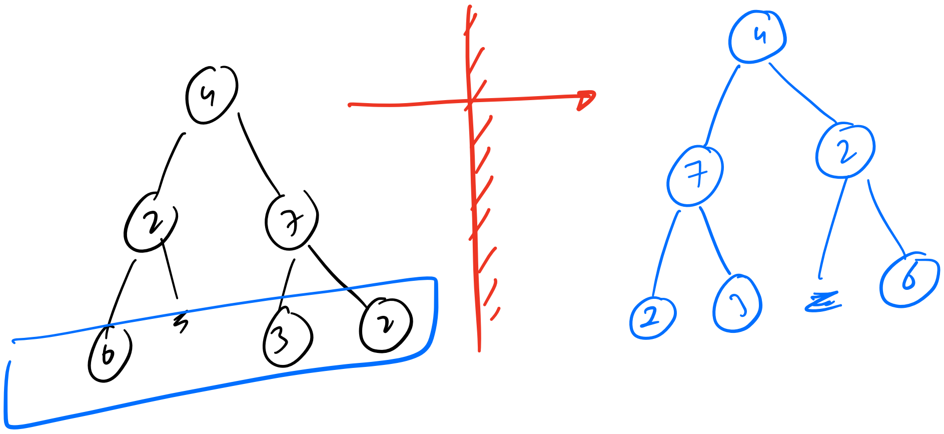
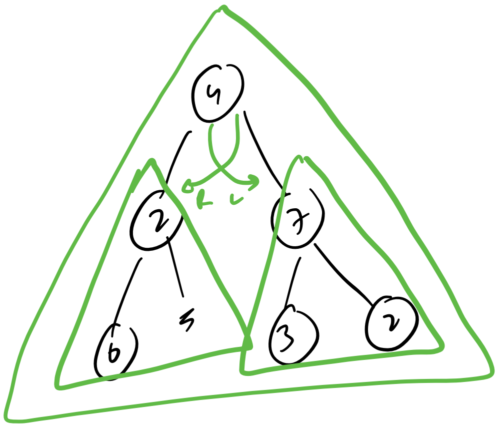
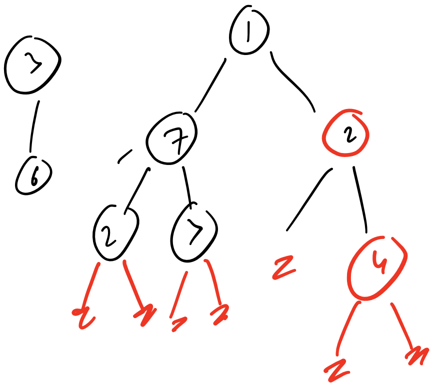
Q Given a B.T. Invert it!

∀ nodes swap the left & right child!





```
void invert(Node root) {
    if(root == NULL) ret;

    invert( root.left);
    ________.right_
    swap( root.left, root.right);
}
```

