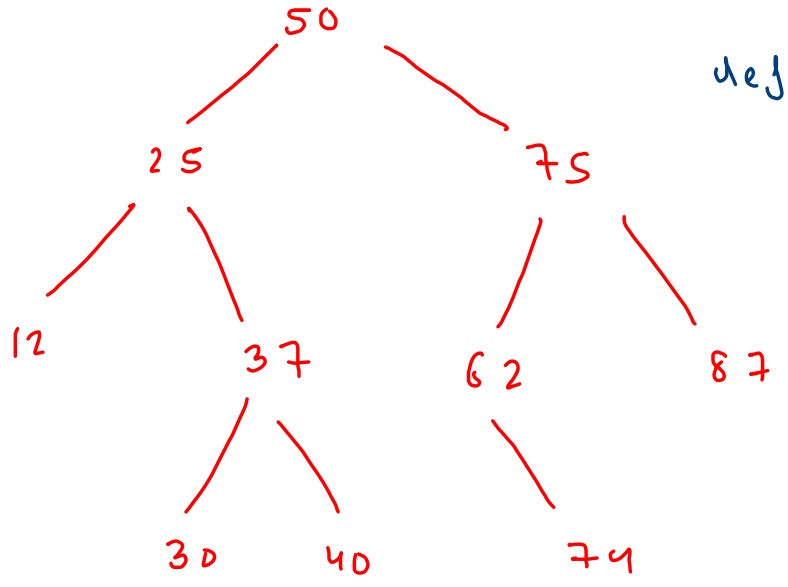
Binary search tree



$\forall$ nodes in < node.val < $\forall$ nodes

left subtree in right subtree

Inorder : L N R

BST inorder sorted.

