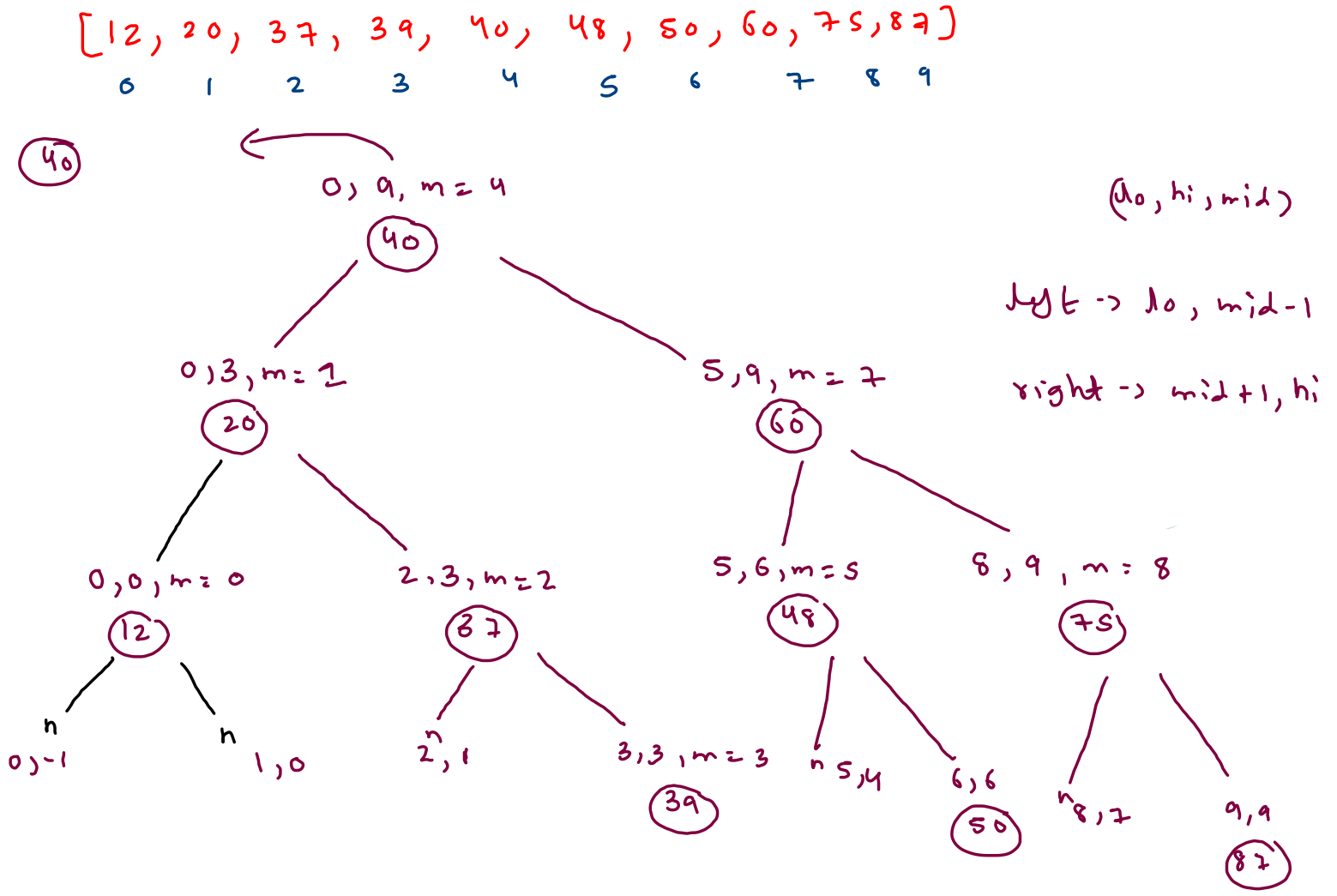
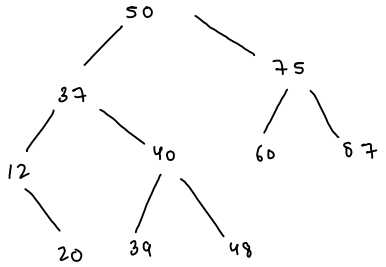

