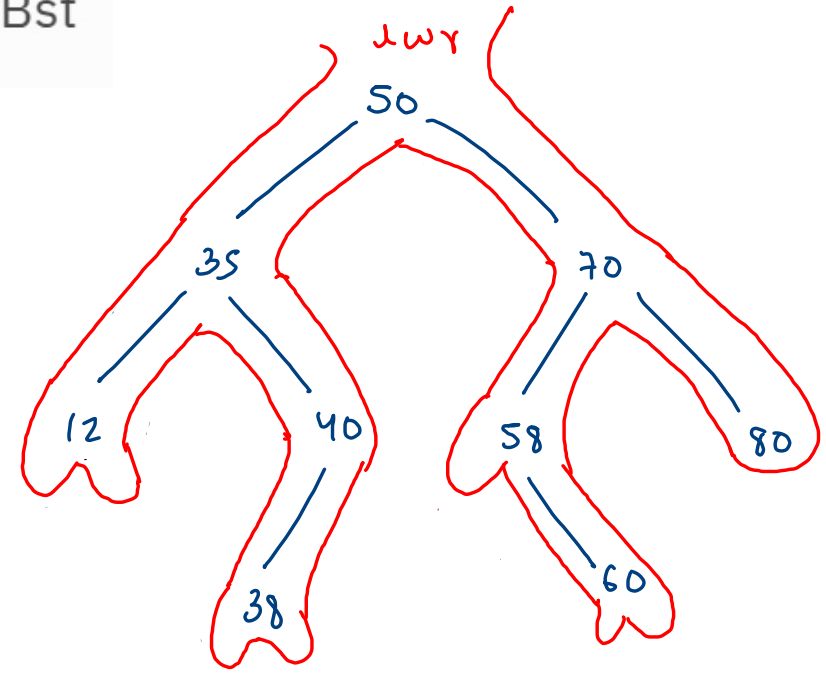
Validate Bst



(i) extra space

(ii) O(n)

p = null 12 35 38 40 50 58 60 70

helper(node.left);

```
//work
if(p.val > n.val){
    return false;
}
prev = node;
```

Inorder : 12 35 38 40 50 58 60 70 80

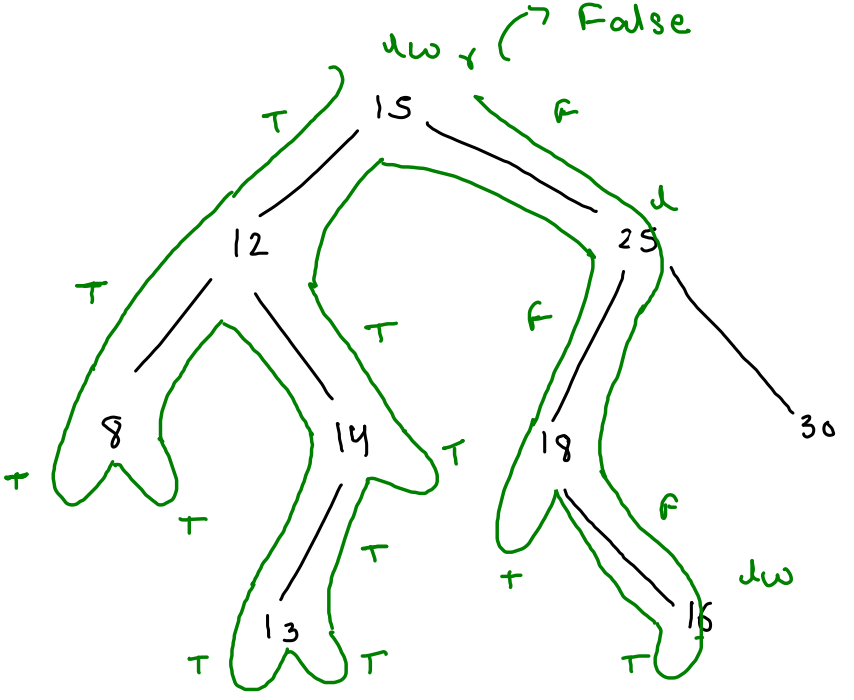
p n

helper(node.right);

```java
public static boolean helper(TreeNode node) {
    if(node == null) {
        return true;
    }

    boolean lans = helper(node.left);

    //work and update (inorder)
    if(prev != null && prev.val >= node.val) {
        return false;
    }
    prev = node;

    boolean rans = helper(node.right);

    return lans && rans;
}
```

