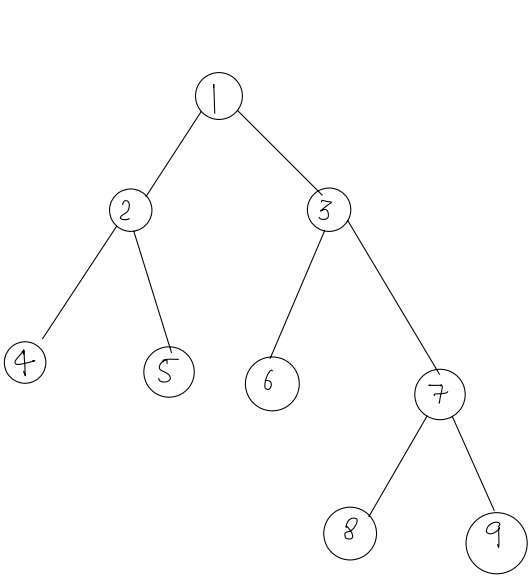
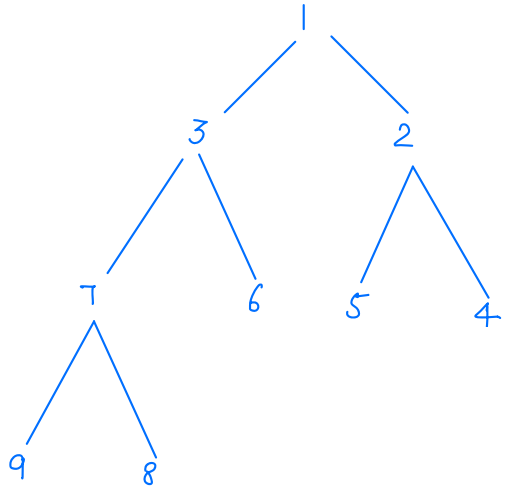
Lecture :- Trees 5

Agenda

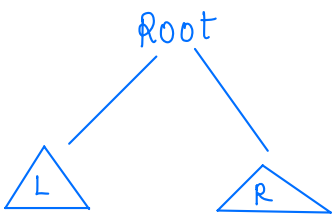
- Invert a binary tree
- Equal tree partition
- Next pointer in a binary tree [Hard]
- Root to leaf path sum = k
- Diameter of binary tree.

Qu Invert a binary tree

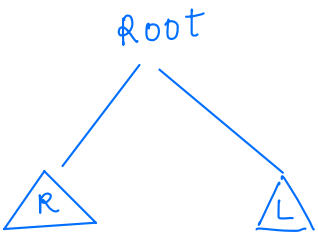


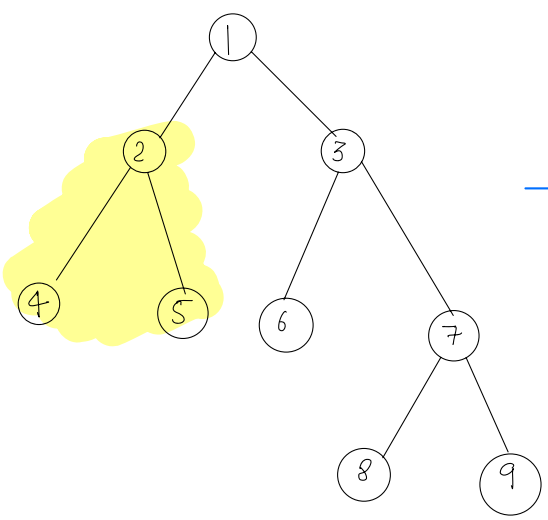


Approach

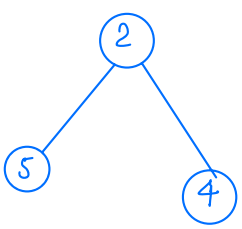


invert

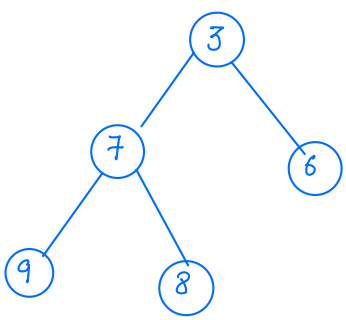




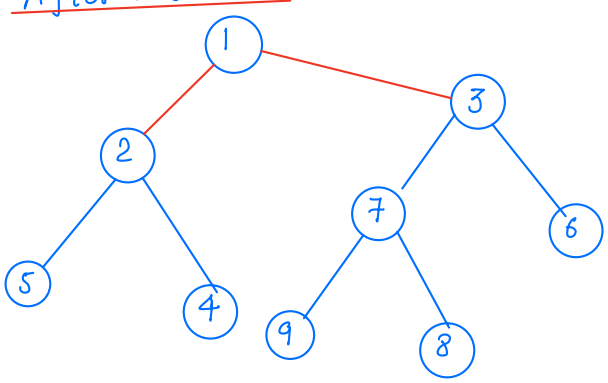
1. Rec (root.left)