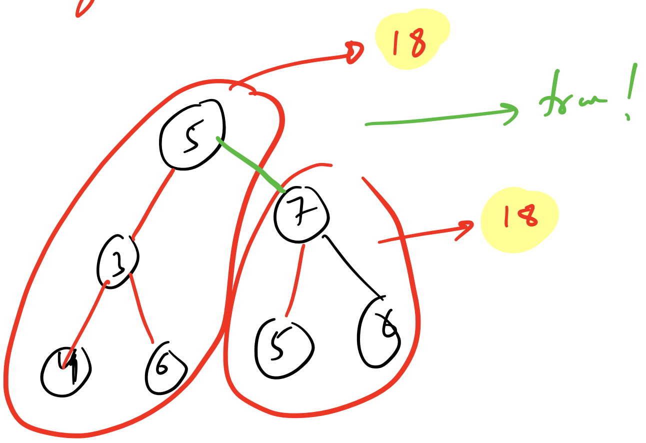


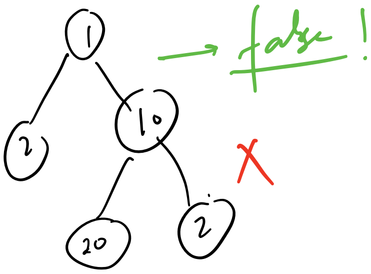
$\boxed{TC = O(N)}$

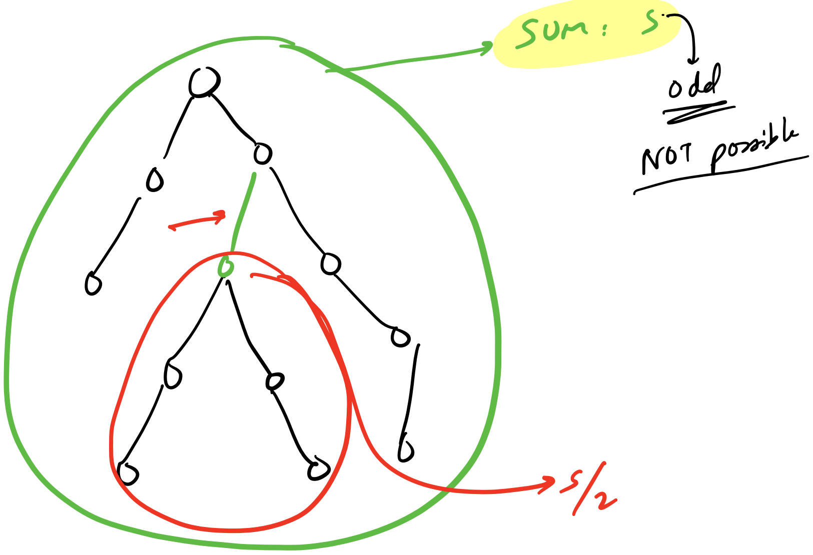
$\boxed{SC = O(H)}$

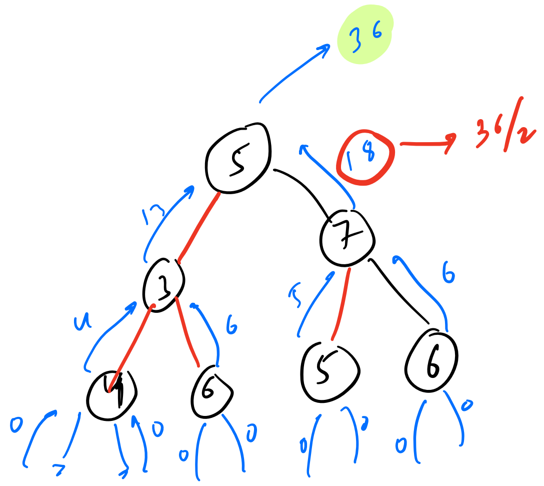
Q Given a B.T. Check if equal tree partition is possible?

→ partition the tree into two trees having exactly same sum of node values by removing 1 edge.









1. find sum

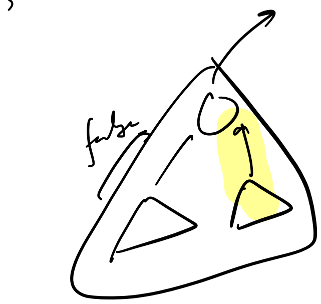
```
int   sum ( Node root) {
   if ( root == NULL) ret 0;
         ret   sum (root. left)
            + sum  (root. right)
            + root. data;

}
```

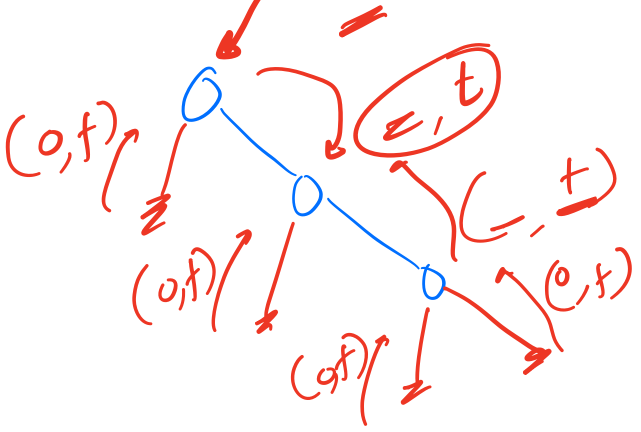
```
s = sum (root);
if ( s % 2  != 0)  ret false;
```

```
ret  check (root) . found ;
```

```
Info check(Node root){
    if(root == NULL){
        ret Info(0, false);

    }
    Info L = check(root.left);
    if( L.found == true){
        ret Info( 0, true);

    }
    Info R = check(root.right);
    int csum = L.sum + R.sum + root.data ;
    ret Info( csum, (R.found == true)||(csum == S/2)));

}
```

```
Info{
 int sum ;
 bool found;
}
```



