Arr	0	1	1	3	4	5	6	7	8	9
	0	1	2	3	4	5	6	7	8	9

Union (4,3)
Union (8,4)
Union (9,3)
Union (6,5)

							5		3	3
Arr	0	1	1	3	3	5	~~6~~	7	~~8~~	~~9~~
	0	1	2	3	4	5	6	7	8	9

We now have 5 subsets.

{0}

{1,2}

{3,4,8,9}

{5, 6}

{7}

Each subset is a connected component.

find(0,7) → false

find (8,9) → true

We have,

Arr									
0	1	1	3	3	5	5	7	3	3
0	1	2	3	4	5	6	7	8	9

Union(5,2)

Arr									
0	1	1	3	3	1	1	7	3	3
0	1	2	3	4	5	6	7	8	9

<u>Code Snippets:</u>

```
int find (int arr[], int a, int b)
{
    if (arr[a] = =arr [b])
        return 1;
    else
        return 0;
}
```

PAGE NO. :
DATE : / /

```
void union( int arr[],int n, int a, int b)
{
    int temp = arr[a];
    inti;
    for (i=0; i<n; i++)
    {
        if(arr[i]==temp)
            arr[i]=arr[b];
    }
}
```

* Kruskal's Algorithm *

- Used to find the minimum spanning tree

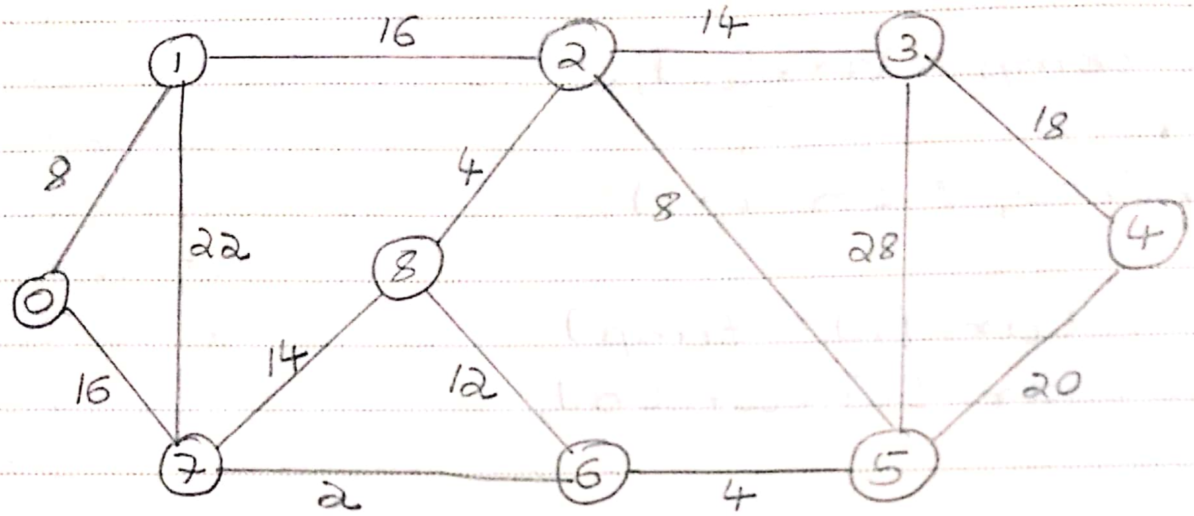
<u>Method</u>:

- Sort all the edges
- Keep adding edges one by one until
    - There is no cycle
    - all the nodes are connected

Every spanigng tree of n vertices will have n-1 edges.

<u>Alternate</u>: Prim's.

Example:

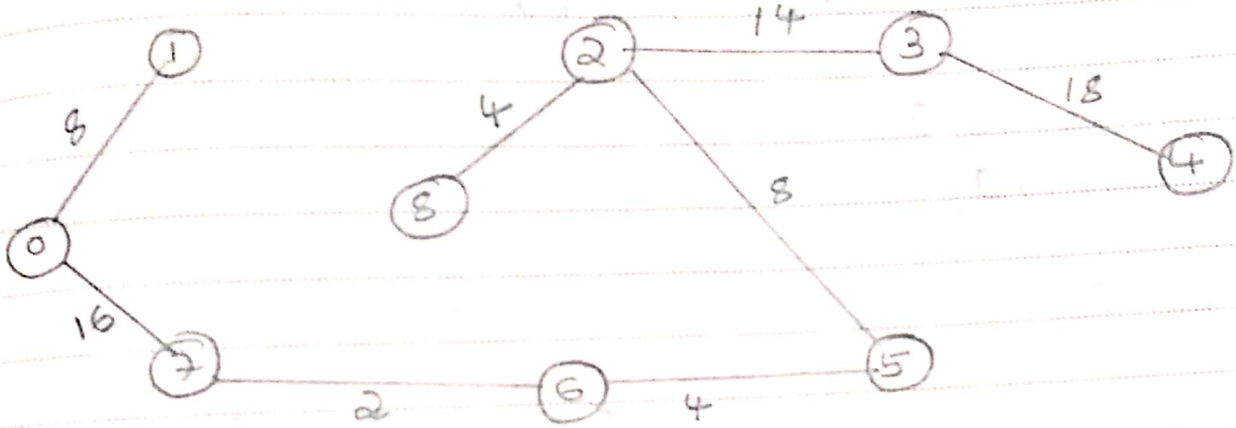


Sorted Edges: (6,7) → 2 (2,8) → 4 (5,6) → 4
(0,1) → 8 (2,5) → 8 (6,8) → 12 (2,3) → 14
(7,8) → 14 (0,7) → 16 (1,2) → 16 (3,4) → 18
(4,5) → 20 (1,7) → 22 (3,5) → 28