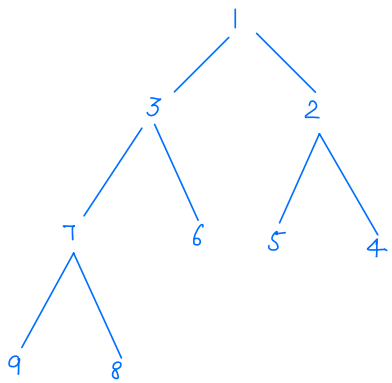
2. Rec (root.right)



After 1 and 2



swap L & R

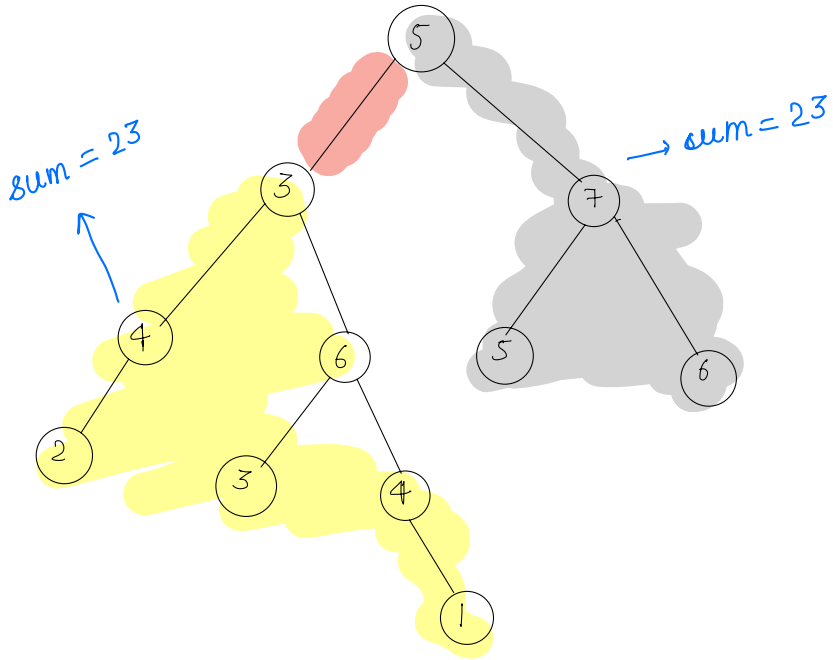


Pseudocode

```
void invert(root) {
    if(root==null) {
        return;
    }
    invert(root.left);
    invert(root.right);
    temp = root.left;
    root.left = root.right;
    root.right = temp;
}
```

TC: O(n)
SC: O(h)

Qu Given a binary tree. check if it is possible to remove an edge from binary tree such that sum of resultant 2 trees are equal. [Medium - hard]



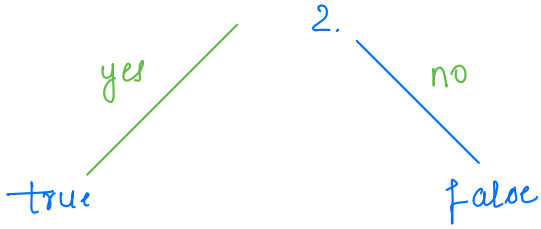
true

Observation

if sum of tree is odd —— return false;