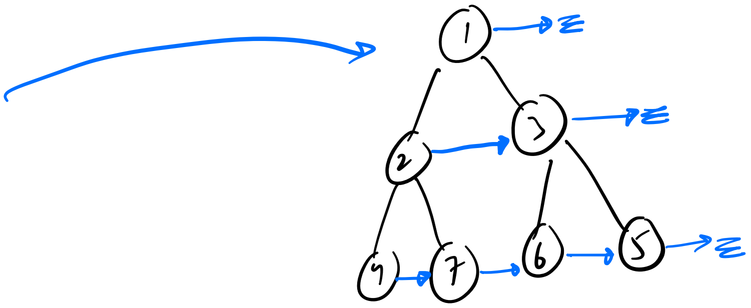


$\boxed{TC : O(N)}$

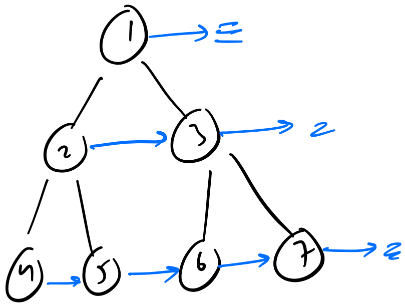
$\boxed{SC : O(H)}$

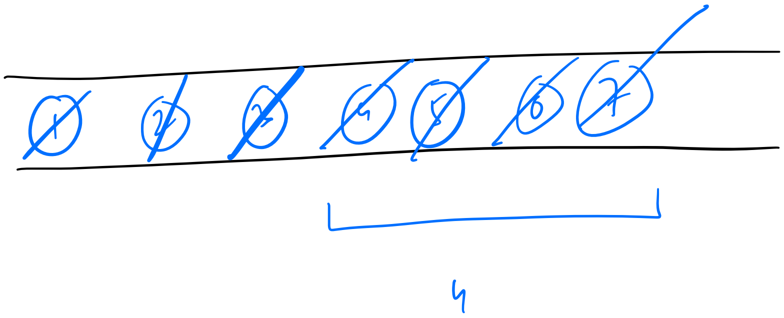
Q Given a perfect B.T.
Assign the next pointers of all the nodes to the next node on it's right

Node {
  Node left;
  ___ right;
  ___ next;
}



I) LOT





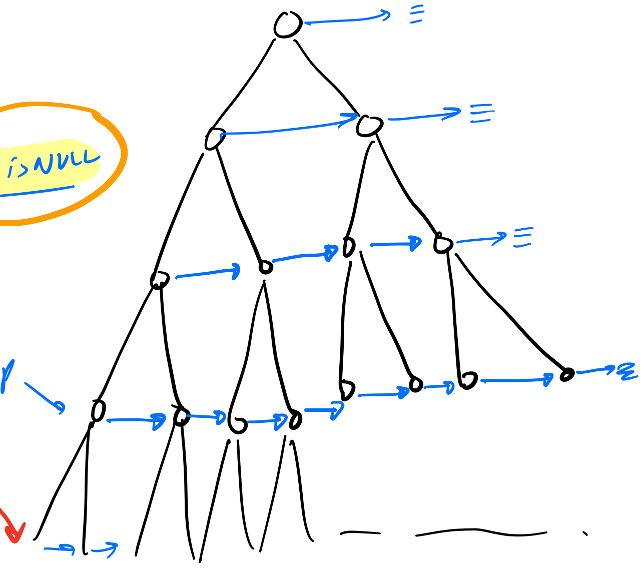
```
Queue < Node > q;
q. enqueue ( root);
while ( ! q. isEmpty()) {
    sz = q. size ();
    f ( i = 0; i< sz ; i++) {
        Node f = q. front();
        q. dequeue ();
        if( i==sz-1){
            f. next = NULL;
        }
        else {
            f. next = q. front();
        }
        if( f. left != NULL) {
            q. enque ( f. left);
        }
        if ( f. right != NULL) {
            q. enque ( f. right);
        }
    }
}
```

$\boxed{TC = O(N)}$

$\boxed{SC = O(\cancel{W})} \rightarrow N$

II

```
P1 = P.left
    p.left.next = p.right
    p.right.next = p.next.left    if isNULL
    p = p.next;
p = P1
```



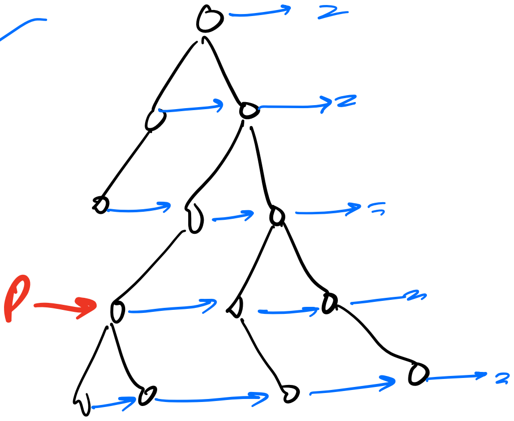
$\boxed{TC = O(N)}$

$\boxed{SC = O(1)}$

HW

Solve the above problem for a B.T.