inorder : 12 25 30 37 40 50 62 74 75 87

Sorted array → balanced bst

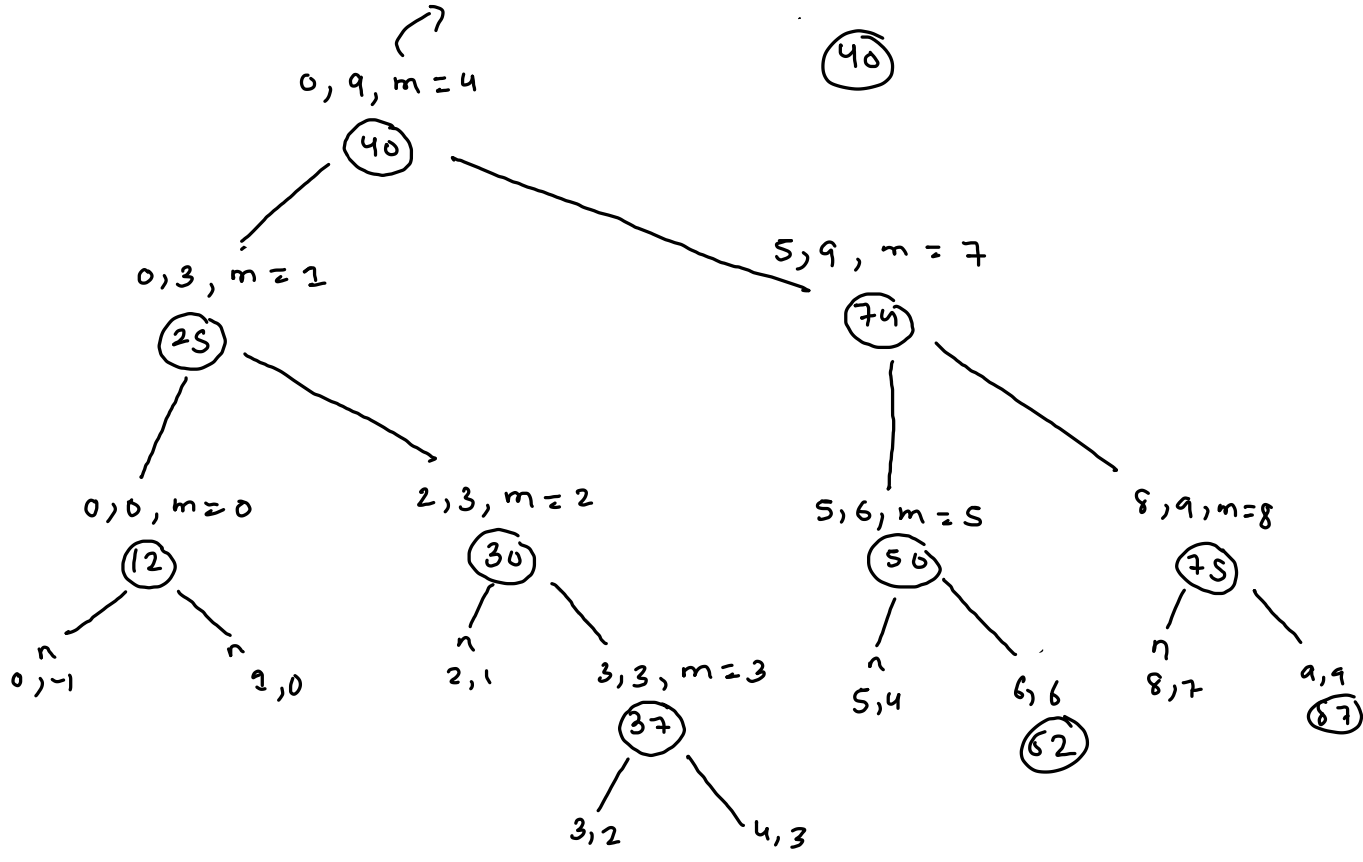
Construct

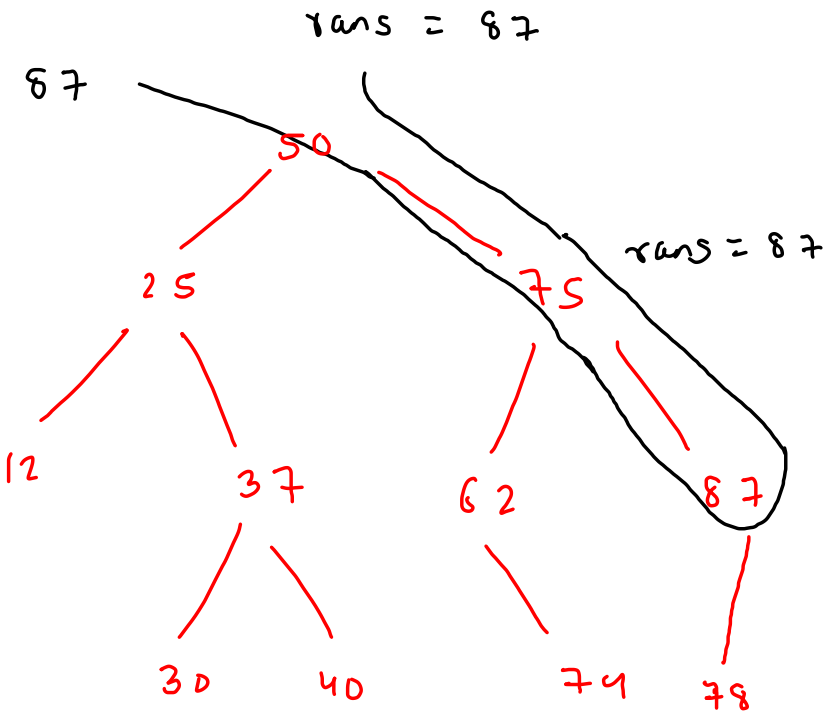
12	25	30	37	40	50	62	74	75	87
0	1	2	3	4	5	6	7	8	9

```java
public TreeNode helper(int[]nums,int lo,int hi) {
    if(lo > hi) {
        return null;
    }

    int mid = (lo + hi)/2;

    TreeNode node = new TreeNode(nums[mid]);

    node.left = helper(nums,lo,mid-1);
    node.right = helper(nums,mid+1,hi);

    return node;
}
```

