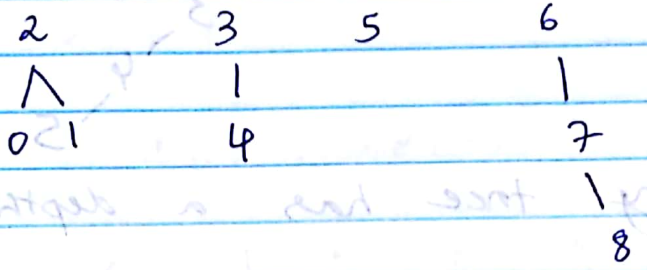
Assignment - 8

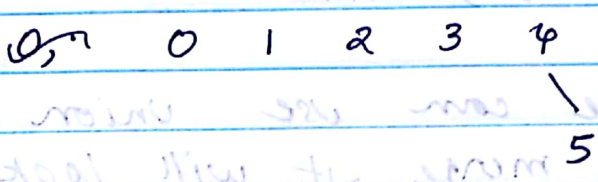
1.

2	2	-1	-1	3	-1	-1	6	7
0	1	2	3	4	5	6	7	8

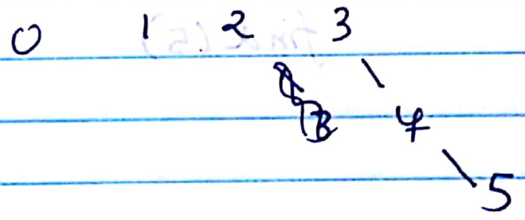


2. 0, 1, 2, 3, 4, 5

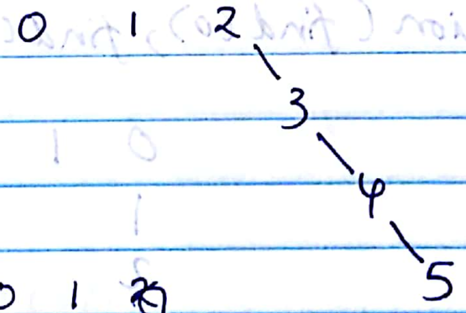
Union (4,5)



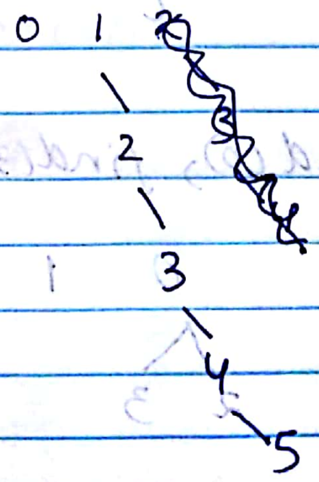
Union (3,4)



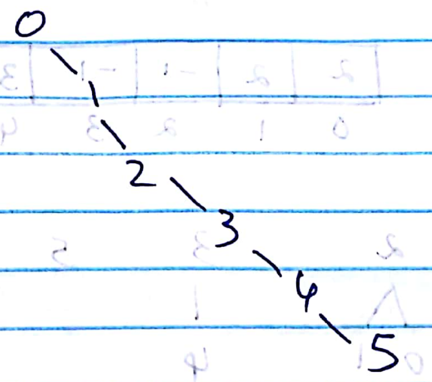
Union (2,3)



Union (1,2)



Union (0,1)

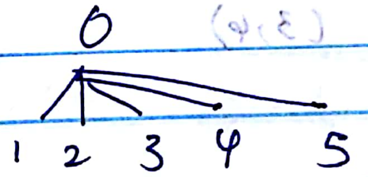


The resulting tree has a depth of 5 i.e (N-1).

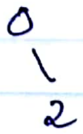
Although '5' can be accessed at O(1) time, find(5) in the resulting tree will take O(N) time.

We can use union function to make a merge, it will look as following:

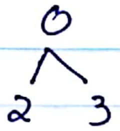
find(5)



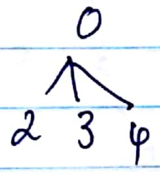
3 union (find(0), find(2))



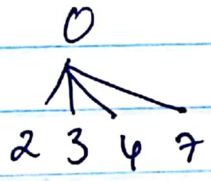
union (find(0), find(3))



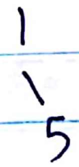
union (find(0), find(4))



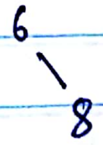
union (find(0), find(7))



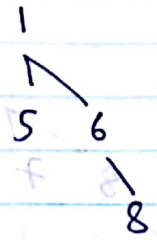
union(find(1), find(5))