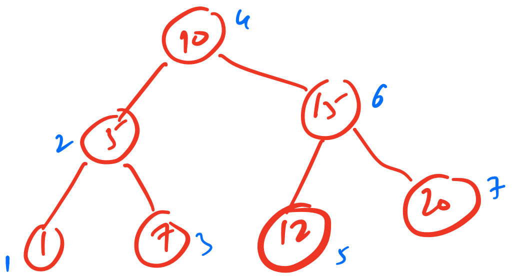
$\boxed{SC : O(1)}$



Q Given a BST. find the Kth smallest node!

K = 3 → 7

K = 6 → 15



INORDER of a BST is SORTED!

I)

```
     k
IN: 1 5 7 10 12 15 20
```

$\boxed{TC : O(N)}$

$\boxed{SC = O(N)}$

II

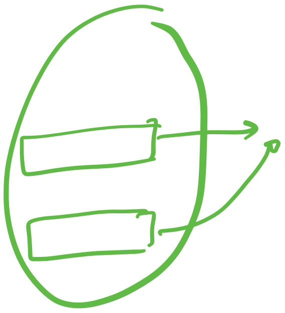
```
ct=0;
inorder ( root) {

    inorder (root.left);
    ct++;
    if(ct==k){
        print (root.data);
    }
    inorder (root.right);
}
```

$\boxed{TC = O(N)}$

$\boxed{SC = O(H)}$

III Morris →

ct=0

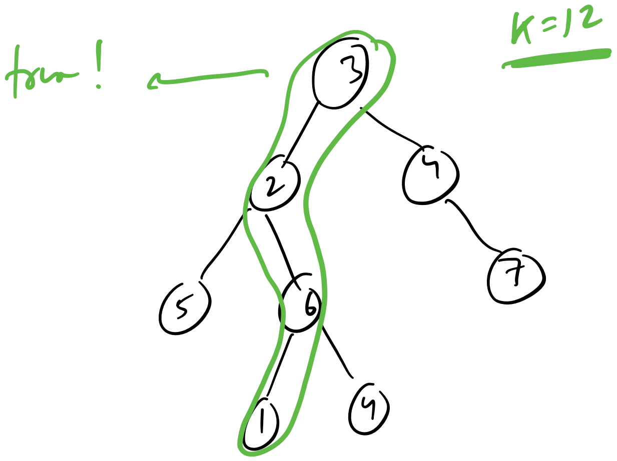


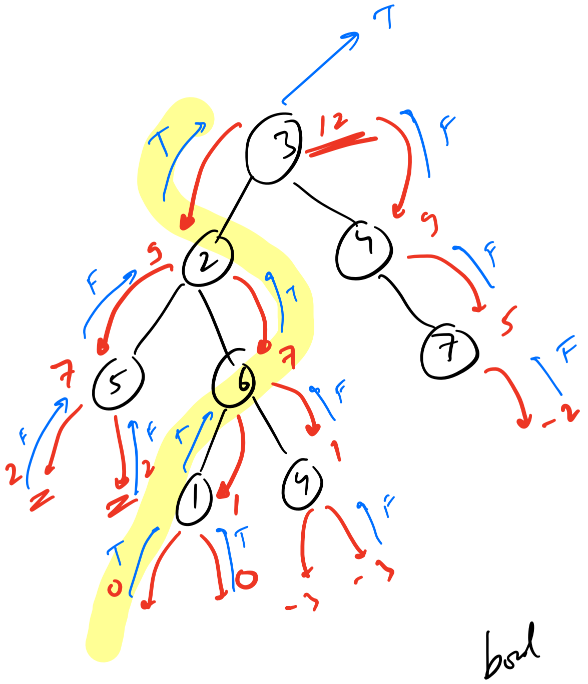
```
ct++
if(ct==k)
    print (node.data)
```

$\boxed{TC : O(N)}$

$\boxed{SC : O(1)}$

Q Given a B.T. Check if there is a root to leaf path with sum == K.





```
bool check ( Node root, int sum) {
    if ( root.left == NULL && root.right == NULL){
        if (sum == root.data ) ret true;
        ret false;
    }
    if (root == NULL) ret false;
    L = check (root.left, sum - root.data);
    if(L == true) ret true;
    ret check (root.right, sum - root.data);
}
```

TC = O(N)

SC = O(H)