left : lo, m-1

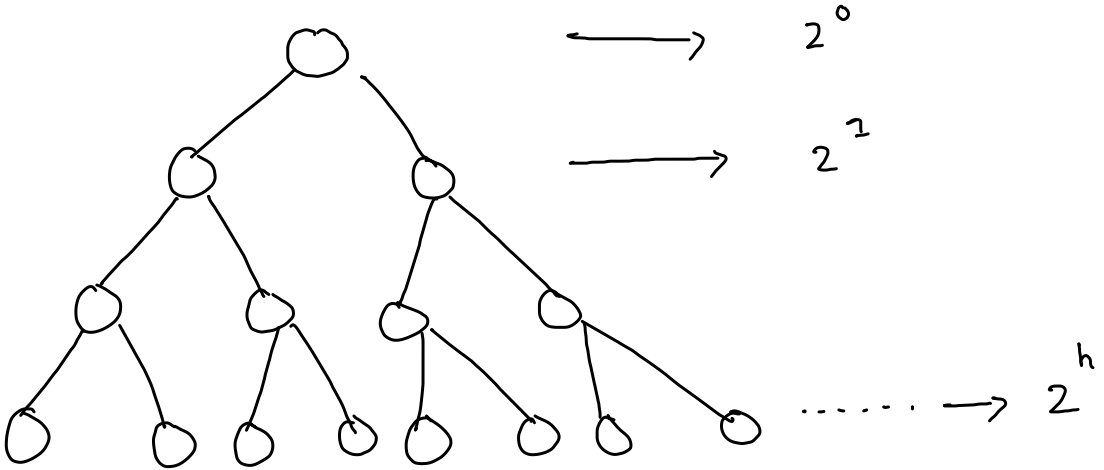
right : m+1, hi






```java
public static int max(Node node) {

    if(node.right != null) {
        int rans = max(node.right);
        return rans;
    }
    else {
        return node.data;
    }
}
```

O(h)