even —— try doing something

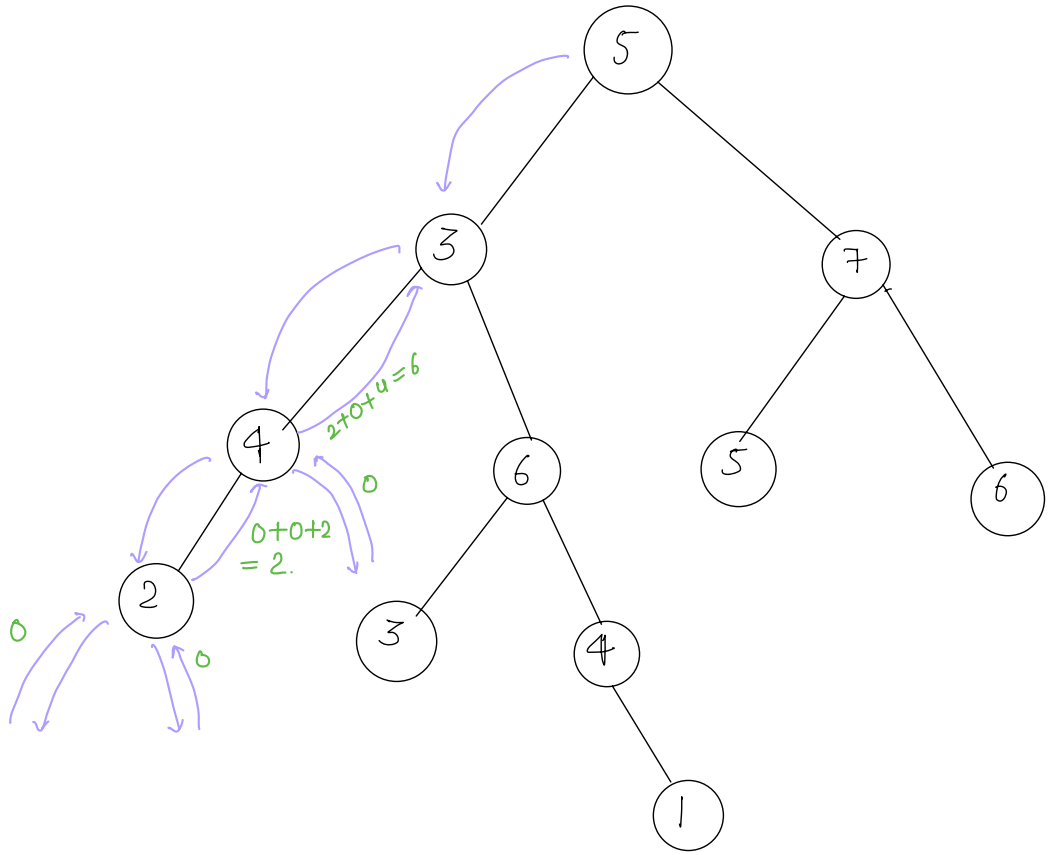
Modified problem statement Check if there is any subtree with sum = total sum of tree / 2.

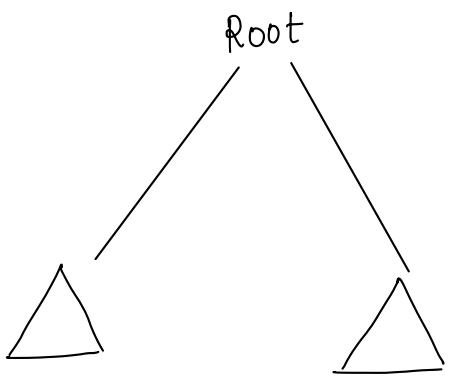


Approach

sum = 46

check ⟶ $\frac{sum}{2}$ = 23





ls = sum of l.s
rs = sum of rs
return ls + rs + root.data

Pseudocode

```
boolean ans = false;
boolean check(root) {
    sum = getsum(root);
    if (sum % 2 != 0) {
        return false;
    }
    helper(root, sum/2);
    return ans;
}

int helper(root, sum) {
    if (root == null) {
        return 0;
    }
    ls = helper(root.left, sum);
    rs = helper(root.right, sum);
    if (ls == sum || rs == sum) {
        ans = true;
    }
    return ls + rs + root.data;
}
```

Qu Next pointer in a binary tree

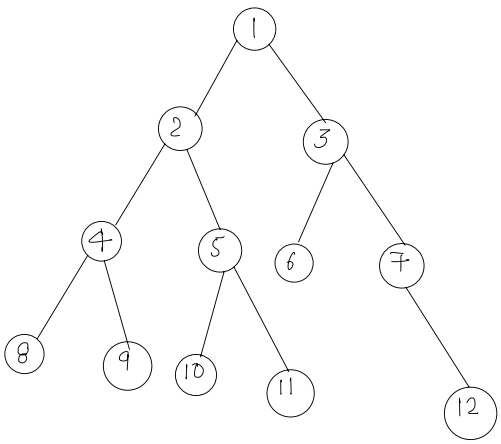
Initially each node next pointer points to null. Update each pointer next pointer to point to next node in same level.

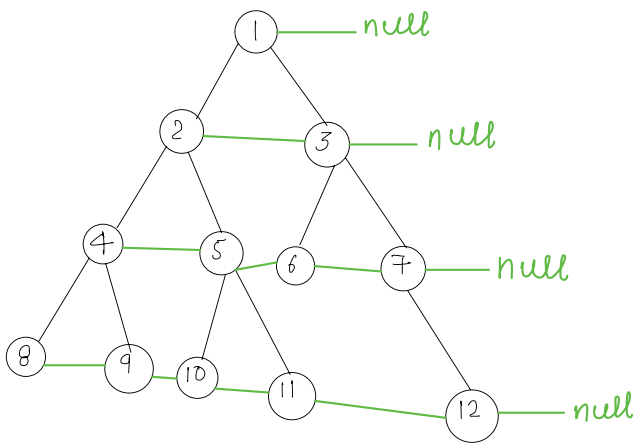
Current structure

```
class TreeNode {
    int data;
    TreeNode left;
    TreeNode right;
}
```