l
w
r

p : ~~n~~ ~~8~~
~~12~~
~~13~~
~~14~~
~~15~~
18

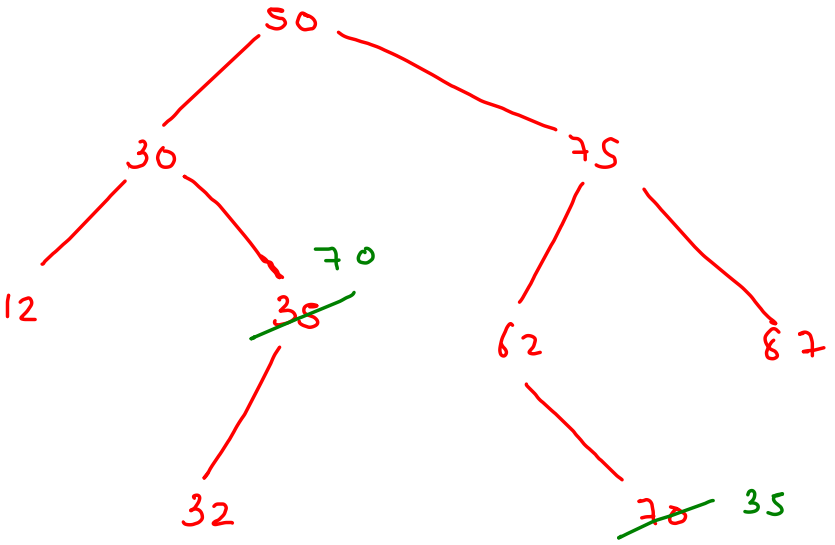


inorder : 8 12 13 14 15 18 16 25 30

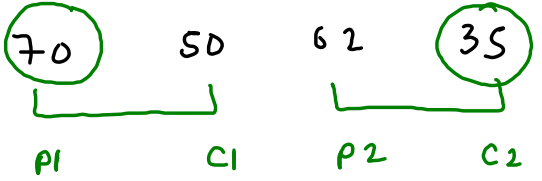
p n

99. Recover Binary Search Tree

You are given the `root` of a binary search tree (BST), where the values of **exactly** two nodes of the tree were swapped by mistake. *Recover the tree without changing its structure.*



