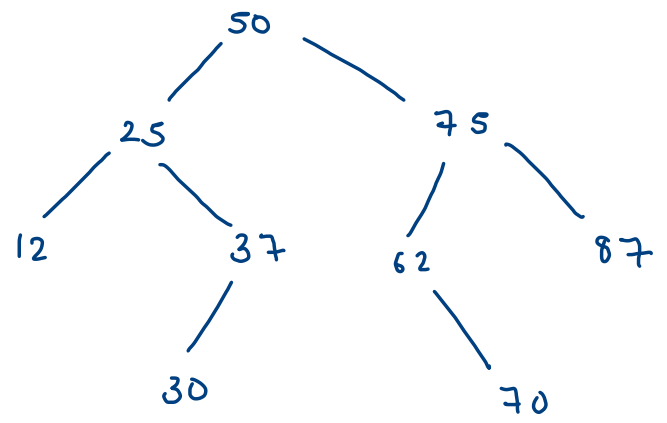
Construct Bst From Levelorder Traversal



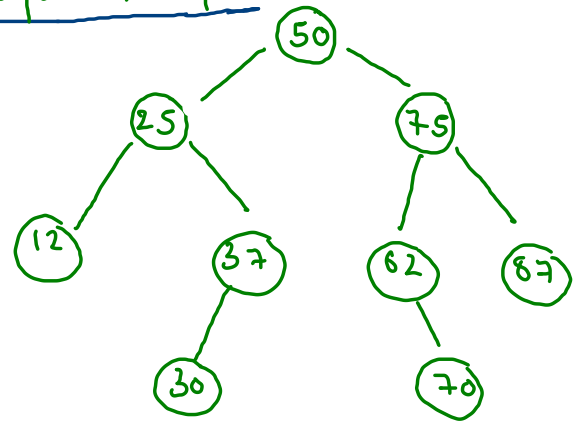
lev: 50 25 75 12 37 62 87 30 70

key: 50 25 75 12 37 62 87 30 70

i

-∞, ∞, null	-∞, 50, 50'	50, ∞, 50'	-∞, 25, 25'	25, 50, 25'	50, 75, 75'	75, ∞, 75'	-∞, 12, 12'	12, 25, 12'

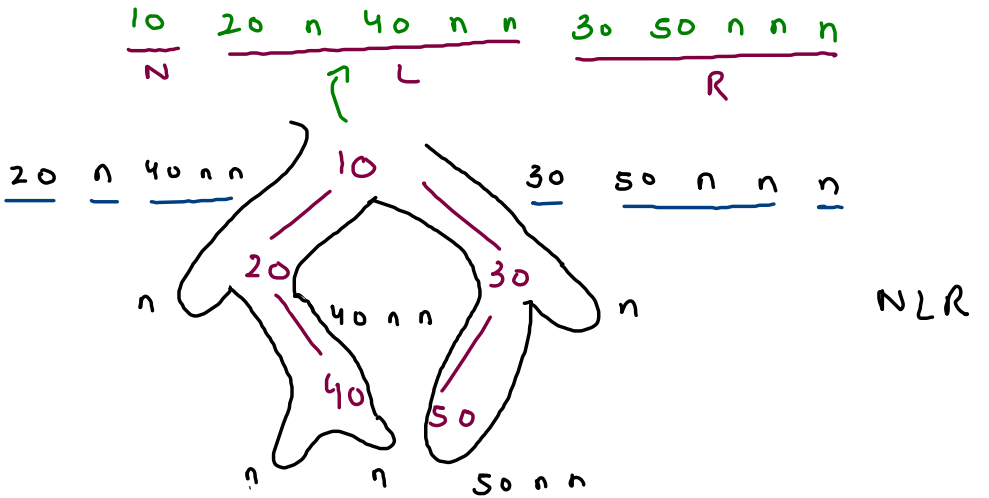
25, 37, 37'	37, 50, 37'	50, 62, 62'	62, 75, 62'	75, 87, 87'	87, ∞, 87'	30L	30R	70L	70R



