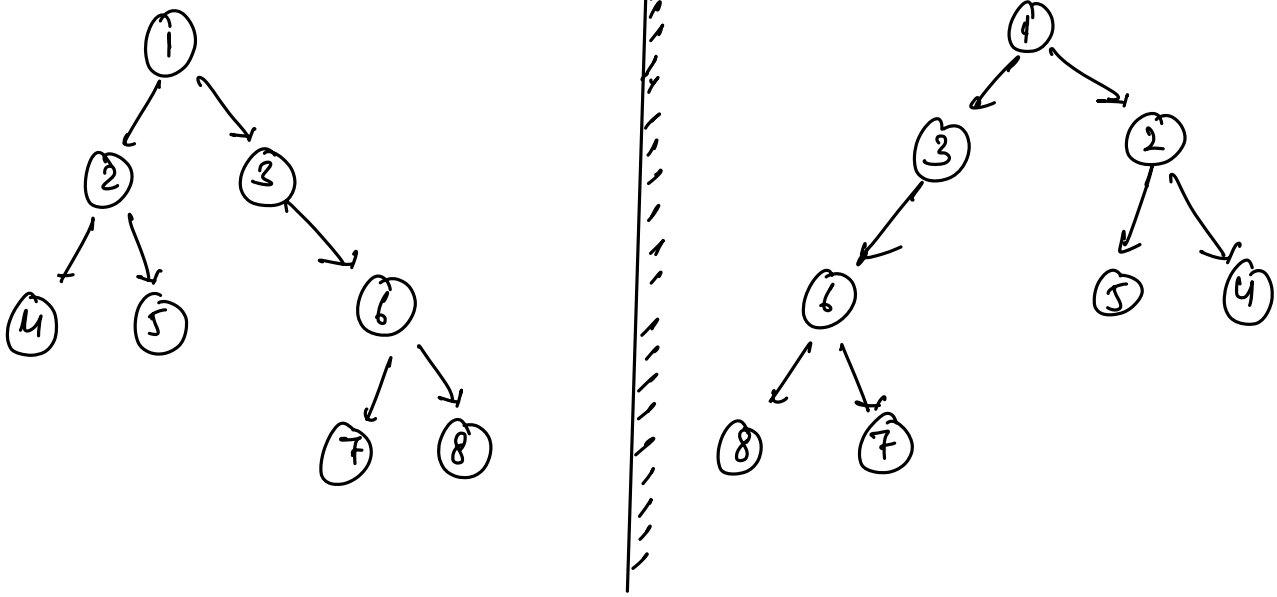
Today's content →

1. Invert Binary Tree
2. Equal Tree Partition
3. Next Pointer in Binary Tree
4. Root to Leaf Path Sum = K
5. Diameter of Binary Tree.

① Invert a Binary Tree

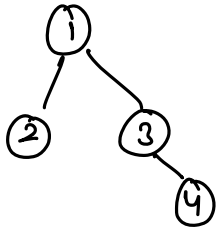


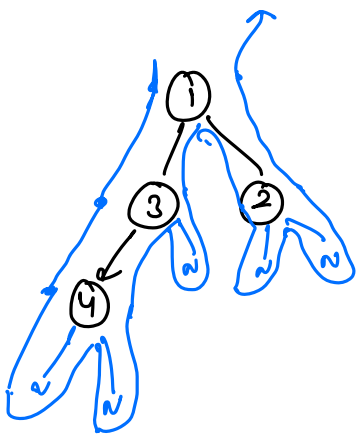
∀ nodes, swap left and right child

```
void invert(Node root){
    if(root == NULL){return;}
    //swapping left and right child
    Node temp = root.left;
    root.left = root.right;
    root.right = temp;

    invert(root.left);
    invert(root.right);
}
```

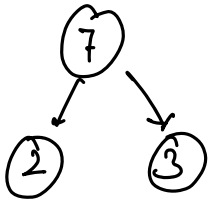
T.C → O(N)
S.C → O(Ht.)