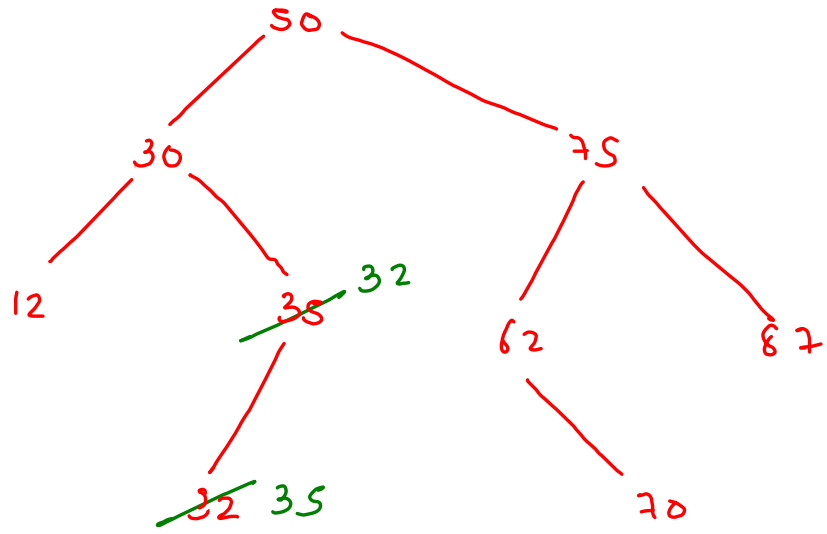
inorder : 12 30 32 70 50 62 35 75 87



fn = p1

sn = c2

two mistake



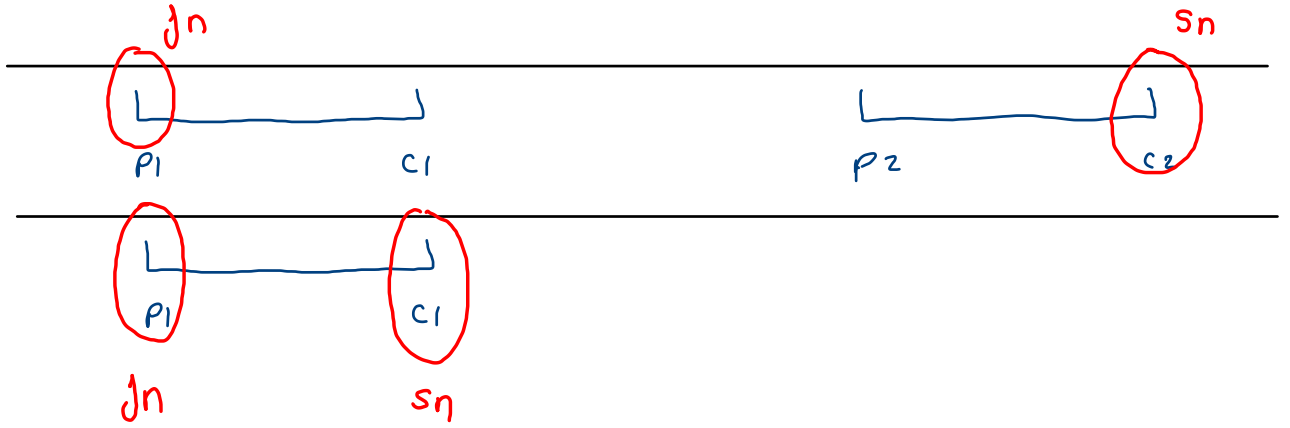
only one mistake

in = p1

sn = c1

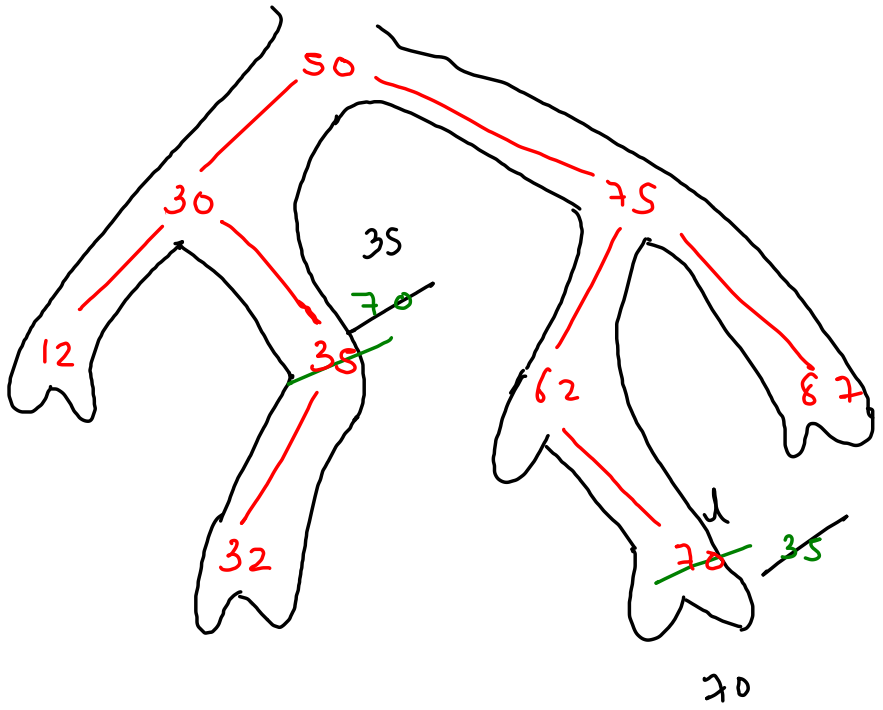
12 30 35 32 50 62 70 75 87

p1 c1




```
jn = null;
sn = null;
```

```
if ( prev.val ≥ cur.val )
    if ( sn == null ) {
        jn = p;
    }
    sn = c;
}
```