$$n = 2^0 + 2^1 + \cdots\cdots + 2^h$$

$$= 1\,(2^{h+1} - 1)$$

$$n = 2^{h+1}$$

$$\log_2 n = h + 1$$

$$h \propto \log_2 n$$

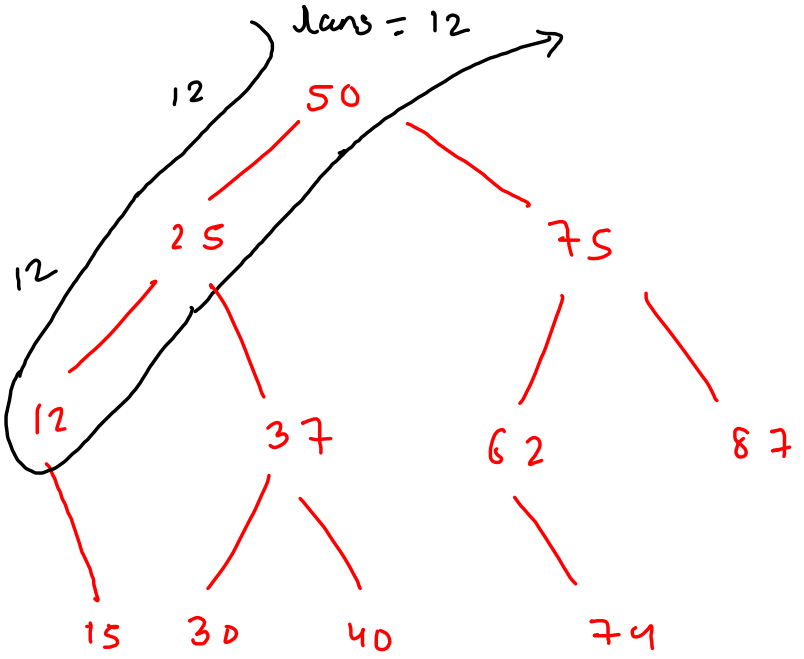
gp

$a = 1$

$r = 2$

$t = h+1$

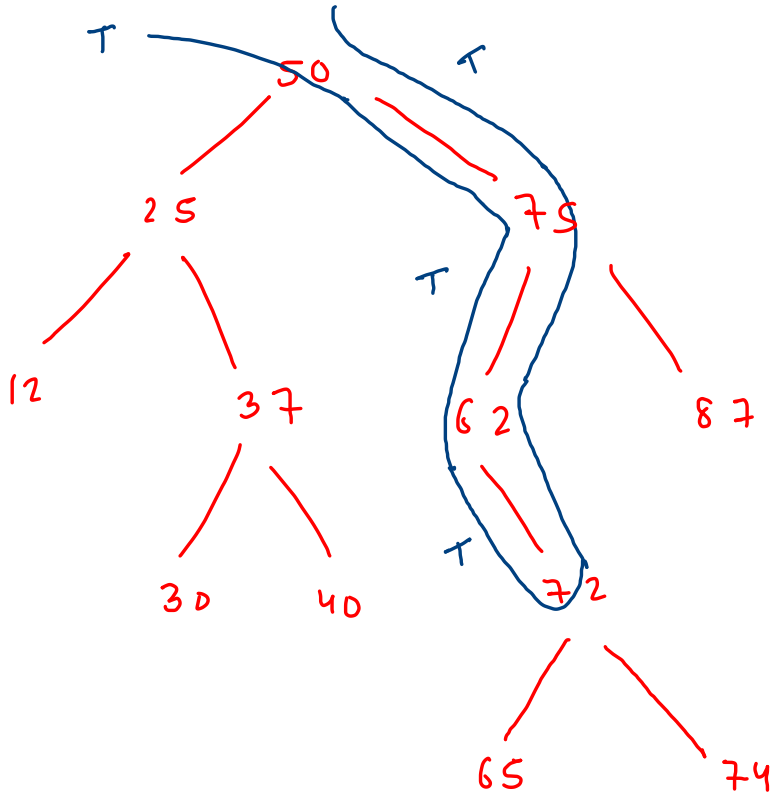
$$S = \frac{a(r^t - 1)}{r - 1}$$



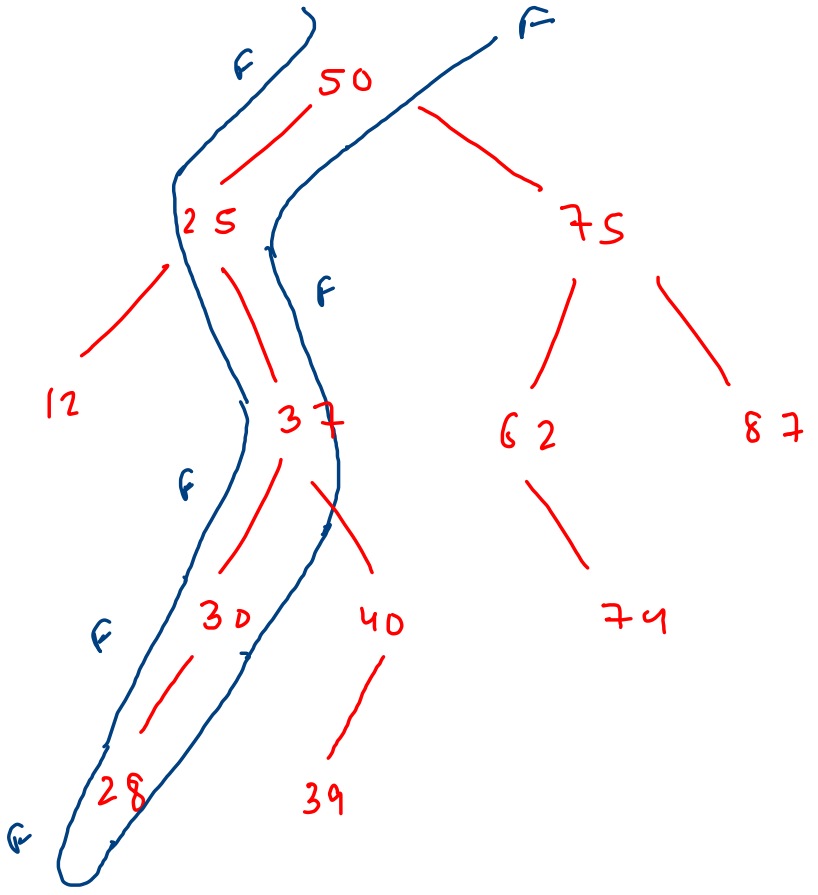

```java
public static int min(Node node) {
    if(node.left != null) {
        int lans = min(node.left);
        return lans;
    }
    else {
        return node.data;
    }
}
```