[12, 20, 37, 39, 40, 48, 50, 60, 75, 87]
0 1 2 3 4 5 6 7 8 9
40
0, 9, m = 4
40
(lo, hi, mid)
left -> lo, mid-1
right -> mid+1, hi
0, 3, m = 1
20
5, 9, m = 7
60
0, 0, m = 0
12
2, 3, m = 2
37
5, 6, m = 5
48
8, 9, m = 8
75
n
0, -1
n
1, 0
n
2, 1
3, 3, m = 3
39
n
5, 4
6, 6
50
n
8, 7
9, 9
87

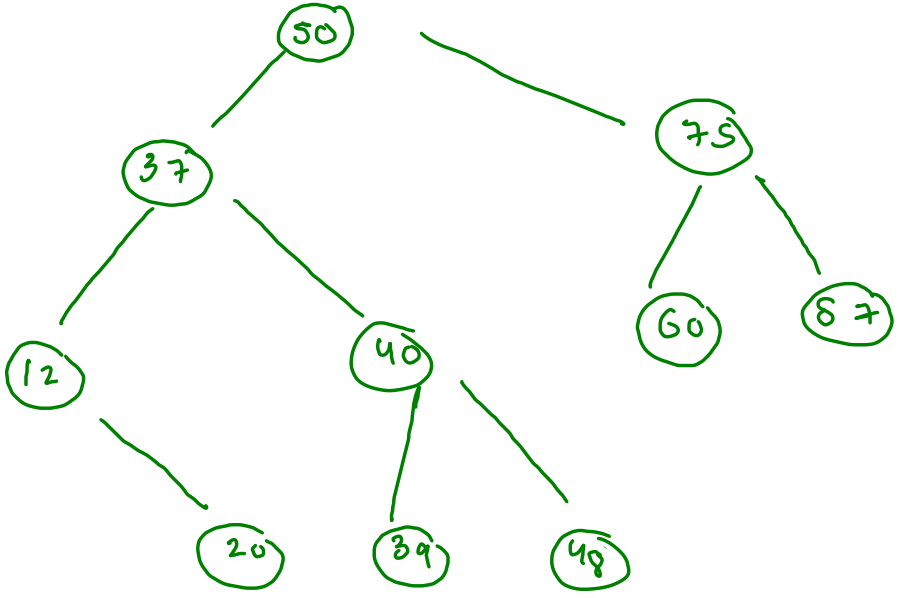


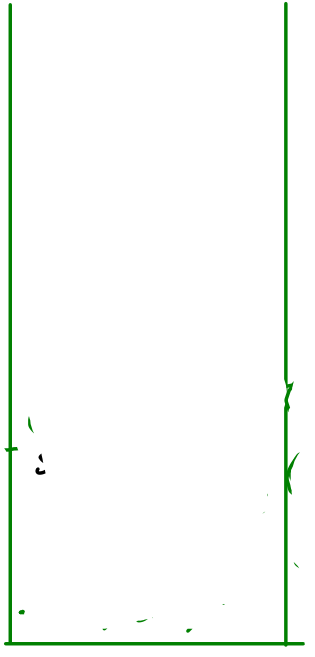

pre : [ 50, 37, 12, 20, 40, 39, 48, 75, 60, 87 ]