Pair: lr, rr, parent

297. Serialize and Deserialize Binary Tree

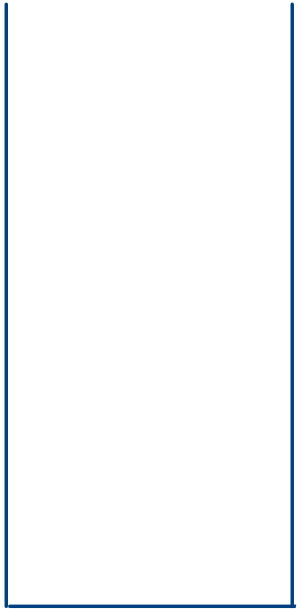
Serialize

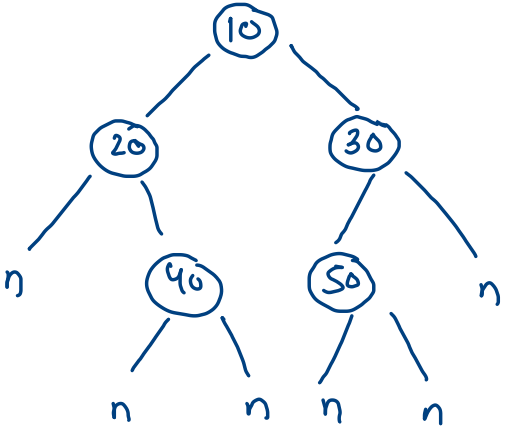


Deserialise

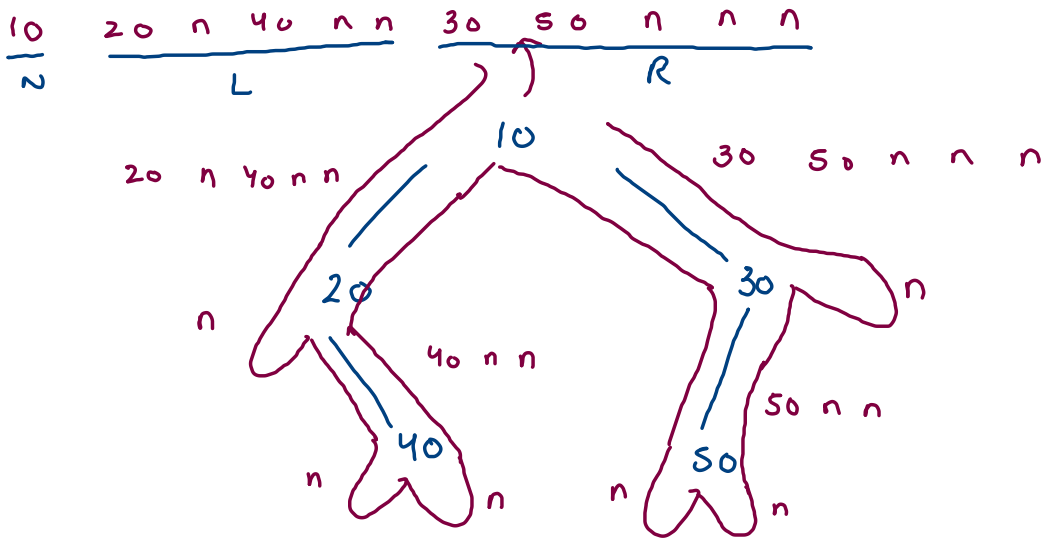
10 20 n 40 n n 30 50 n n n

i



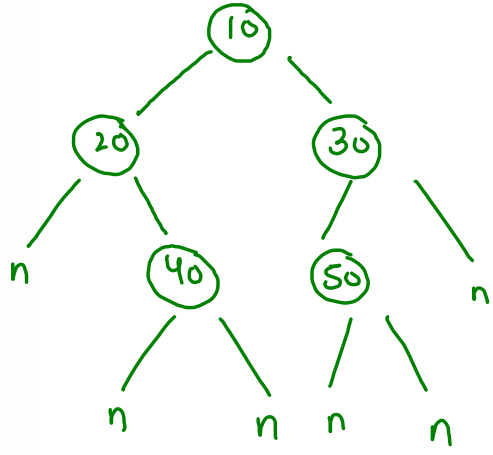


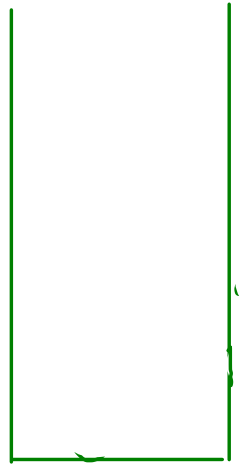
```java
// Encodes a tree to a single string.
public String serialize(TreeNode root) {
    if(root == null) {
        return "n";
    }

    String la = serialize(root.left);
    String ra = serialize(root.right);

    return root.val + " " + la + " " +ra;
}
```