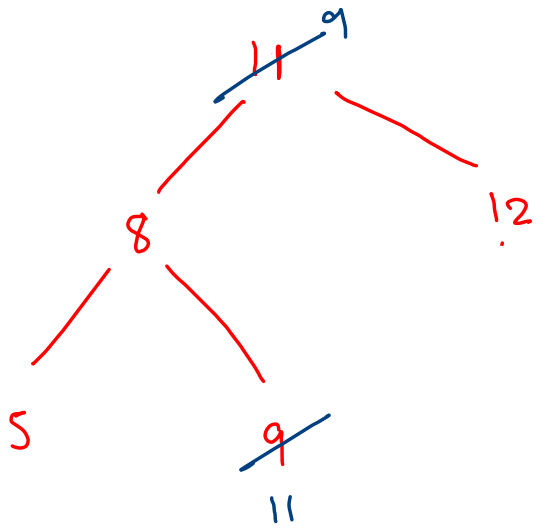

```
public static void helper(TreeNode node) {
    if(node == null) {
        return;
    }

    helper(node.left);


    //work and update(inorder)
    if(prev != null && prev.val >= node.val) {
        if(sn == null) {
            fn = prev;
        }
        sn = node;
    }
    prev = node;

    helper(node.right);
}
```

fn = ~~x~~ 70

sn = ~~x~~ ~~50~~ 35

p = ~~x~~ ~~12~~ ~~30~~ ~~35~~ ~~32~~ ~~70~~ ~~62~~ ~~50~~ ~~75~~ 87

```java
public static void helper(TreeNode node) {
    if(node == null) {
        return;
    }

    helper(node.left);


    //work and update(inorder)
    if(prev != null && prev.val >= node.val) {
        if(sn == null) {
            fn = prev;
        }
        sn = node;
    }
    prev = node;

    helper(node.right);
}
```



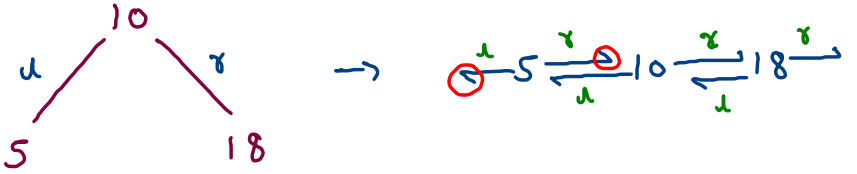
5 8 11 9 12

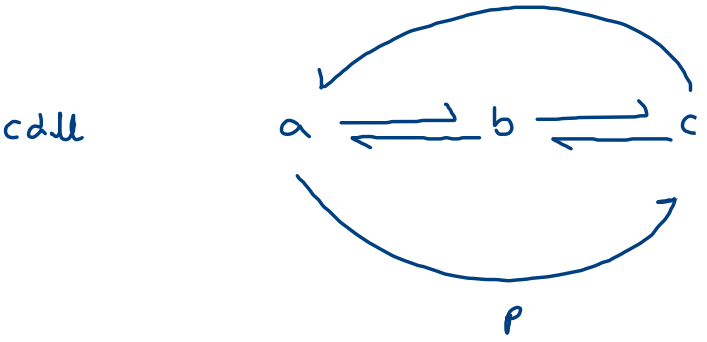
fn = 11

sn = 9

Convert Bst To Sorted Doubly Linked List

1. Convert a Binary Search Tree to a sorted Circular Doubly-Linked List in place.
2. The left and right pointers in nodes are to be used as previous and next pointers respectively in converted DLL.
3. The order of nodes in DLL must be the same as in Inorder for the given Binary Search Tree. The first node of Inorder traversal (leftmost node in BST) must be the head node of the DLL.

