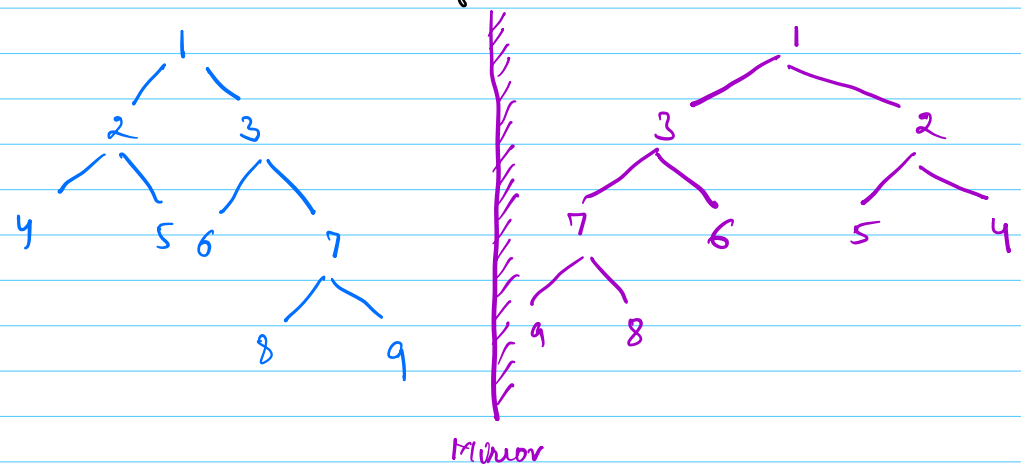
7/8/2023

Problems on Trees

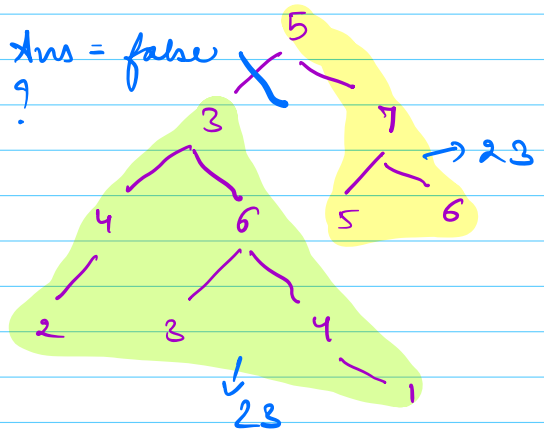
Q1 Invert the given binary tree.




```
Node invert (root) {

    if (root == null) return null;
    temp = root.left;
    root.left = invert (root.right)

    root.right = invert ( temp );

    return root;

}
```

TC: O(N)
SC: O(H)

Q2 Check if it is possible to remove an edge from the given binary tree such that the sum of resultant 2 trees is equal.



① Total sum → odd, ans = false
↳ even → ?

TC: O(N), SC: O(H)

② Check if there is a subtree with sum = totalSum/2

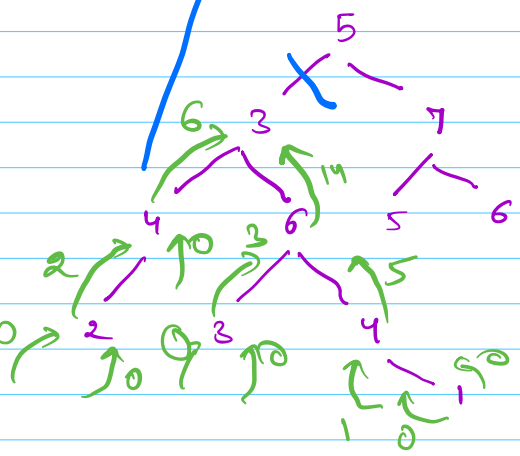
TC: O(N), SC: O(H)

ans = false ②

```
int sum(root, total){
    if (root == null) return 0;
    s = root.data + sum(root.left, total)
                  + sum(root.right, total);
    if (s*2 == total) ans = true;
    return s;
}
```

```
int sum(root) {                    ①
    if (root == null) return 0;
    return root.data +
    sum(root.left) +
    sum(root.right);
}
```

```
int lSum = sum(r.left);
int rSum = sum(r.right);
s = lSum + rSum + r.data;
```