0 1 2 3 4 5 6 7 8 9

left -> right range

right -> left range



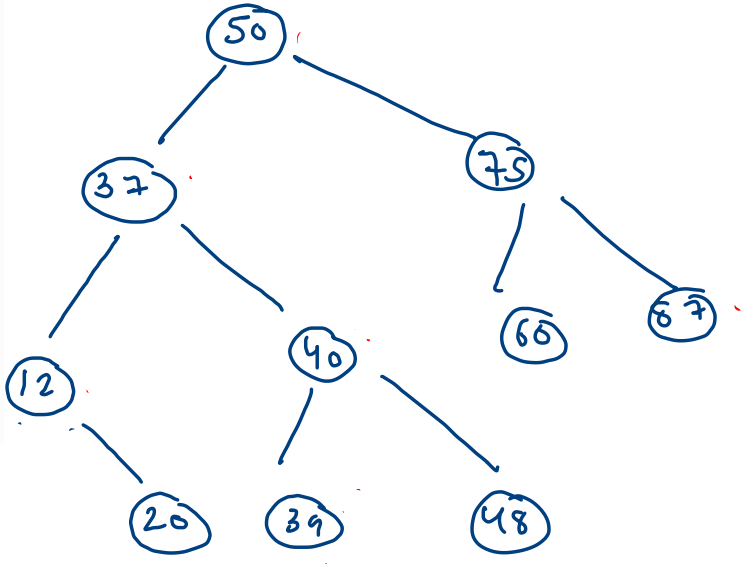
lr, rr

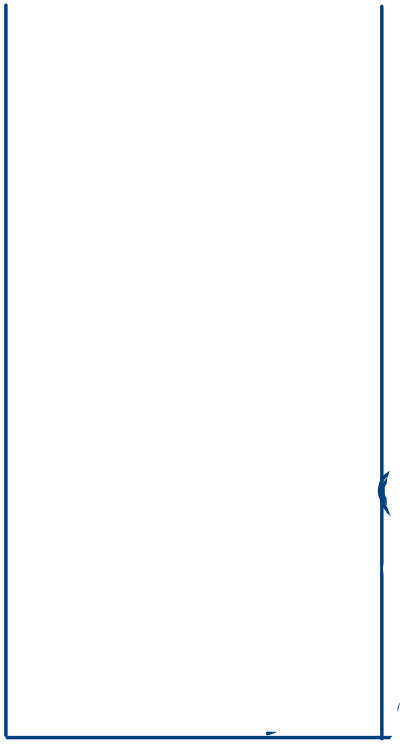
pre : [50, 37, 12, 20, 40, 39, 48, 75, 60, 87]

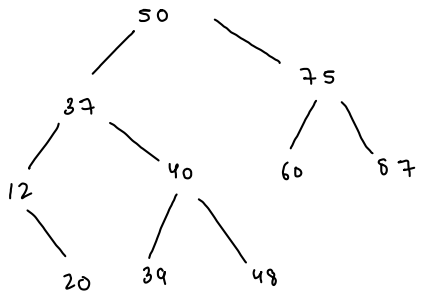
0 1 2 3 4 5 6 7 8 9



```java
static int idx;
//lr -> left range
//rr -> right range

public static TreeNode buildTree(int[]preorder,int lr,int rr) {
    if(idx >= preorder.length || preorder[idx] < lr || preorder[idx] > rr) {
        return null;
    }
    else {
        TreeNode node = new TreeNode(preorder[idx]);
        idx++;
        node.left = buildTree(preorder,lr,node.val);
        node.right = buildTree(preorder,node.val,rr);

        return node;
    }
}

public static TreeNode bstFromPreorder(int[] preorder) {
    idx = 0;
    return buildTree(preorder,Integer.MIN_VALUE,Integer.MAX_VALUE);
}
```