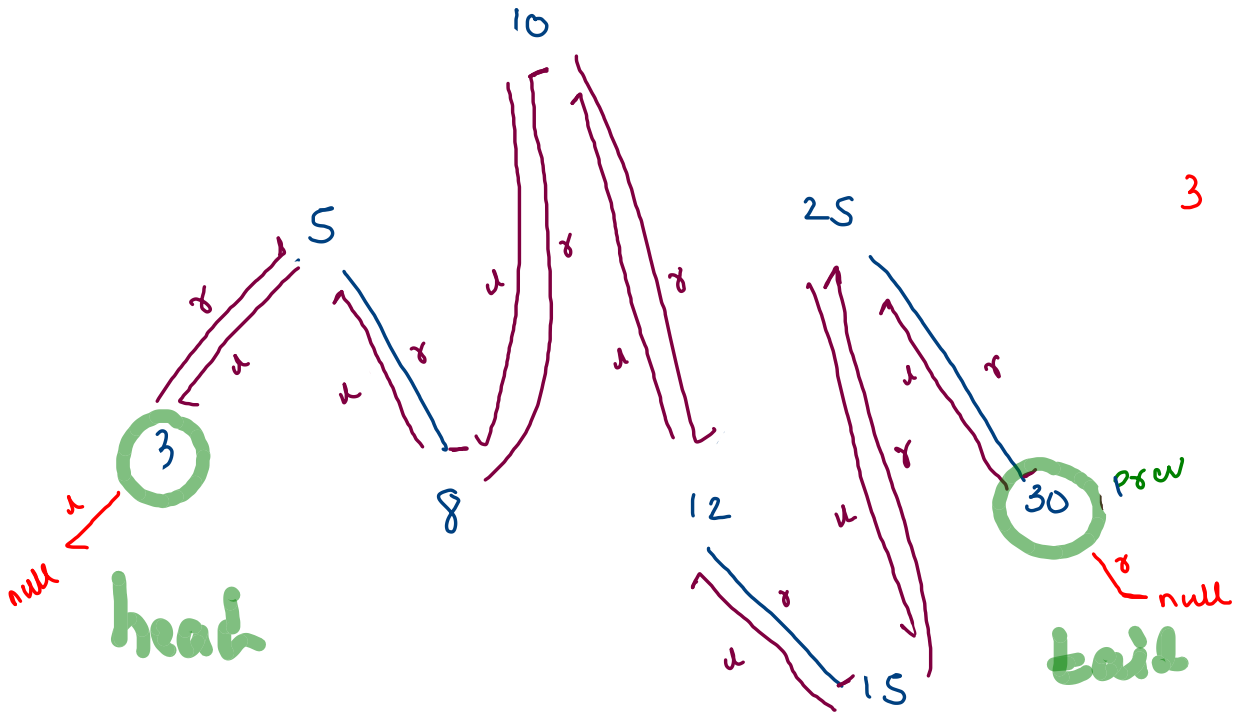




left : prev

right : next

p = ~~p~~ ~~3~~ 5



3 5 8 10 12 15 25 30

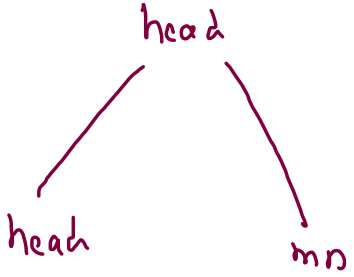
left :: prev

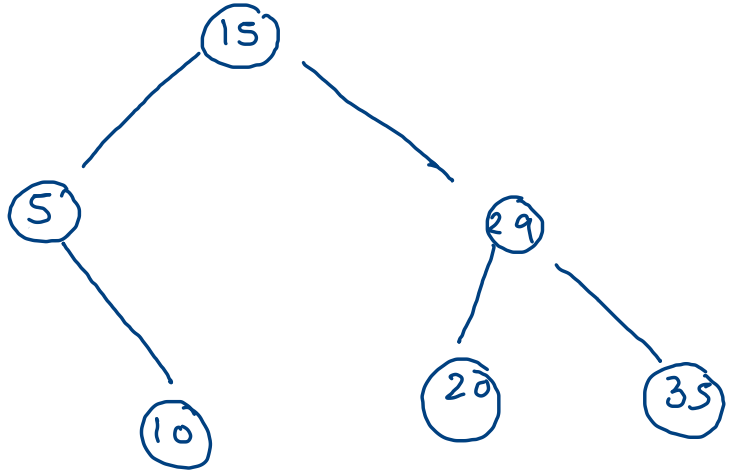
right :: next

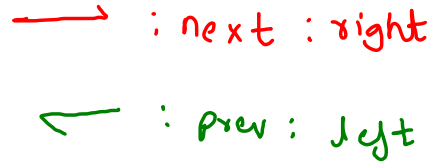
prev.right = node

node.left = prev

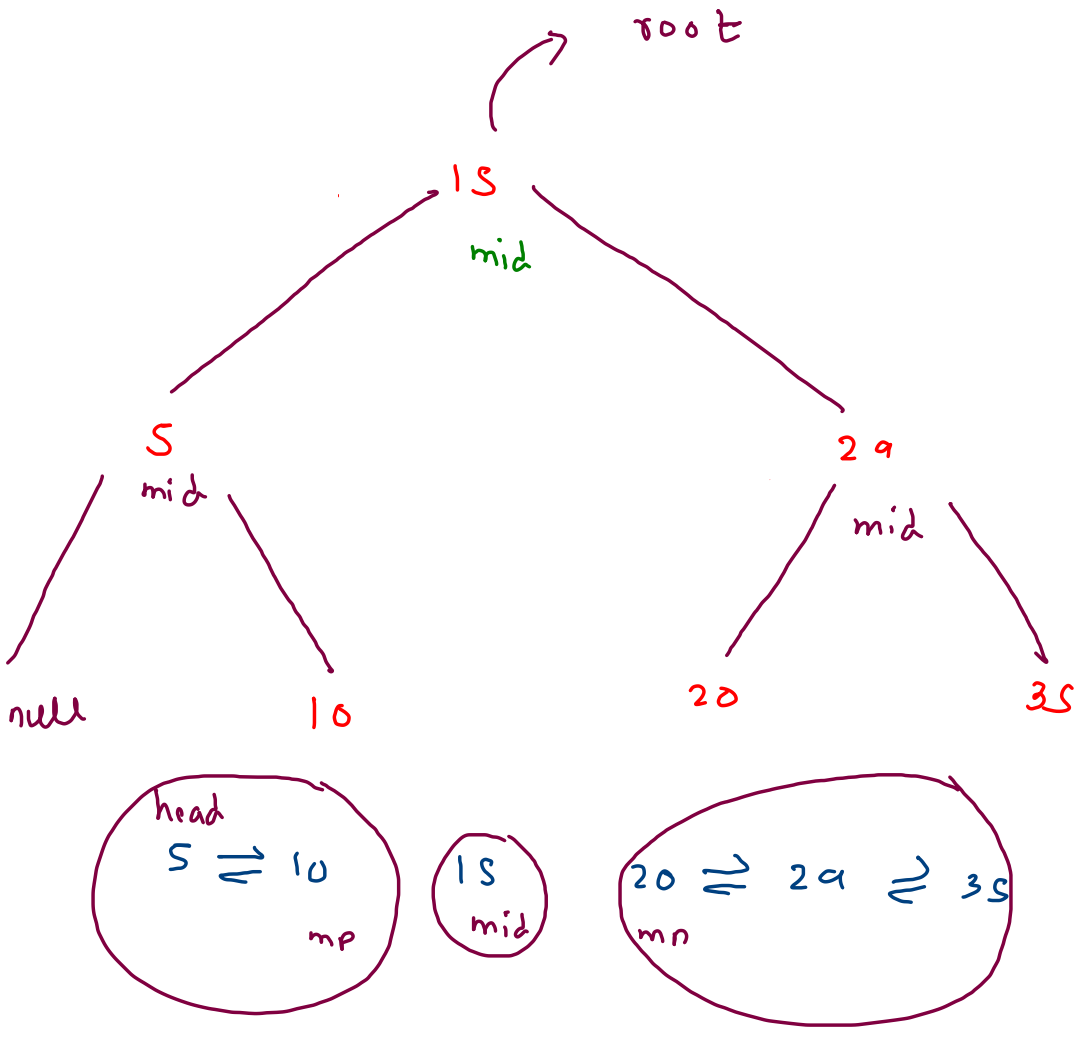
Convert Sorted Doubly Linked List To Binary Search Tree







```
// left : prev, right : next
public static Node SortedDLLToBST(Node head) {
    if(head == null || head.right == null) {
        return head;
    }

    Node mid = midNode(head);
    Node mp = mid.left;
    Node mn = mid.right;

    if(mp != null) {
        mp.right = mid.left = null;
    }
    mid.right = mn.left = null;

    mid.left = SortedDLLToBST(mp == null ? null : head);
    mid.right = SortedDLLToBST(mn);

    return mid;
}
```