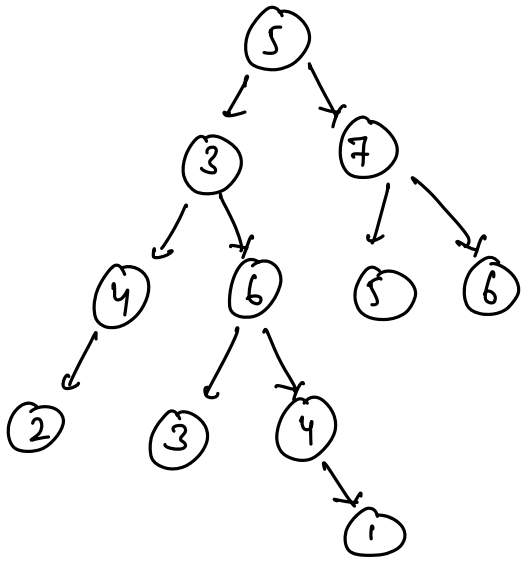


Equal Tree Partition

Check if it is possible to remove an edge from the given binary tree such that sum of resultant two trees is equal.



ans → false



ans → true

observations

① If total sum of binary tree is odd → return false
even → check & find ans.

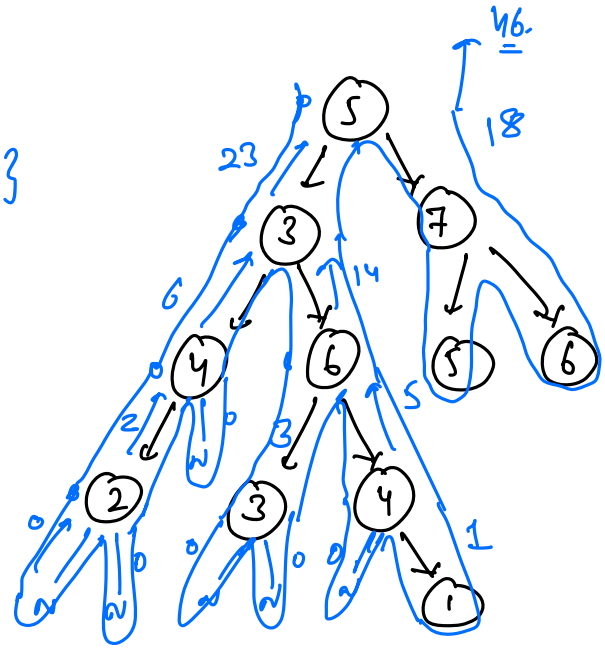
② One of the two trees will be a sub-tree.

We are looking for a sub-tree with sum = $\frac{\text{total Sum}}{2}$.

code...

```
int sum(Node root) {
    if(root == NULL) {return 0}
    return  sum(root.left) + sum(root.right) + root.val;
}
```

```
totalSum = sum(root);
if (totalSum %2 == 1) { return false }
boolean ans = false;
```



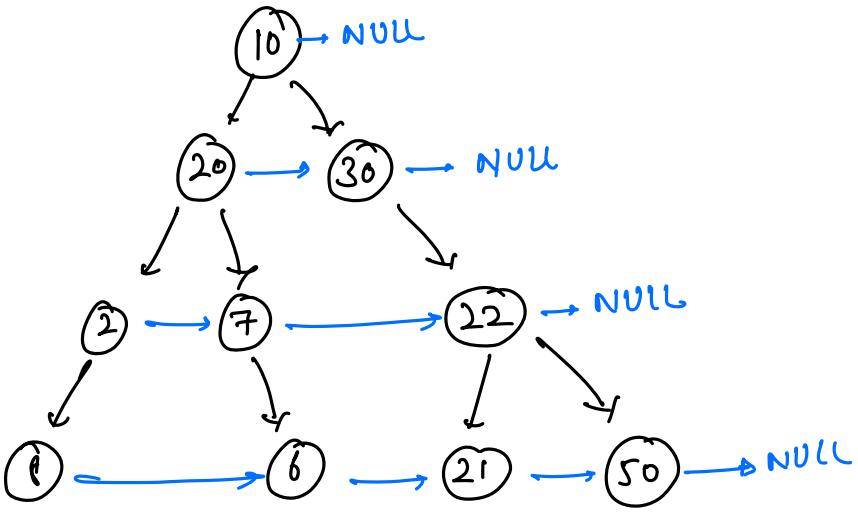

```
int sum2(Node root) {
    if(root == NULL) {return 0}
    lans = sum2(root.left);
    rans = sum2(root.right);
    if( lans == totalSum/2 || rans == totalSum/2){
        ans = true;
    }
    return  lans + rans + root.val;
}
```

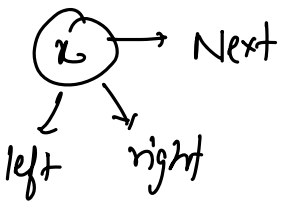
ans = ~~false~~ true

T.C → O(N)
S.C → O(Ht.)