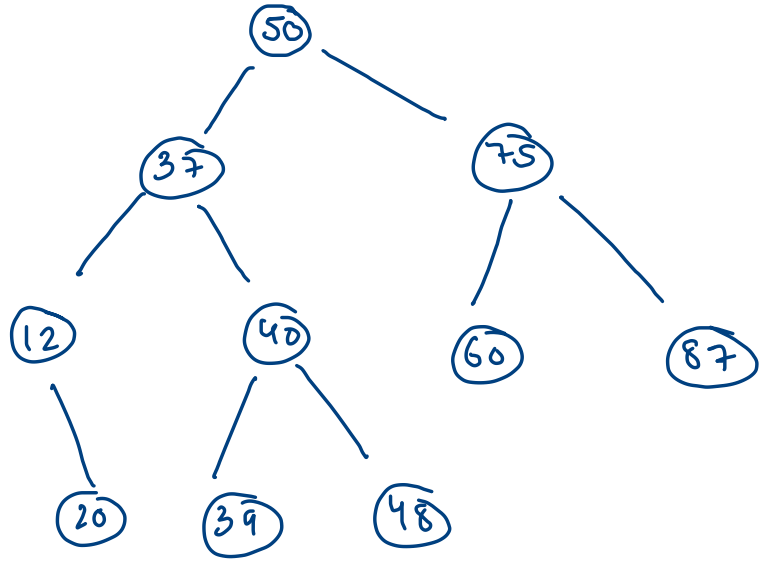

Post: 20, 12, 39, 48, 40, 37, 60, 87, 75, 50

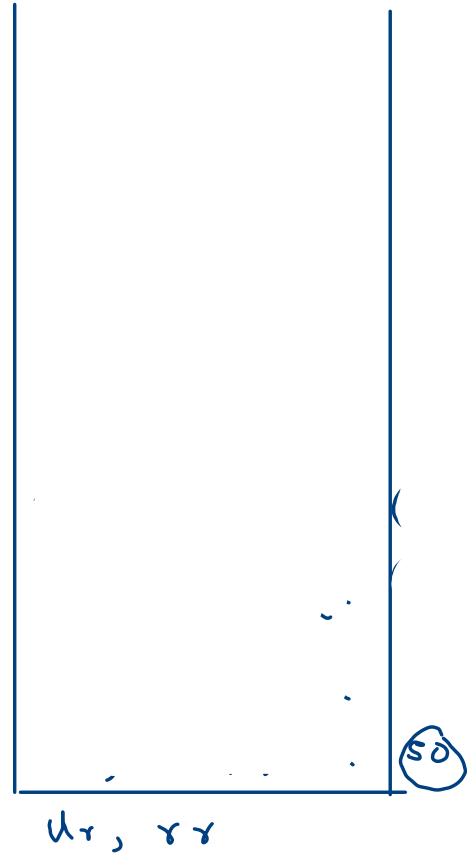
0 1 2 3 4 5 6 7 8 9

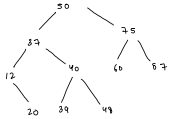
i /1

right -> left range

left -> right range

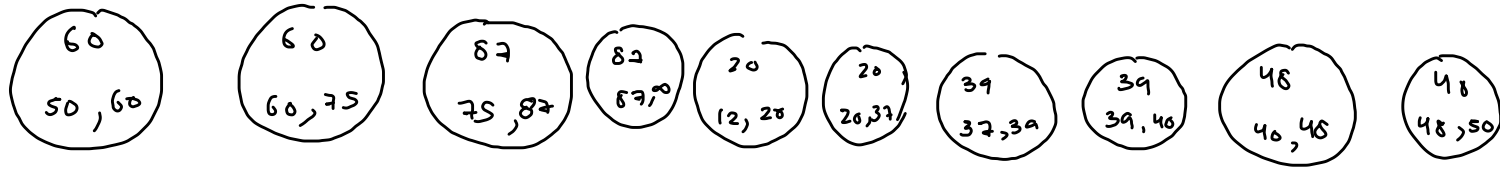




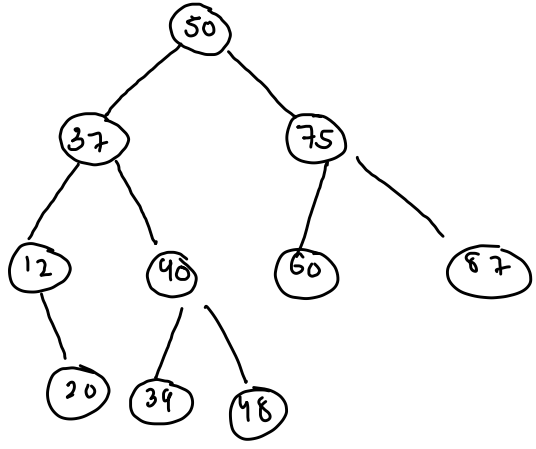

do: 50, 37, 75, 12, 40, 60, 87, 20, 39, 48

i



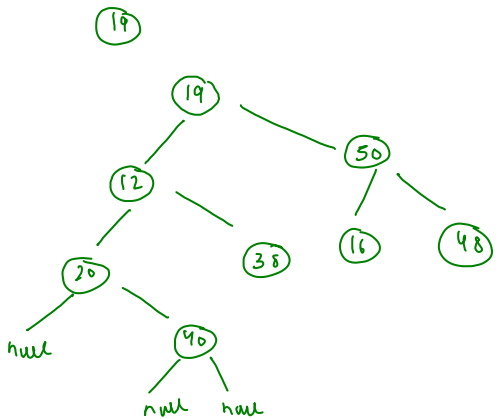

```java
public static TreeNode constructBSTFromLevelOrder(int[] LevelOrder) {
    ArrayDeque<Pair>q = new ArrayDeque<>();
    int n = LevelOrder.length;

    q.add(new Pair());
    TreeNode root = null;

    int idx = 0;

    while(q.size() > 0 && idx < n) {
        Pair rem = q.remove();

        int val = LevelOrder[idx];

        if(val < rem.lr || val > rem.rr) {
            continue;
        }

        TreeNode node = new TreeNode(val);
        idx++;

        if(rem.par == null) {
            root = node;
        }
        else {
            if(rem.par.val > node.val) {
                rem.par.left = node;
            }
            else if(rem.par.val < node.val){
                rem.par.right = node;
            }
        }

        Pair lcp = new Pair(node,rem.lr,node.val);
        Pair rcp = new Pair(node,node.val,rem.rr);

        q.add(lcp);
        q.add(rcp);

    }

    return root;
```