find

data = 72

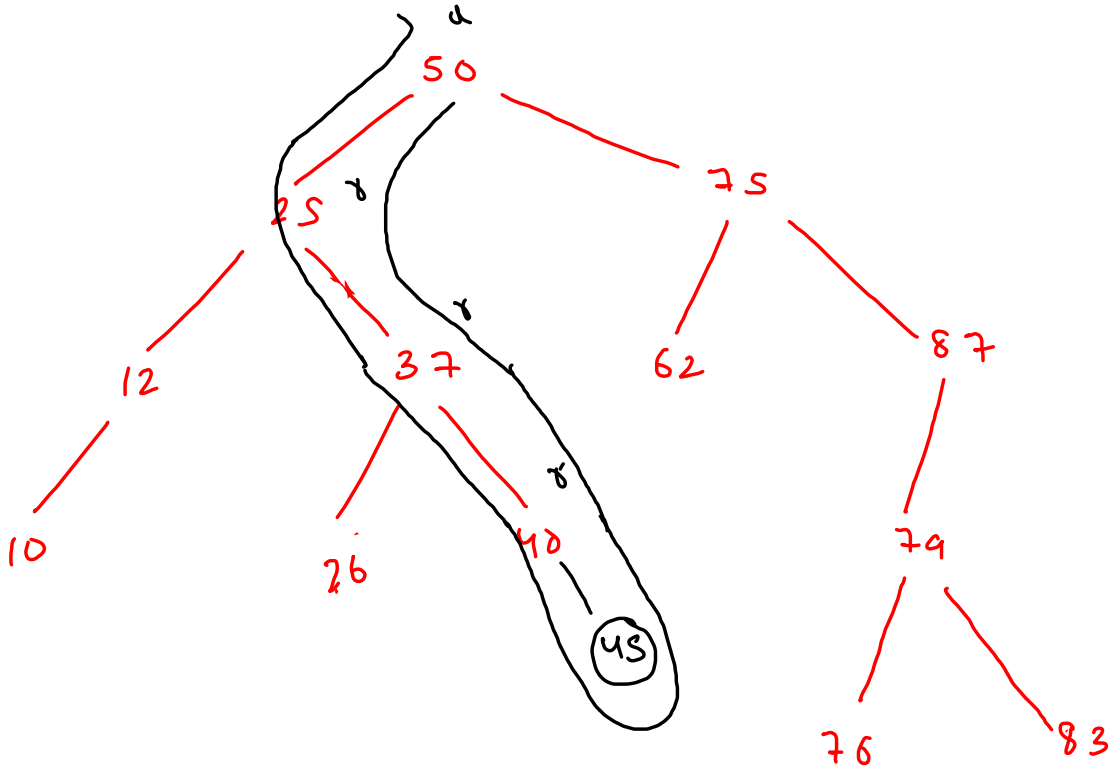



```java
public static boolean find(Node node, int data){
    if(node == null) {
        return false;
    }

    if(node.data < data) {
        //go to right
        return find(node.right,data);
    }
    else if(node.data > data) {
        //go to left
        return find(node.left,data);
    }
    else {
        return true;
    }
}
```




```java
public static boolean find(Node node, int data){
    if(node == null) {
        return false;
    }

    if(node.data < data) {
        //go to right
        return find(node.right,data);
    }
    else if(node.data > data) {
        //go to left
        return find(node.left,data);
    }
    else {
        return true;
    }
}
```

data = 26



add (28)