Question structure

```
class TreeNode {
    int data;
    TreeNode left;
    TreeNode right;
    TreeNode next;
}
```



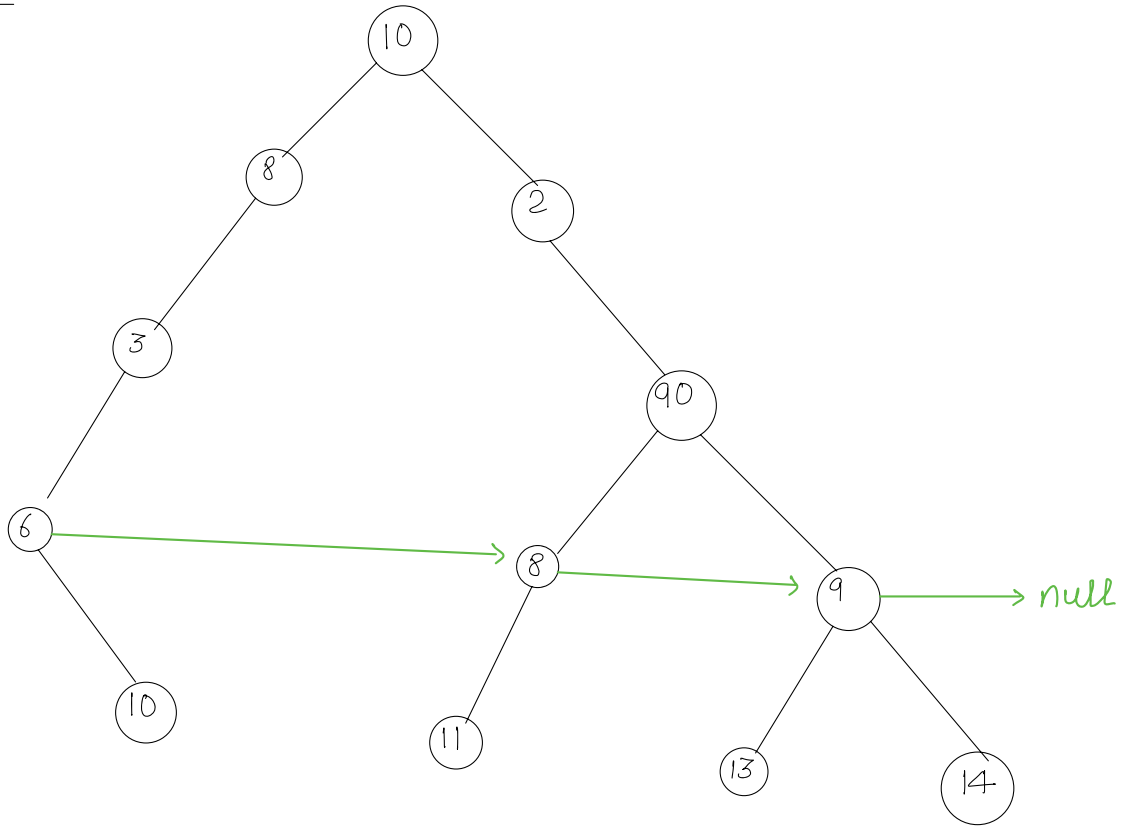


5 — left = 10
right = 11
next = 6

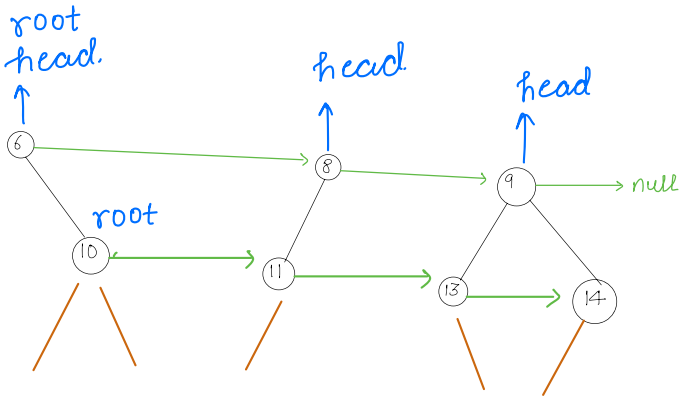
Approach1 Level order traversal.