Steps	(u,v)	i = find(u) j = find(v)	o/p (u,v)	Union(i,j) arr 0	1	2	3	4	5	6	7	8
init				0	1	2	3	4	5	6	7	8
1	(6,7)	6, 7	(6,7)	0	1	2	3	4	5	7	7	8
2	(2,8)	2, 8	(2,8)	0	1	8	3	4	5	7	7	8
3	(5,6)	5, 7	(5,6)	0	1	8	3	4	7	7	7	8
4	(0,1)	0, 1	(0,1)	1	1	8	3	4	7	7	7	8
5	(2,5)	8 7	(2,5)	1	1	7	3	4	7	7	7	7
6	(6,8)	7 7	discard									
7	(2,3)	7 3	(2,3)	1	1	3	3	4	3	3	3	3
8	(7,8)	3 3	discard									
9	(0,7)	1 3	(0,7)	3	3	3	3	4	3	3	3	3
10	(1,2)	3 3	discard									
11	(3,4)	3 4		4	4	4	4	4	4	4	4	4

DATE : / /

Spanning Tree:



Minimum Cost = 8+16+2+4+8+4+14+18
= 74.

Union-find : Root Method

Arr	~~0~~ 2	0 ~~1~~	2	3	4	5
	0	1	2	3	4	5

copy

Union (0, 2)

Union(1, 0)

root of 0

root of 1

Union (1, 4)

Arr	2	0	4	~~3~~ 4	4	5
	0	1	2	3	4	5

Union (3, 4)

Scanned with CamScanner

DATE : / /

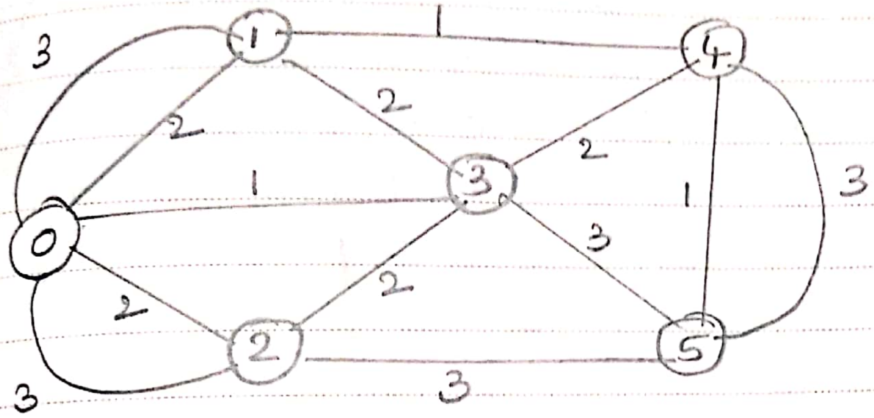
Code Snippets :

```
int root (int arr[], int i)
{
    while (arr [i] != i)
        i = arr [i];

    return i;
}

int find (int u, int v}, int arr [])
{
    if((root (arr, u)) == (root (arr, v)))
        return 1;
    else
        return 0;
}

int union ( int arr[], int u, int v)
{
    int rootu = root (arr, u)
    int rootv = root (arr, v)
    arr [rootu] = rootv;
}
```

Scanned with CamScanner

Kruskals Example:

Compute MAX Spanning Tree.



Sorted Edges: (descending)

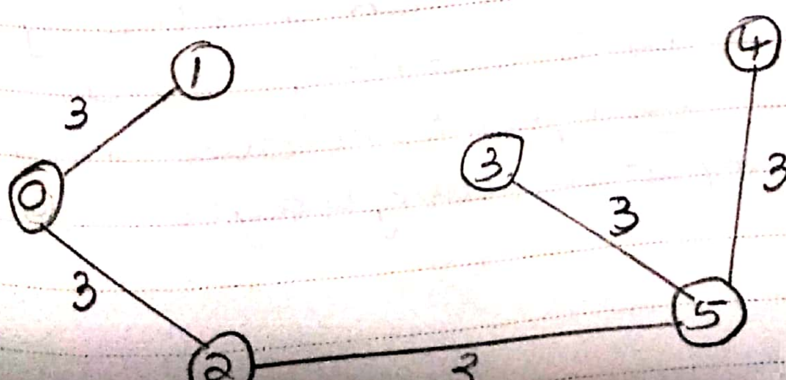
(2,5) → 3

(0,1) → 3   (0,2) → 3   (3,5) → 3   (4,5) → 3

(0,1) → 2   (0,2) → 2   (1,3) → 2   (2,3) → 2

(3,4) → 2   (0,3) → 1   (1,4) → 1   (4,5) → 1

Steps	(u,v)	i = find(u), j = find(v)	Output	Union(i,j): 0 1 2 3 4 5
init				0 1 2 3 4 5
1	(0,1)	0 1	(0,1)	1 1 2 3 4 5
2	(0,2)	1 2	(0,2)	2 2 2 3 4 5
3	(2,5)	2 5	(2,5)	5 5 5 3 4 5
4	(3,5)	3 5	(3,5)	5 5 5 5 4 5
5	(4,5)	4 5	(4,5)	5 5 5 5 5 5



Tree cost = 15.