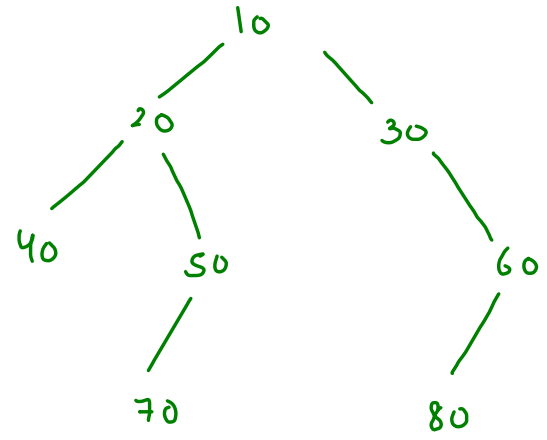
```java
int idx;
public TreeNode deserialize(String data) {
    String[]arr = data.split(" ");
    idx = 0;

    return helper(arr);
}

public TreeNode helper(String[]arr) {
    if(arr[idx].equals("n") == true) {
        idx++;
        return null;
    }
    else {
        TreeNode node = new TreeNode(Integer.parseInt(arr[idx]));
        idx++;
        node.left = helper(arr);
        node.right = helper(arr);

        return node;
    }
}
```

10 20 n 40 n n 30 50 n n n



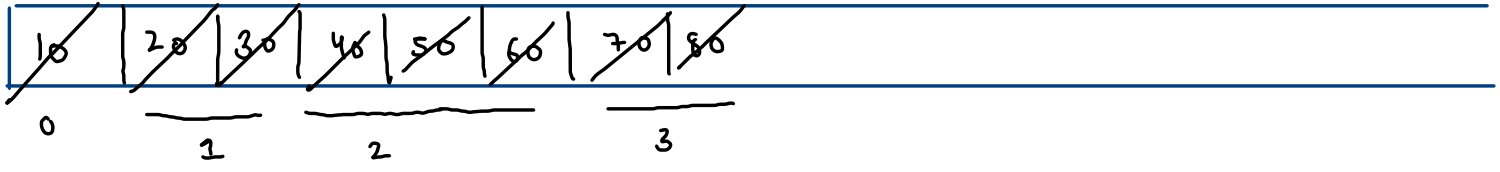


level-order (normal)

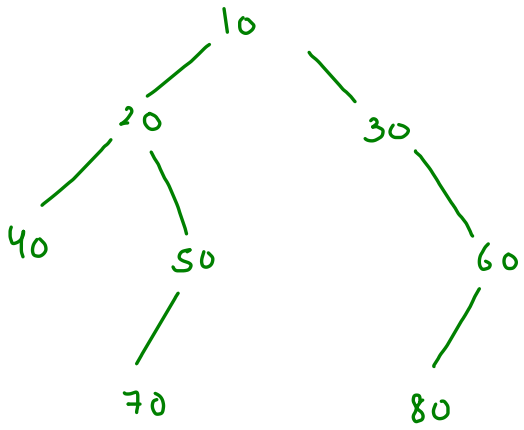


10 20 30 40 50 60 70 80

remove
work
add children



level order (linewise)