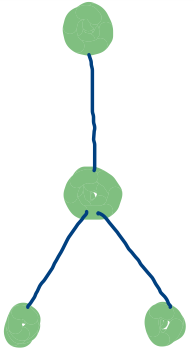
337. House Robber III

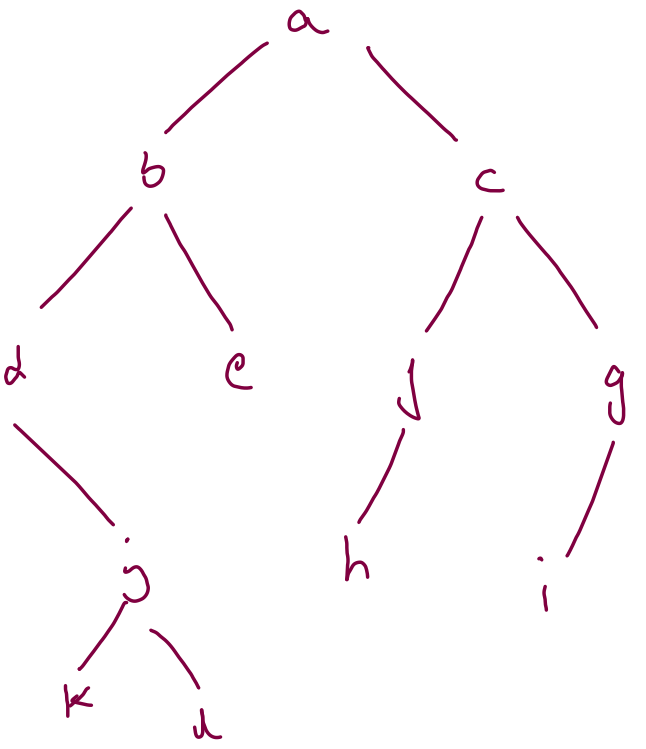
Medium 4908 78 Add to List Share

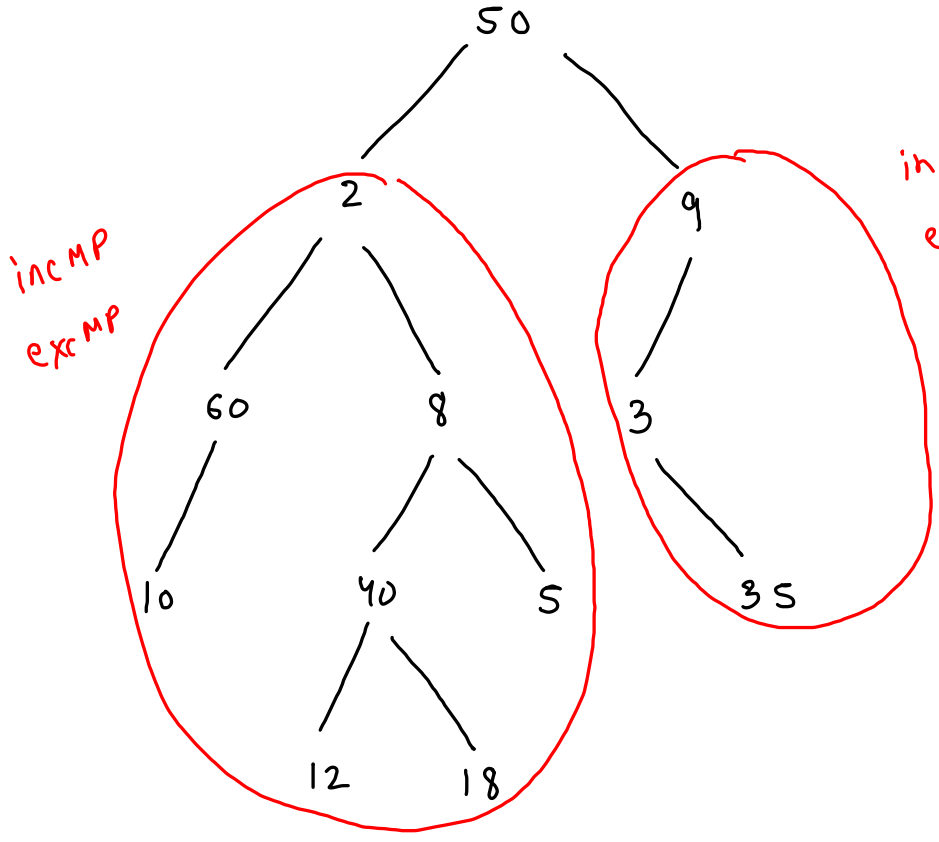
The thief has found himself a new place for his thievery again. There is only one entrance to this area, called `root`.