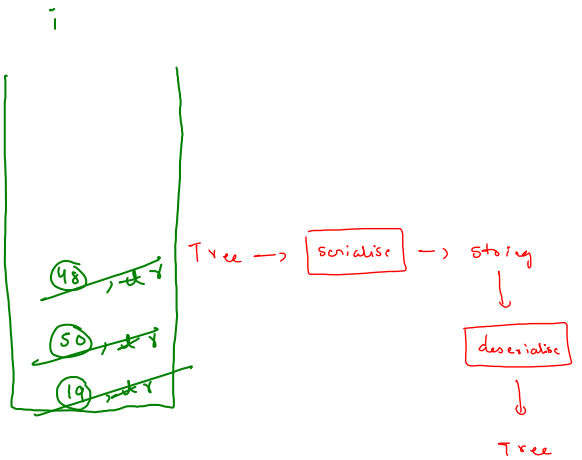
root = 50



Str: 19 12 20 n 40 n n 38 n n

50 16 n n 48 n n



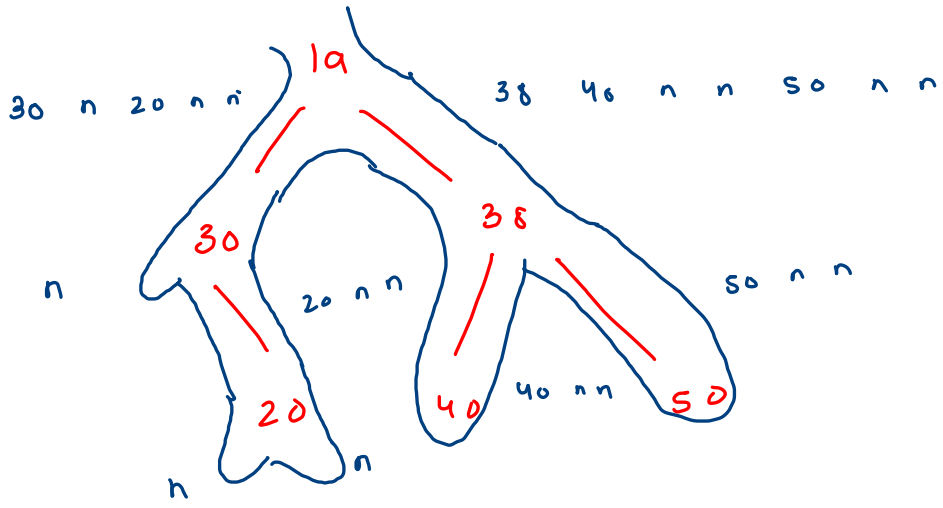

abc bcd def ing

abc | bcd def ing

bcd | def ing abc,

def | ig bcd

```java
public static String serialize(TreeNode root) {
    if(root == null) {
        return "n";
    }

    String ls = serialize(root.left);
    String rs = serialize(root.right);

    return root.data+ " " + ls + " " + rs;
}
```