TC: O(n)
SC: O(h)

Approach SC: O(1)



Pseudo code

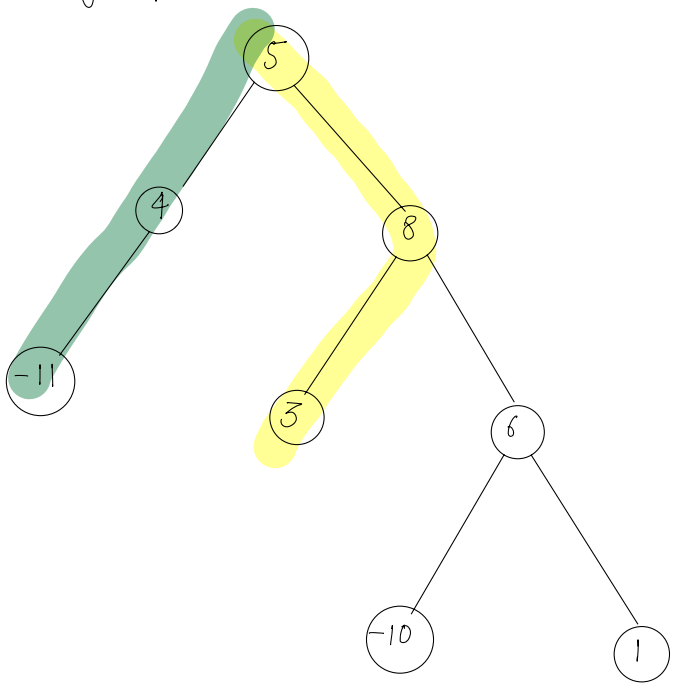



```
void connect(TreeNode root) {
    temp = null;
    if(root == null) {
        return;
    }
    root.next = null;
    while(root != null) {
        head = root;
        while(head != null) {
            if(head.left != null) {
                if(head.right != null) {
                    head.left.next = head.right;
                } else {
                    head.left.next = getNextRight(head);
                }
            }
            if(head.right != null) {
                head.right.next = getNextRight(head);
            }
            head = head.next;
        }
        if(root.left != null) {
            root = root.left;
        }
        else if(root.right != null) {
            root = root.right;
        } else {
            root = getNextRight(root);
        }
    }
```

TC: O(n)
SC: O(1)

Break: 8:27 - 8:37

Qu Given a binary tree and integer K, determine if exists a root to leaf path in tree such that adding up all the node values along path equals K.

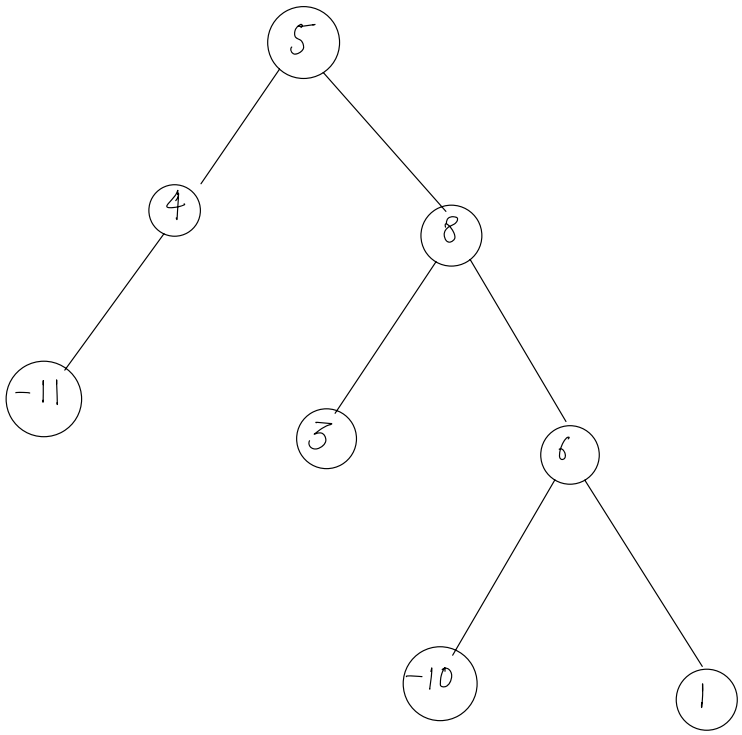


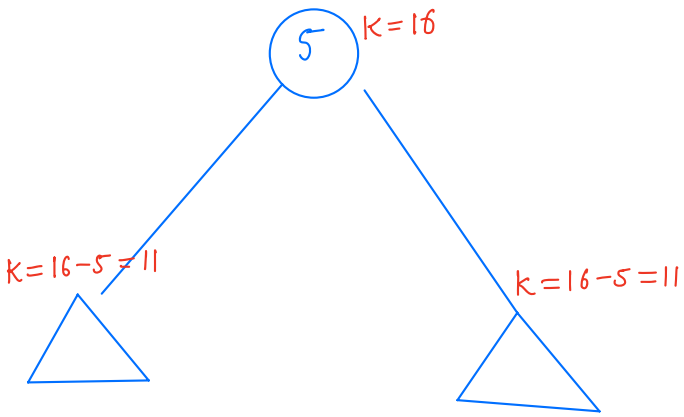
K=16 — true.