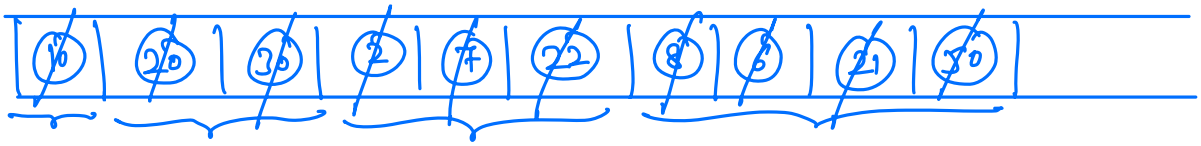
Next Pointer in Binary Tree →

Initially ∀ nodes, next pointers points to NULL. Update the next pointer to point to the next node in same level. (left to right)



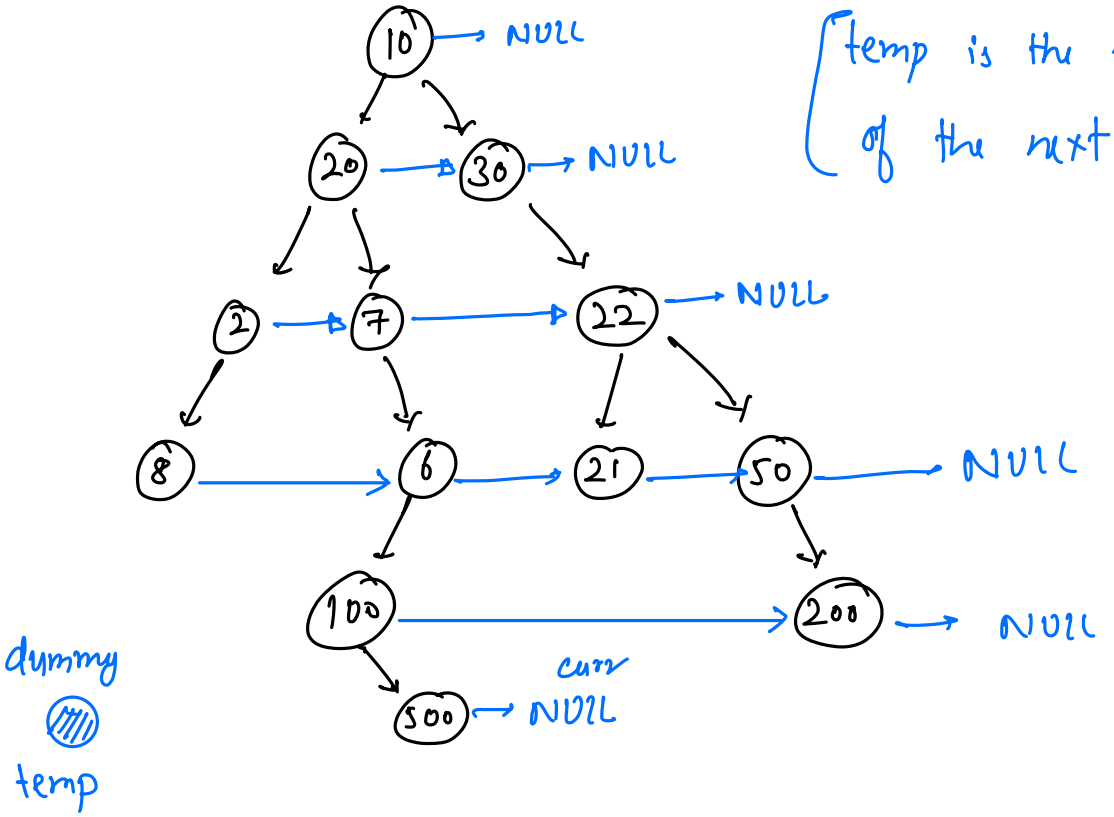


B.f idea → level order traversal.



similar to printing all the levels - linewise.