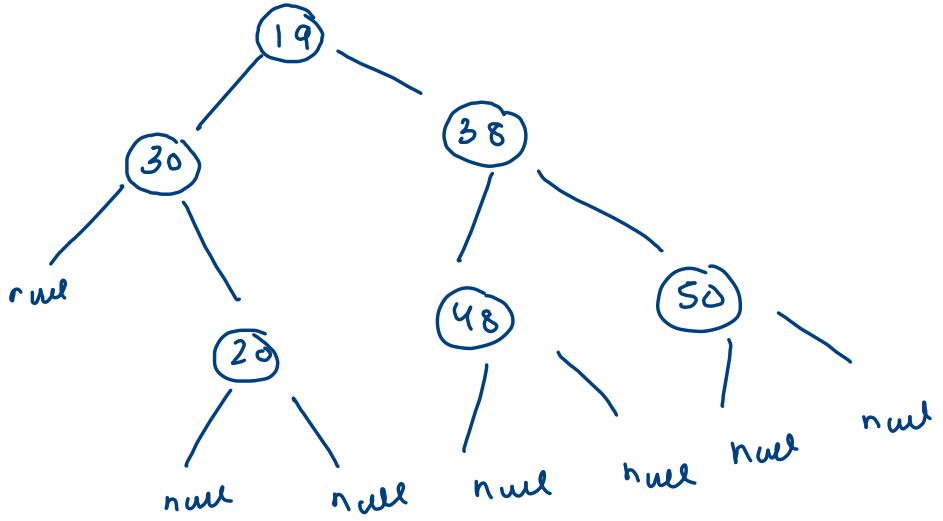

Serialise : 19 30 n 20 n n 38 40 n n 50 n n

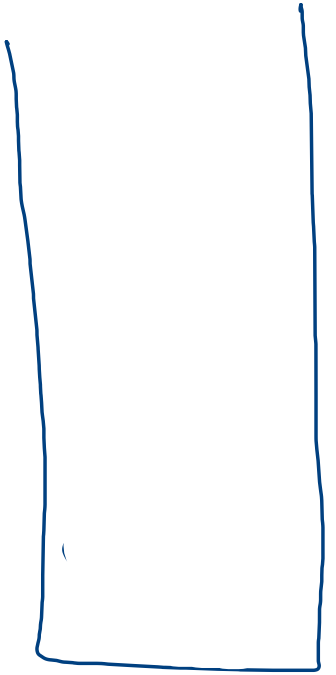
↓

String[] arr = [19, 30, n, 20, n, n, 38, 48, n, n, 50, n, n]

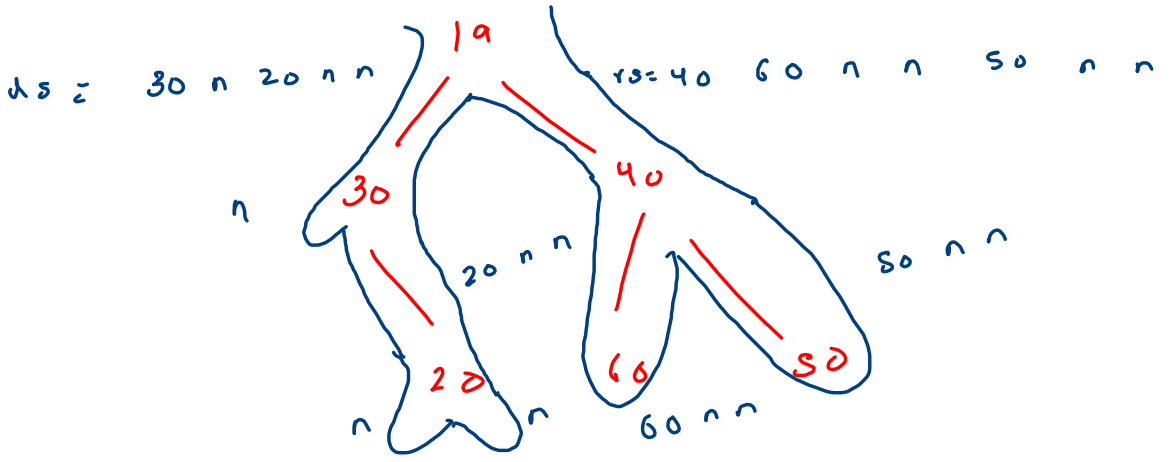
String[] arr = [19, 30, n, 20, n, n, 38, 48, n, n, 50, n, n]

i



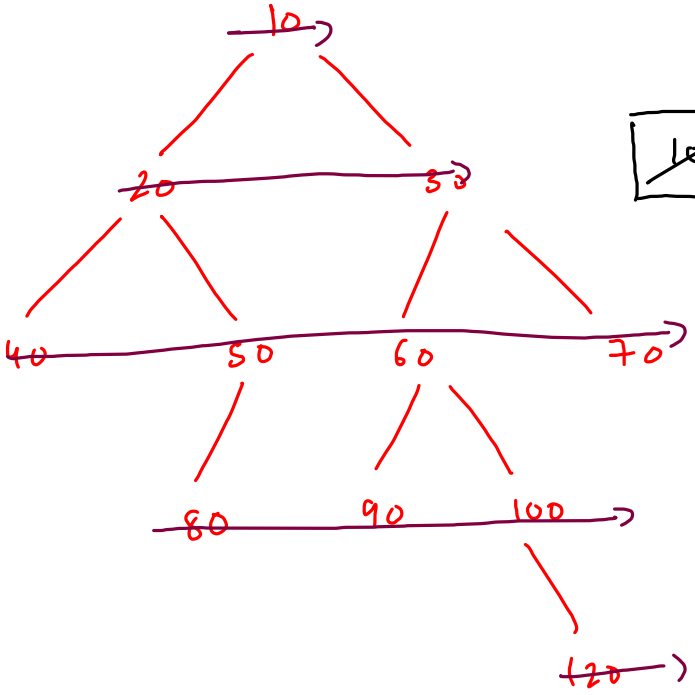



```java
public static String serialize(TreeNode root) {
    if(root == null) {
        return "n";
    }

    String ls = serialize(root.left);
    String rs = serialize(root.right);

    return root.data+ " " + ls + " " + rs;
}
```