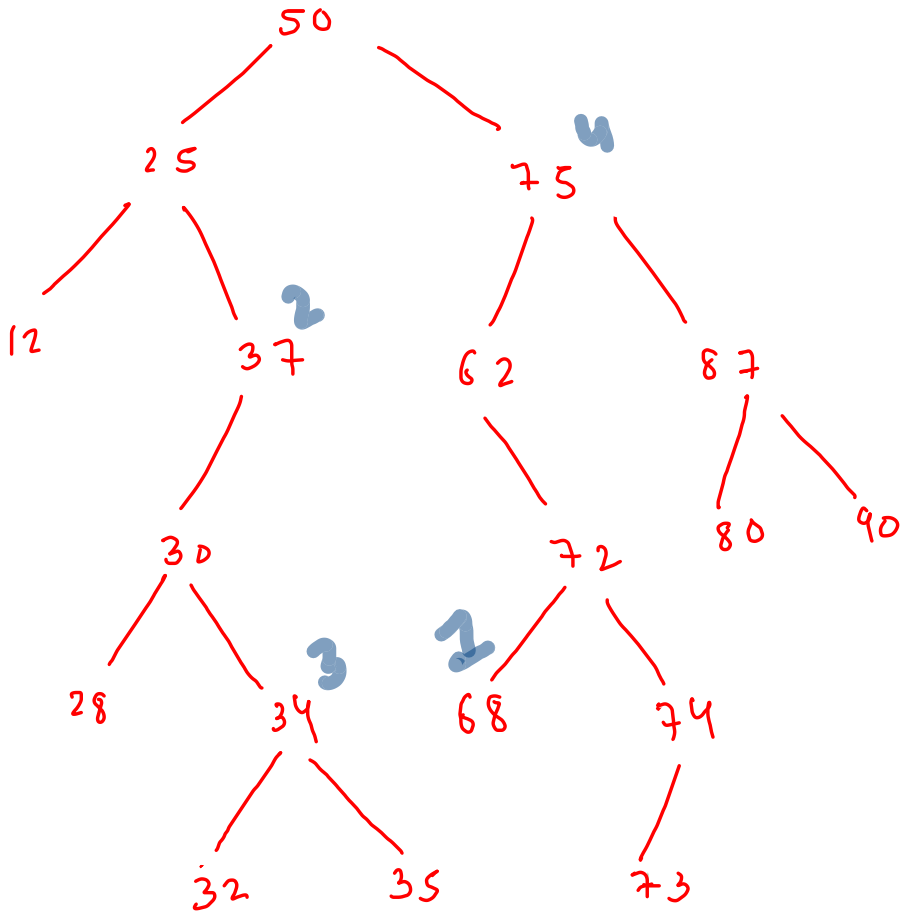
add (77)

add (37) no work

add (45)

```java
public static Node add(Node node, int data) {
    if(node == null) {
        return new Node(data,null,null);
    }

    if(node.data < data) {
        //go to right
        node.right = add(node.right,data);
    }
    else if(node.data > data) {
        //go to left
        node.left = add(node.left,data);
    }
    else {
        //do nothing
    }

    return node;
}
```