Besides the `root`, each house has one and only one parent house. After a tour, the smart thief realized that all houses in this place form a binary tree. It will automatically contact the police if **two directly-linked houses were broken into on the same night**.

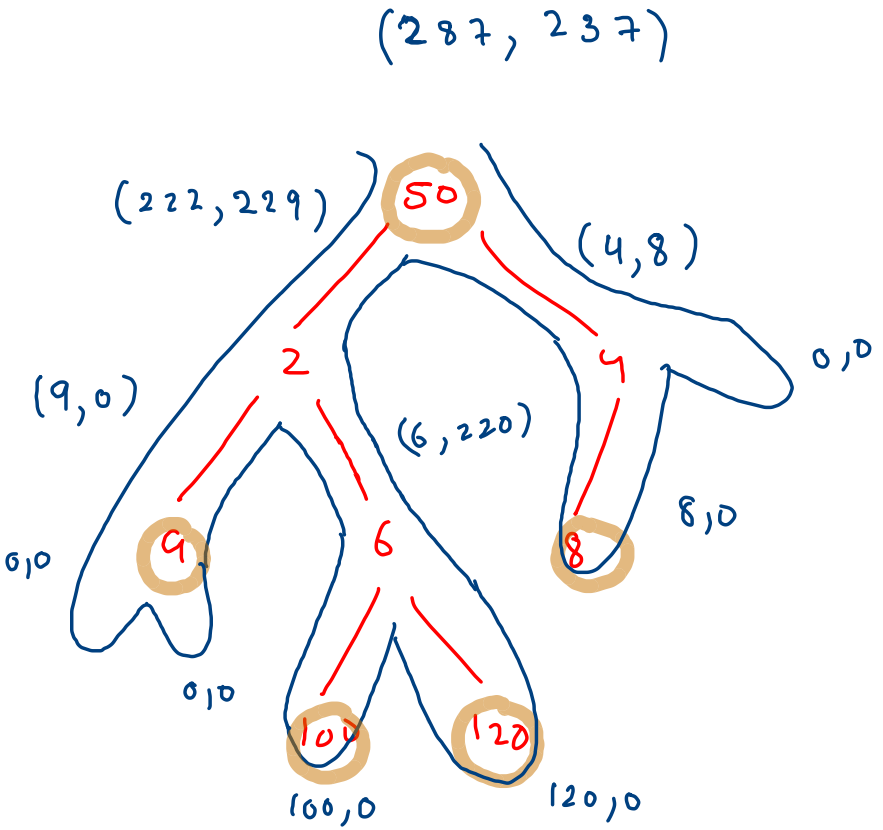
Given the `root` of the binary tree, return *the maximum amount of money the thief can rob* ***without alerting the police***.








```
lp = helper(node.left);
rp = helper(node.right);

incMP = node.val + lp.excMP + rp.excMP

excMP = Math.max(lp.incMP, lp.excMP)
      + Math.max(rp.incMP, rp.excMP)
```



(incMP, excMP)

```java
public static Pair helper(TreeNode node) {
    if(node == null) {
        return new Pair(0,0);
    }

    Pair lp = helper(node.left);
    Pair rp = helper(node.right);

    int incMP = node.val + lp.excMP + rp.excMP;
    int excMP = Math.max(lp.incMP,lp.excMP) + Math.max(rp.incMP,rp.excMP);

    return new Pair(incMP,excMP);
}
```