Sum of Left Leaves 1 (Day 44)

Problem | Submissions | Leaderboard | Discussions

Given preorder of binary tree first you need to construct binary tree, you have root of a binary tree, return the sum of all left leaves.

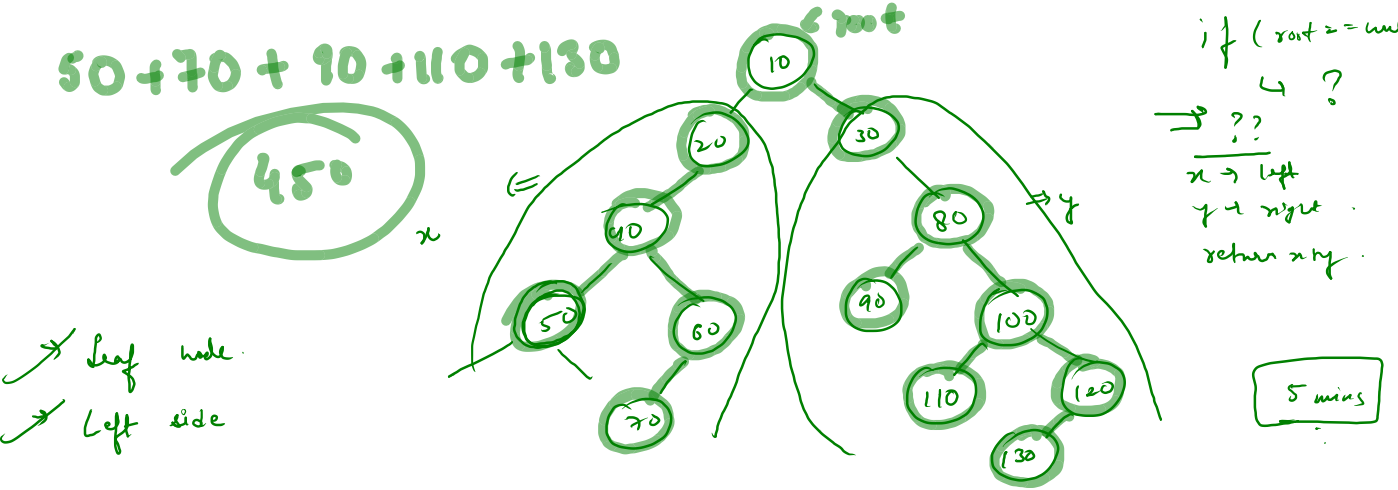
A leaf is a node with no children. A left leaf is a leaf that is the left child of another node.

Sample Input 0

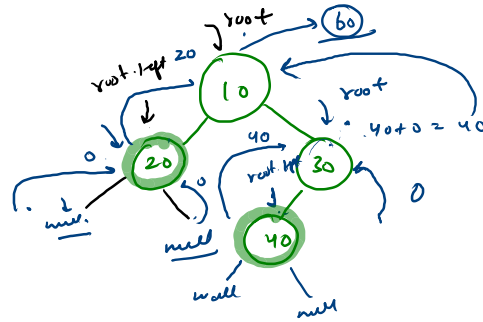
```
23
10 20 50 n 60 n n 30 70 65 n n 80 n 90 n n 40 100 n n n n
```

Sample Output 0

```
165
```



```java
public static int sumOfLeftLeaves(Node root){
    if(root == null){
        return 0;
    }

    if(root.left != null && root.left.left == null && root.left.right == null){
        return root.left.data + sumOfLeftLeaves(root.right);
    }

    int leftSide = sumOfLeftLeaves(root.left);
    int rightSide = sumOfLeftLeaves(root.right);

    return leftSide + rightSide;
}
```



20 + 40 ⇒ 60

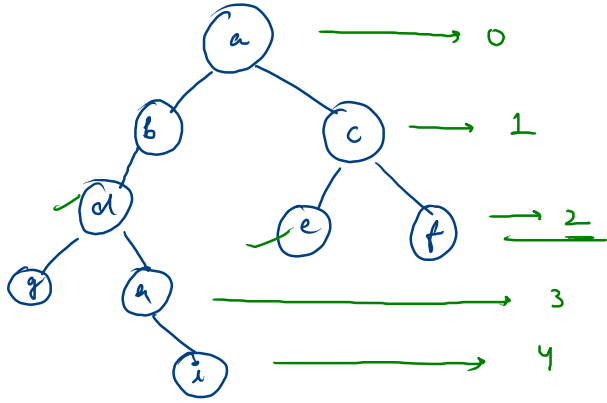
Cousins in Binary Tree (Day 45)

Problem | Submissions | Leaderboard | Discussions

Given preorder of binary tree first you need to construct binary tree, now you have root of a binary tree with unique values and the values of two different nodes of the tree x and y, return true if the nodes corresponding to the values x and y in the tree are cousins, or false otherwise.

Two nodes of a binary tree are cousins if they have the same depth with different parents.

Note that in a binary tree, the root node is at the depth 0, and children of each depth k node are at the depth k + 1.