T.C → O(N)
S.C → O(N)



[temp is the first node of the next level.]

```
while( curr != NULL ){
    if (curr.left != NULL){
        temp.next = curr.left;
        temp = temp.next;
    }
    if (curr.right != NULL){
        temp.next = curr.right;
        temp = temp.next;
    }
    curr = curr.next;
    if (curr == NULL){
        curr = dummy.next;
        dummy.next = NULL;
        temp = dummy;
    }
}
```

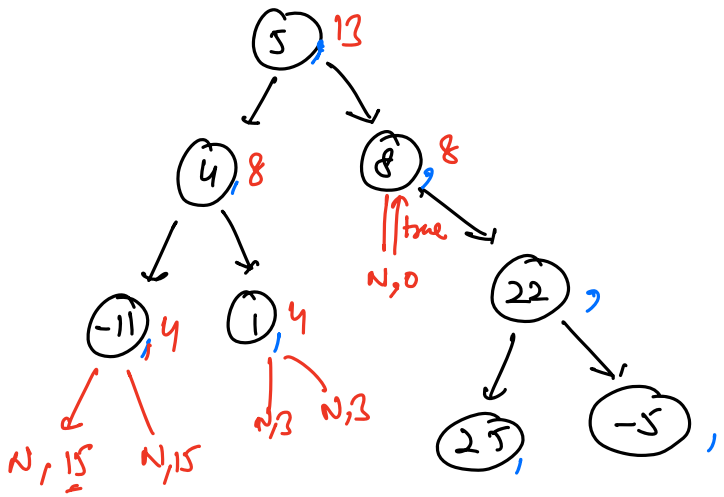
[T.C → O(N)
S.C → O(1)]

Q: Check if root to leaf path sum equals to K.

K=60 Yes.

K=50 false

K=30 true



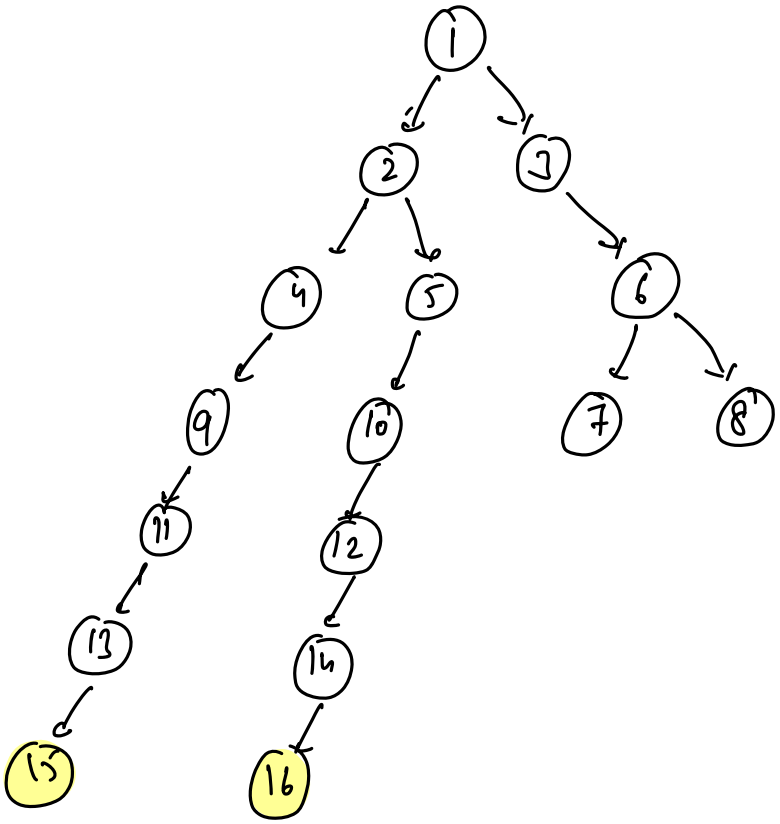

```
boolean check (Node root, int K){
    if( root == NULL) { return false }
    if( root.left == NULL && root.right == NULL) {
        if (K == root.val) return true;
        else return false;
    }
    return check(root.left, K-root.val) ||
           check(root.right, K-root.val) ;
}
```

#dry-run

[T.C → O(N)
S.C → O(Ht)]

Diameter Of Binary Tree

↳ Maximum distance (in edges) b/w any two nodes in a binary tree.



ans = 10

At every node, consider $ht_{l.st} + ht_{r.st} + 2$

diameter = 0

```
int height(Node root) {
    if (root == NULL) { return -1 }
    lht = height(root.left);
    rht = height(root.right);
    diameter = Max(diameter, lht + rht + 2);
    return Max(lht, rht) + 1;
}
```

T.C → O(N)
S.C → O(Ht.)

→ Dry-run

L.L + Trees + Hashmap. + Stack

⇒ Class & objects
⇒ edge-cases
⇒ recursion