level order line-wise

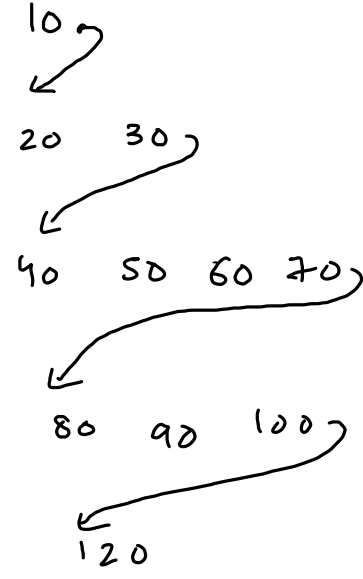
s = 1

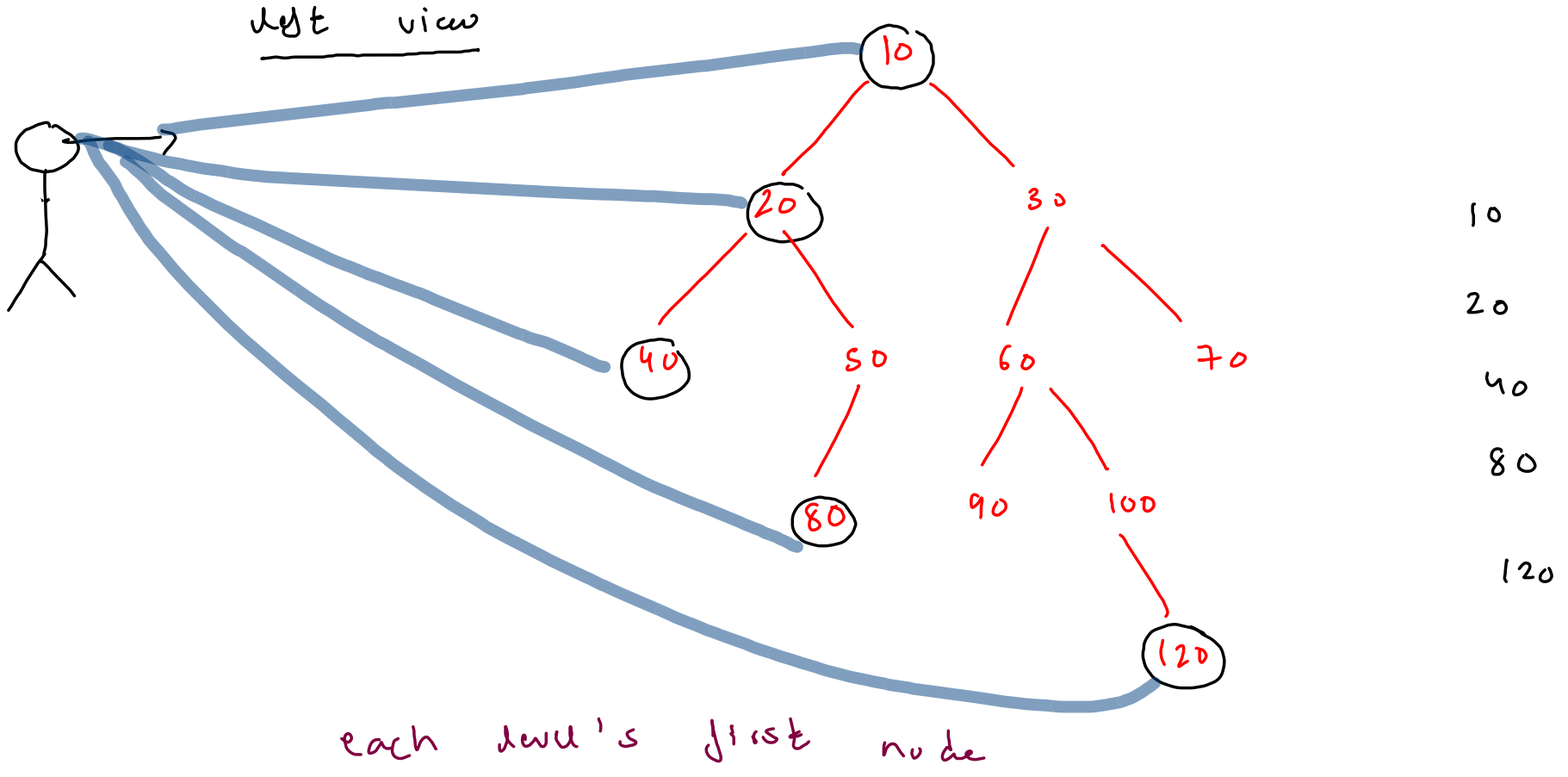



remove

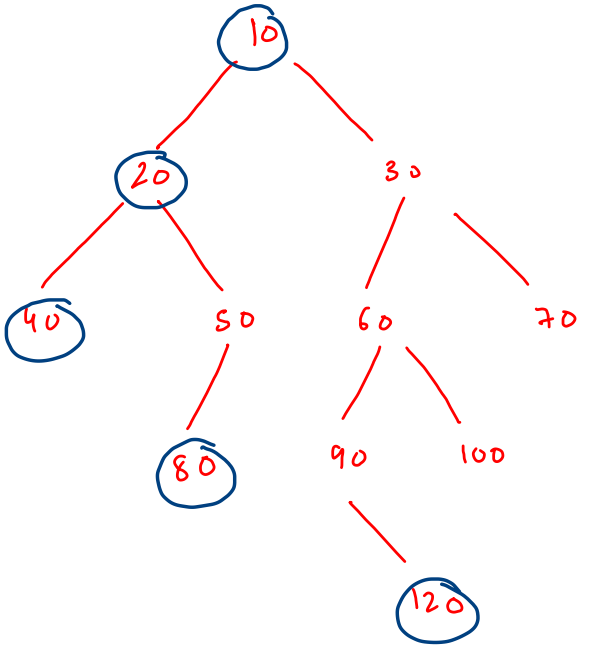
print

ac




left view
10
20
30
40
50
60
70
80
90
100
120
each level's first node
10
20
40
80
120

```java
public static ArrayList<Integer> leftView(TreeNode root) {
    ArrayList<Integer>ans = new ArrayList<>();
    ArrayDeque<TreeNode>q = new ArrayDeque<>();

    q.add(root);

    while(q.size() > 0) {
        int count = q.size();
        ans.add(q.peek().val);