K=-2 — true

K=13 — false

Approach





Pseudocode

```
boolean check(TreeNode root, int k) {
    if (root == null) {
        return false;
    }
    if (root.left == null && root.right == null) {
        if (root.data == k) {
            return true;
        }
        return false;
    }
    left = check(root.left, k - root.data);
    if (left == true) {
        return true;
    }
    return check(root.right, k - root.data);
}
```

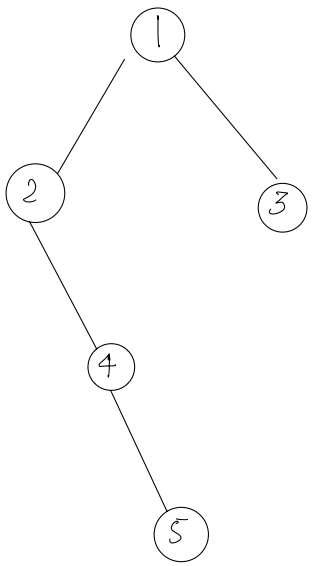
5

TC: O(n)
SC: O(h)

Qu Given a binary tree, find the longest path b/w any 2 nodes in a tree. This path may or may not pass through root. ⟶ Diameter of tree

Height of binary tree [in term of edges]

Edges



height (nodes) = 4

height (edges) = 3

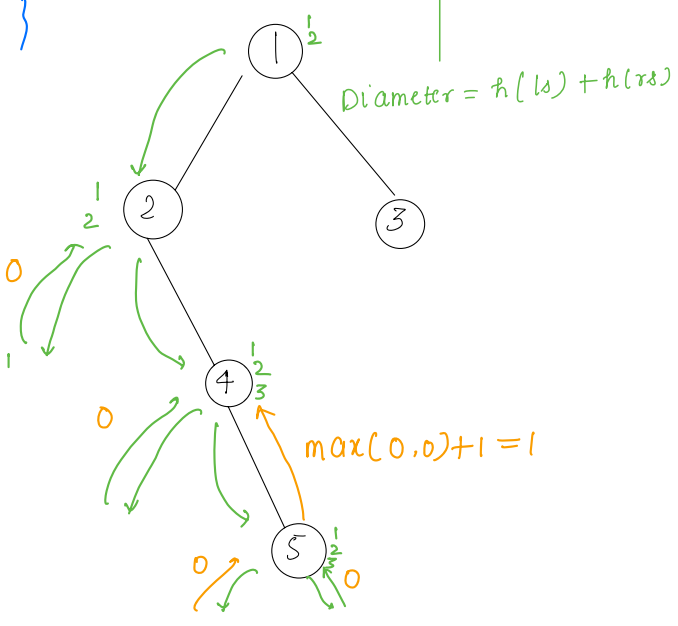
Edges

```
int height(root) {
1  if (root == null) {
       return -1;
   }
2  l = height(root.left)
3  r = height(root.right);
4  return max(l,r) + 1;
}
```

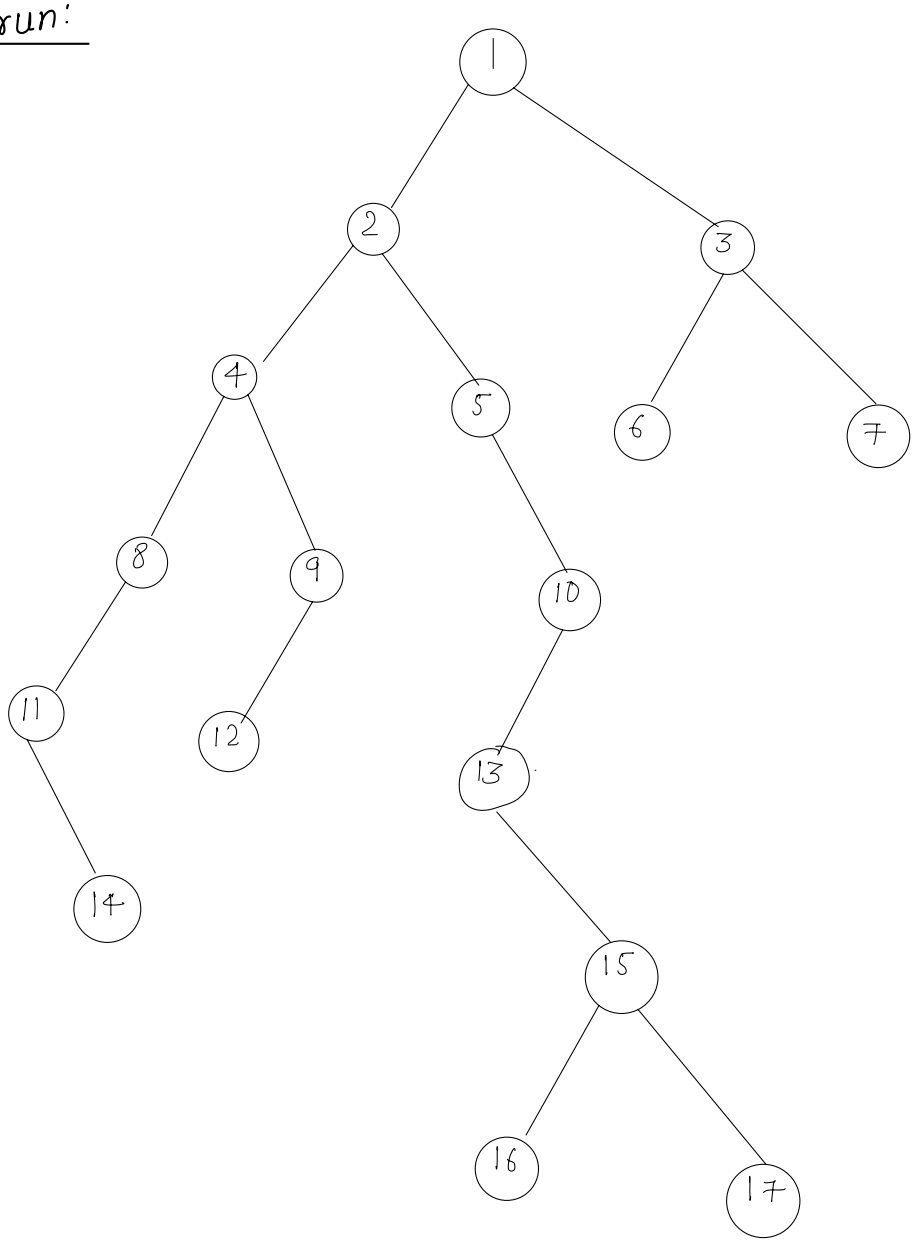
Diameter = h(ls) + h(rs) + 2.