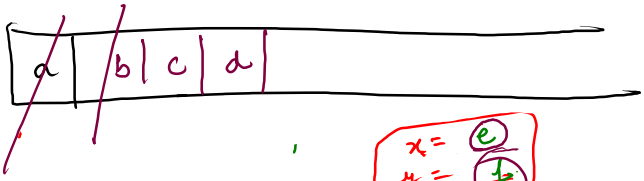


→ Same level ✓

→ their parents are different

e, f ✗

d, c → true ✓

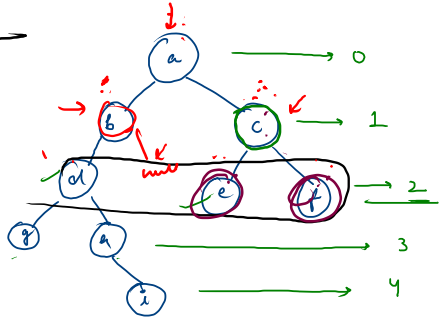


temp. = ~~a~~ ~~b~~ c

x = e
y = f

q.add(root)

temp = d



sz = ~~2~~ 1

is x exist = false

is y exist = false.

```
while ( q.size() != 0 ) {
    int sz = q.size();   // ~~3~~ => 1
    is x exist = false;
    is y exist = false;
    for ( int i = 0; i < sz ; i++ )
    {
        Node temp = q.remove();
        if (temp.left != null)
            q.add (temp.left)
        if (temp.right != null)
            q.add (temp.right)
    }
```

```
if (temp.val == x)  is x exist = true;
if (temp.val == y)  is y exist = true;
```

```
if (temp.left != null  && temp.right != null)
{
    if (temp.left.val == x && temp.right.val == y)
        return false;          // parents are same
    if (temp.left.val == y && temp.right.val == x)
        return false;
}
```

parent is same or not

```
if (is x exist == true && is y exist == true)
    return true;
return false
```

```java
public static boolean isCousins(Node root, int x, int y){
    Queue<Node> q = new LinkedList<>();

    q.add(root);

    while(q.size() != 0){
        int sz = q.size();
        boolean isXExist = false;
        boolean isYExist = false;

        for(int i=0;i<sz;i++){
            Node temp = q.remove();

            // Same Level
            if(temp.data == x) isXExist = true;
            if(temp.data == y) isYExist = true;

            // Same parent or not
            if(temp.left != null && temp.right!= null){
                if(temp.left.data == x && temp.right.data == y){
                    return false;
                }

                if(temp.left.data == y && temp.right.data == x){
                    return false;
                }
            }

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

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

        if(isXExist == true && isYExist == true){
            return true;
        }
    }
    return false;

}
```

→ Construction of B.T.

→ At least 2 traversals.

→ Postorder + Inorder.

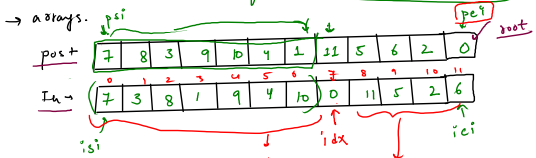
→ Preorder + Inorder.

→ Postorder + Preorder.

→ Levelorder

Construct B.T from Postorder & Inorder.

→ arrays.



psi = 0
pei = 11
isi = 0
iei = 6

→ psi , pei
→ isi , iei

post → L R N

In → L N R

→

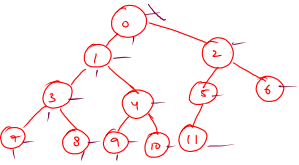
idx = isi

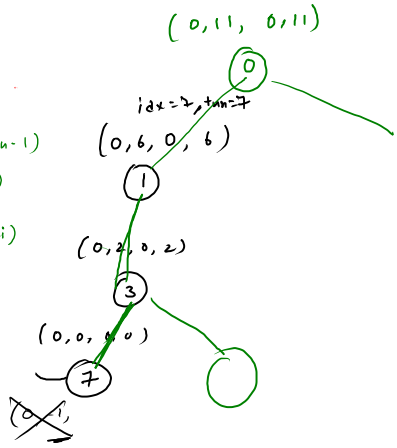
while (in[idx] != post[pei])
idx++;

tnn → idx - isi → 7 - 0 = 7

{ leftp = (psi , psi+tnn-1)
lefti = (isi , idx-1)

{ rightp = (psi+tnn , pei)
righti = (idx+1 , iei)



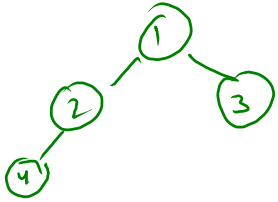


(0,2,0,2) → ~~l~~ r

(0,6,0,6) → l

(0,11,0,11) · l

```java
public static Node helper(int [] post,int psi, int pei,int [] in, int isi,int iei){
    if(isi>iei){
        return null;
    }
    Node node = new Node(post[pei]);

    int idx = isi;

    while(in[idx] != post[pei]){
        idx++;
    }

    int tnn = idx-isi;

    node.left = helper(post,psi,psi+tnn-1,in,isi,idx-1);
    node.right = helper(post,psi+tnn, pei-1, in,idx+1,iei);

    return node;

}
```




```java
public static Node constructTree(int [] post, int [] in){
    return helper(post,0,post.length-1, in, 0,in.length-1);
}
```