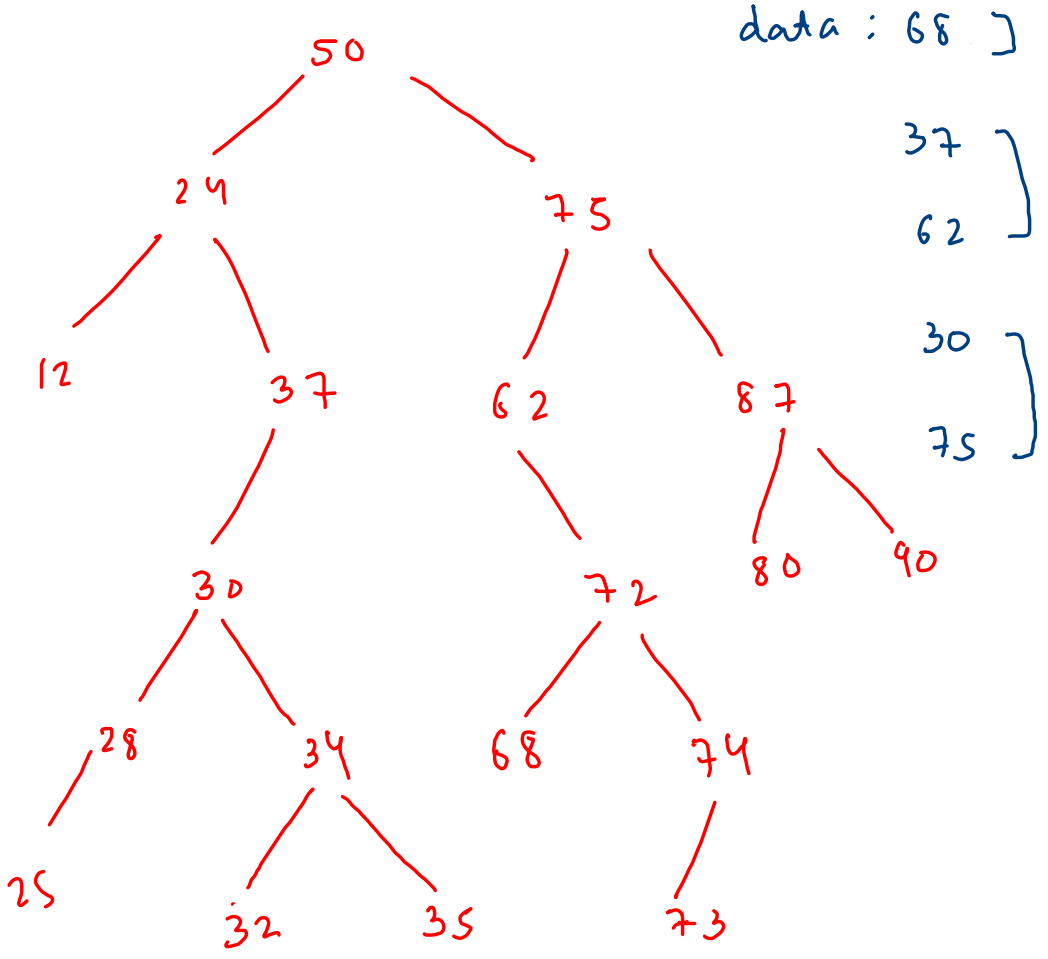
remove node



① node → 0 children
return null on your behalf.

② node → 1 child
return your single child on your behalf

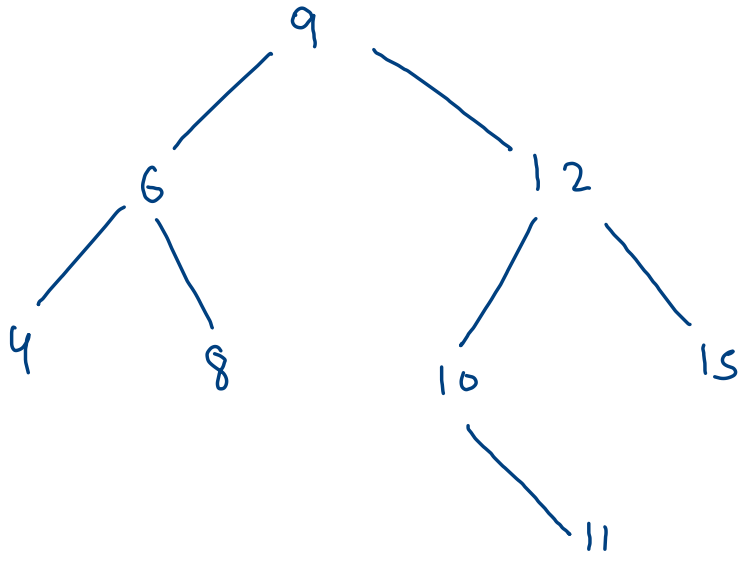
③ node → 2 child
node.data = dmax;
remove (node.left, dmax);




```java
public static Node remove(Node node, int data) {
    if(node == null) {
        //data is not present in the tree
        return null;
    }

    if(node.data < data) {
        node.right = remove(node.right,data);
    }
    else if(node.data > data) {
        node.left = remove(node.left,data);
    }
    else {
        //perform removal
        if(node.left == null && node.right == null) {
            //node is a leaf node
            return null;
        }
        else if(node.left == null) {
            //single child -> right child
            return node.right;
        }
        else if(node.right == null) {
            //single child -> left child
            return node.left;
        }
        else {
            //node has both the childs
            int lmax = max(node.left);
            node.data = lmax;
            node.left = remove(node.left,lmax);
        }
    }

    return node;
```