Nodes

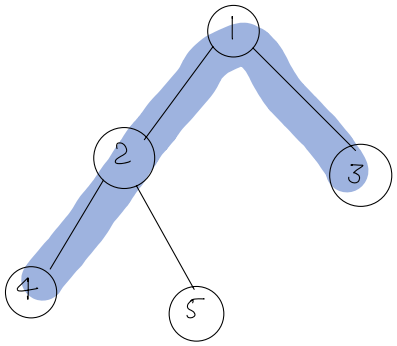
```
int height(root) {
1  if (root == null) {
       return 0;
   }
2  l = height(root.left)
3  r = height(root.right);
4  return max(l,r) + 1;
}
```



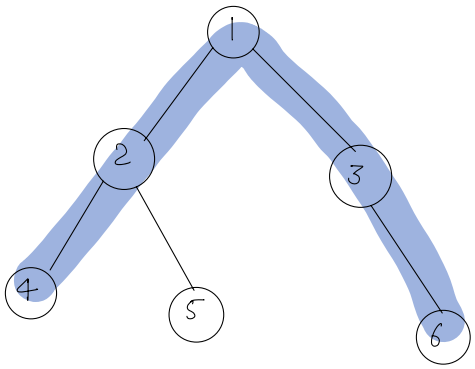
Dry run:



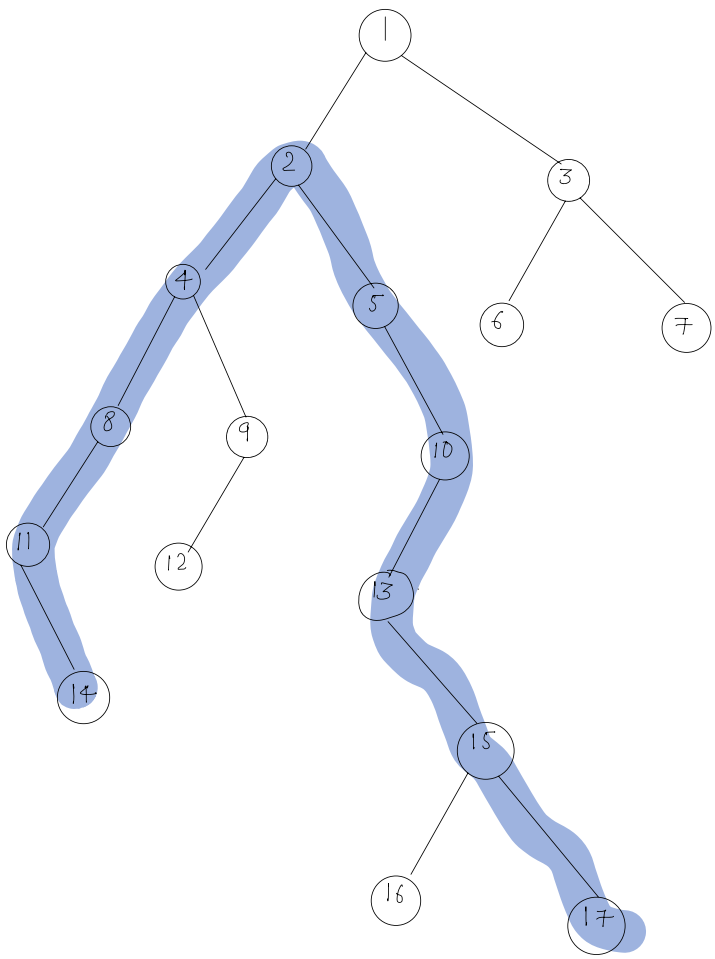
Diameter of binary tree



ans = 3

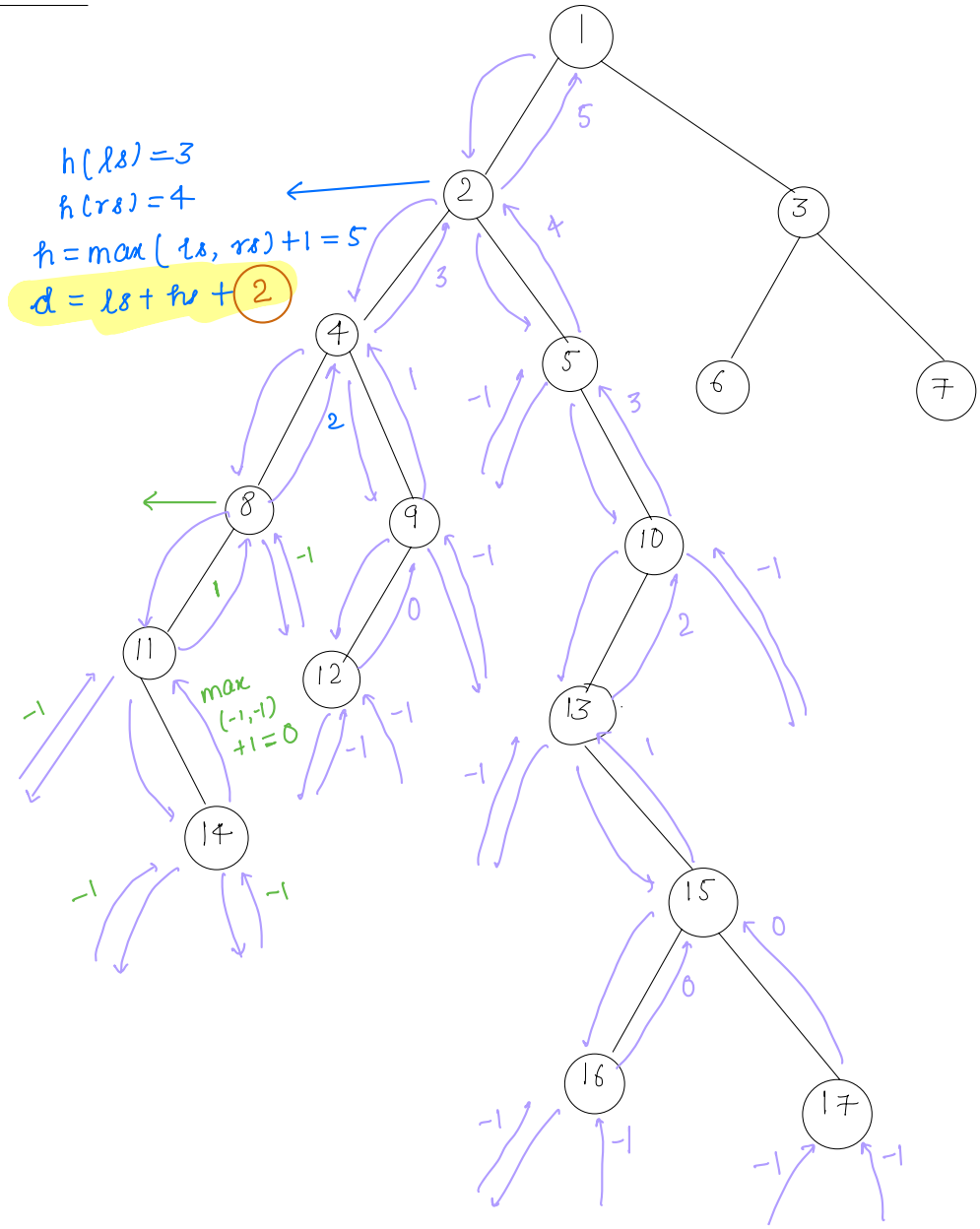


ans = 4



ans = 9

Approach



Pseudocode

```
int diameter = -1;
int height (TreeNode root) {
    if (root == null) {
        return -1;
    }
    l = height (root.left)
    r = height (root.right);
    diameter = max(diameter, lh+rh+2);
    return max (l, r) +1;
}
```

TC: O(n)

SC: O(h)

Thankyou