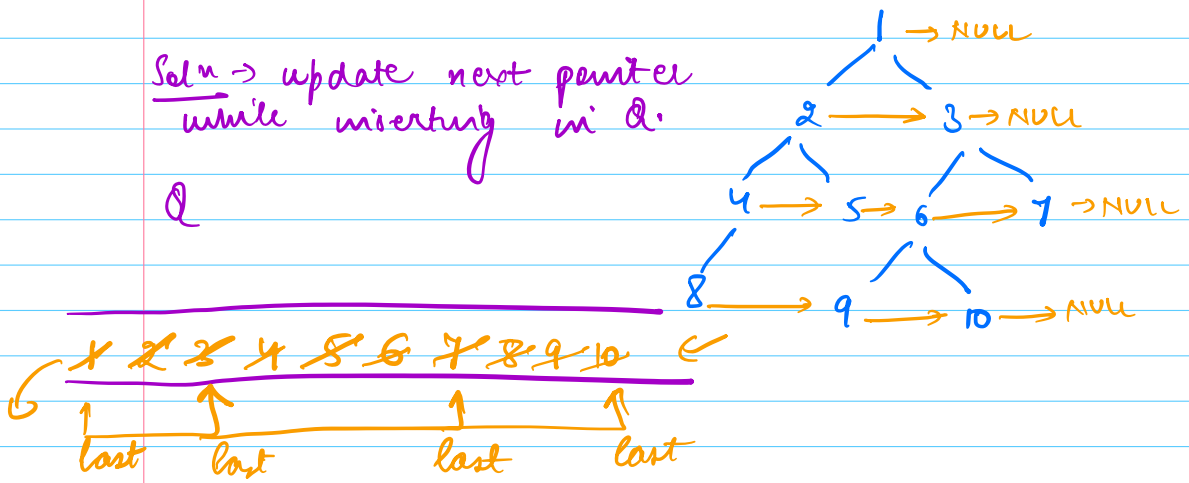
Q3 Next pointer in Binary tree

Initially if nodes, next pointer = null, data → next

Update each node's next pointer to point to the next node at same level.

left → right

Soln → update next pointer while inserting in Q.

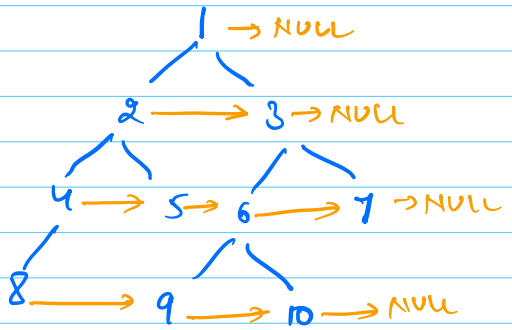


Q

TC: O(N)

SC: O(N) → O(1)

```
void connect(root) {
    curr = root;
    first = root;

    while (first != null) {
        prev = null;
        curr = first;
        first = null;
        while (curr != null) {
            if (curr.left != null) {
                if (prev == null) {
                    prev = curr.left;
                    first = prev;
                } else {
                    prev.next = curr.left;
                    prev = prev.next;
                }
            }
            if (curr.right != null) {
                if (prev == null) {
                    prev = curr.right;
                    first = prev;
                } else {
                    prev.next = curr.right;
                    prev = prev.next;
                }
            }
            curr = curr.next;
        }
    }
}
```

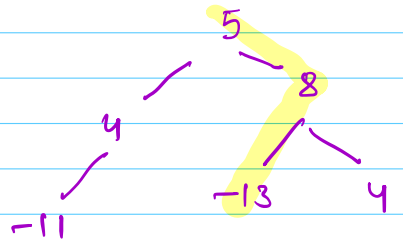


curr	first	prev
1	~~1~~ null	null
	2	~~2~~ 3
2	NULL	NULL
	4	~~4~~ 5
3		~~6~~ 7
4	NULL	NULL
	8	8
5		
6		~~4~~ 10
7		

TC: O(N)
SC: O(1)

Q4 Check if the given binary tree has root to leaf path sum = K.

K = 0, ans = true
K = 2, ans = false




```
boolean check ( root, K) {
    if (root == null) return false;
    if (root.left == null && root.right == null)
            return (K == root.data);
    return check (root.left, K - root.data) ||
            check (root.right, K - root.data);
}
```

TC: O(N)
SC: O(H)

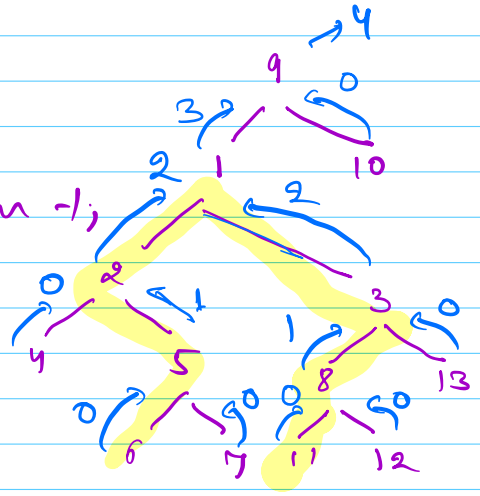
Subtree sum → post
height → post

path sum → pre

Q5. Find diameter of binary tree.

edges in longest path.

```
ans = 0;
int height(root) {
    if (root == null) return -1;
    L = height(root.left)
    R = height(root.right)
    ans = max(ans, L+R+2);
    return max(L, R)+1;
}
```



TC: O(N)
SC: O(H)