replace with
Sum of larger

Sum = 0



regular inorder

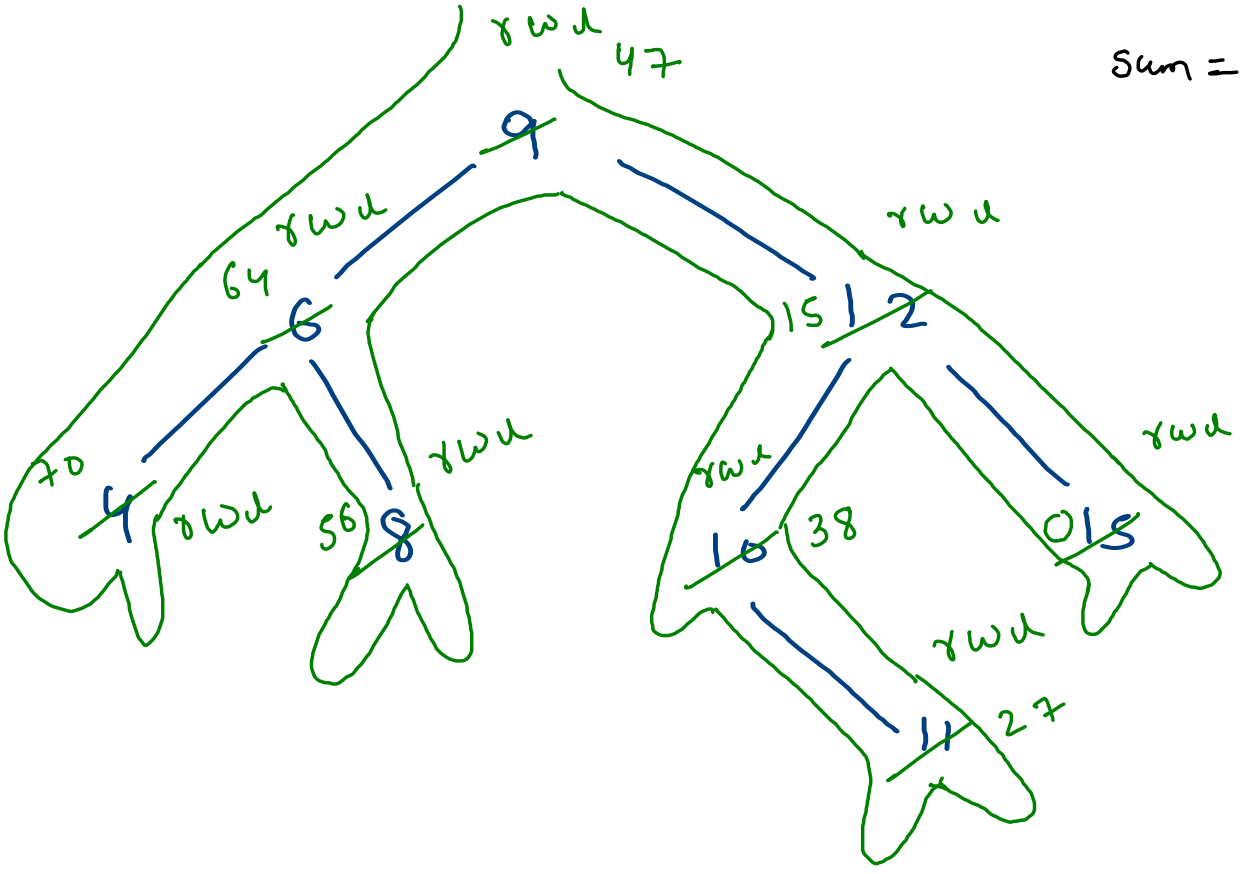
L N R

reverse inorder

R N L

in : 4 6 8 9 10 11 12 15

```
call (node.right);
work();
call (node.left);
```



Sum = 0 + 15 + 12 + 11

+ 10 + 9 + 8 + 6 + 4

```java
static int sum = 0;
public static void rwsol(Node node){
    if(node == null) {
        return;
    }

    rwsol(node.right);

    //work
    int temp = node.data;
    node.data = sum;
    sum += temp;

    rwsol(node.left);
}
```