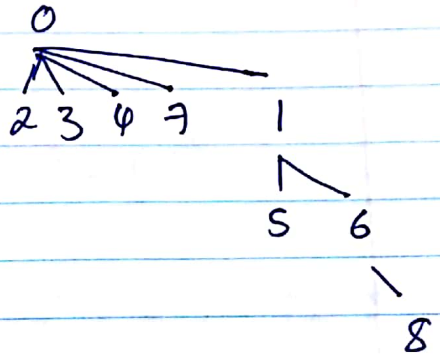
union(find(6), find(8))



union(find(5), find(8))



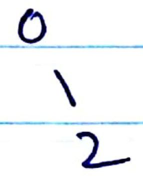
union (find(7), find(8))



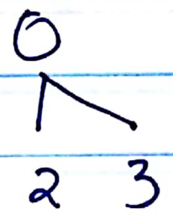
4.

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

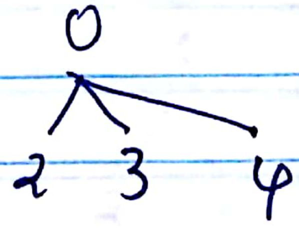
5. Union (find (0), find(2))



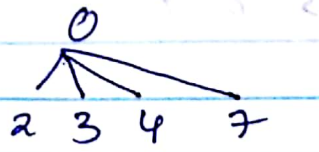
Union (find(0), find(3))



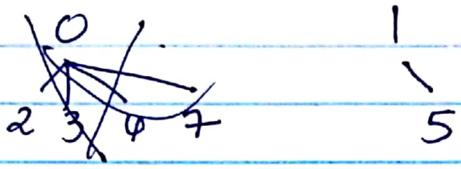
Union (find(0), find(4))



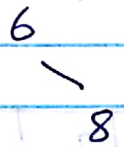
union (find(0), find(7))



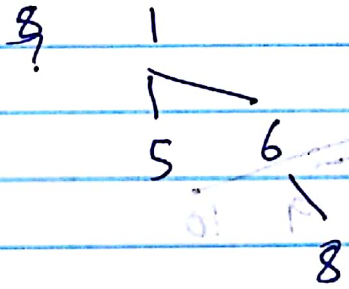
union (find(1), find(5))



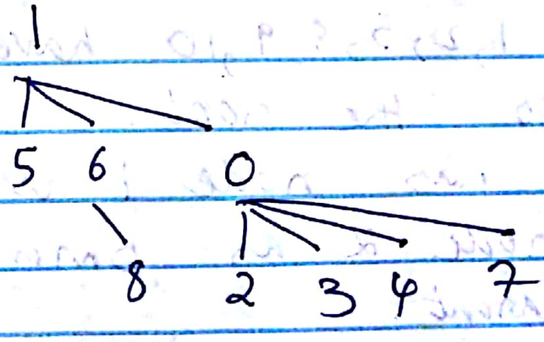
union (find(6), find(8))



union (find(5), find(8))



union(find(7), find(8))

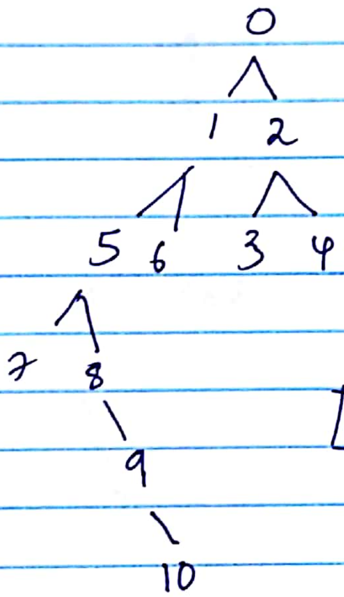


6.

1	-3	0	0	0	1	1	0	6
0	1	2	3	4	5	6	7	8

7.

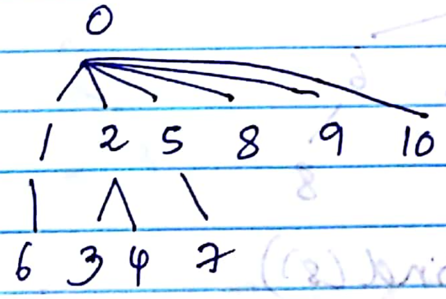
-1	0	0	2	2	1	1	5	5	8	9
0	1	2	3	4	5	6	7	8	9	10



after find(10)

-1	0	0	2	2	0	1	5	0	0	0
0	1	2	3	4	5	6	7	8	9	10

8.



Nodes 1,2,5,8,9,10 have 0 as parent, 0 is also the root.
Node 6 has node 1 as parent, Nodes 3,4 have node 2 as parent, node 7 has 5 as parent.