10

20 30

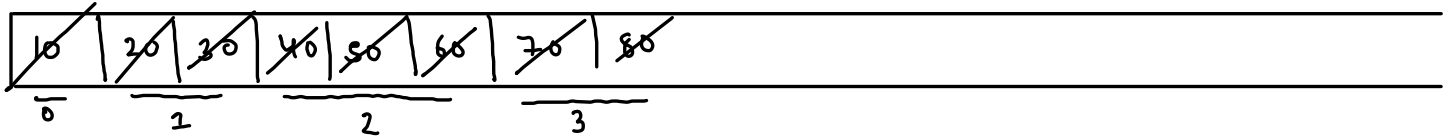
40 50 60

70 80

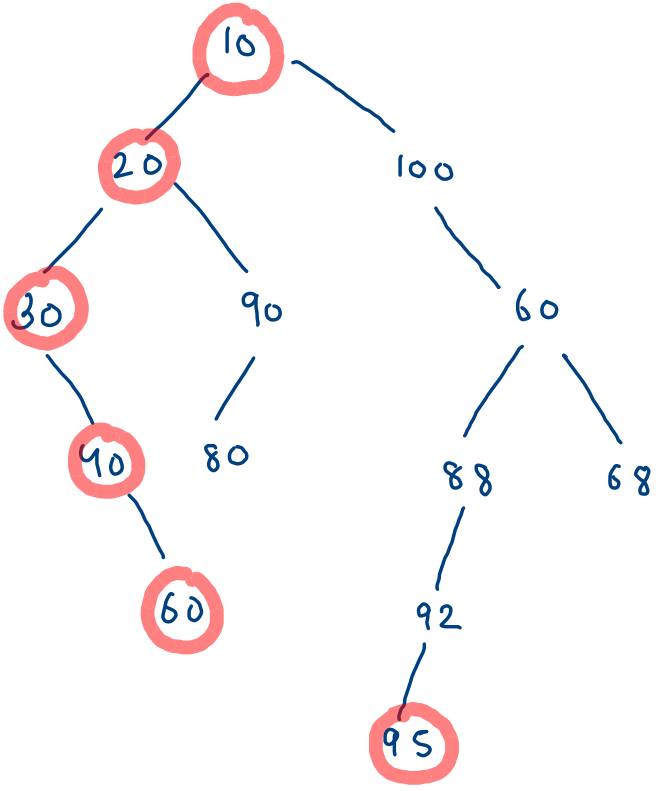
c = 2

count times { remove, work, add children

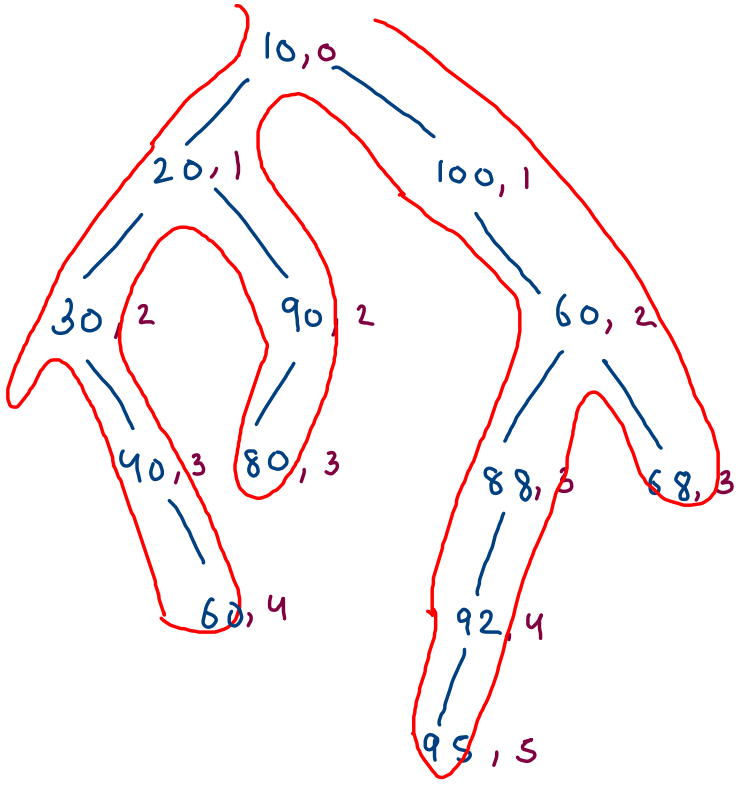
println()



Left View of Binary Tree



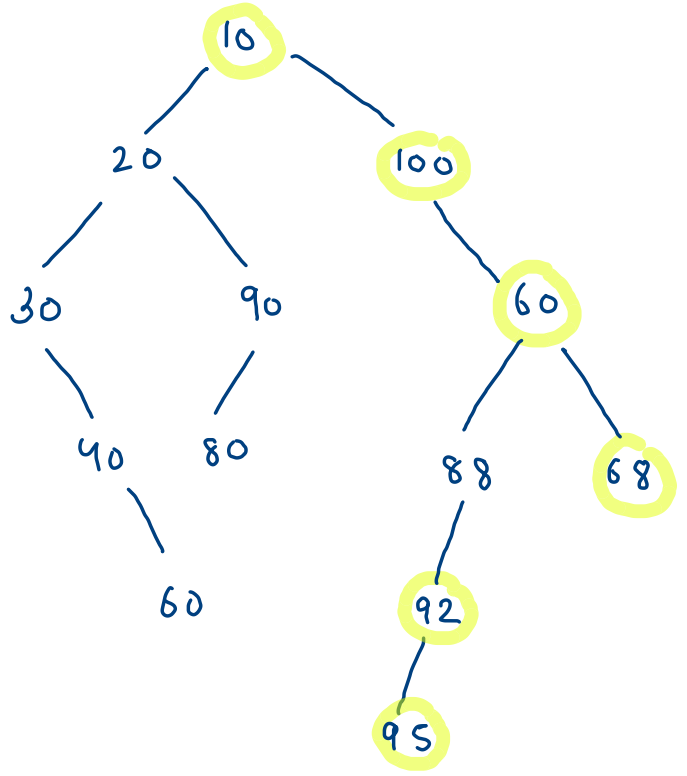
left view : each level's first node.



dfs

dv: 10 20 30 40 60 95

right view



right view = last node of each level