        for(int i=0; i < count;i++) {
            TreeNode rem = q.remove();

            if(rem.left != null) {
                q.add(rem.left);
            }

            if(rem.right != null) {
                q.add(rem.right);
            }
        }
    }
```

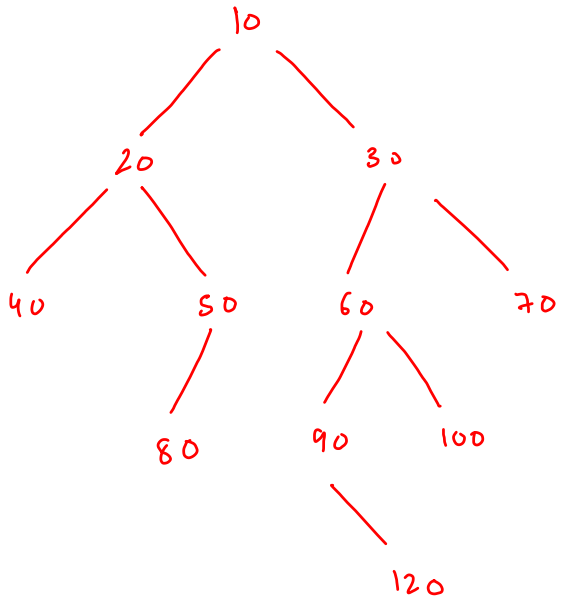



ans : 10 20 40 80 120

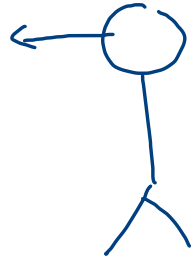
c = 1

| 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 120 |
|---|---|---|---|---|---|---|---|---|---|---|

rem, add children




right view



10 30 70 100 120

each level's last node