Trees 5: Problems on Trees

"You can't be that kid standing at the top of the waterslide, overthinking it. You have to go down the chute."

TINA FEY

* Content

01. Invert Binary Tree

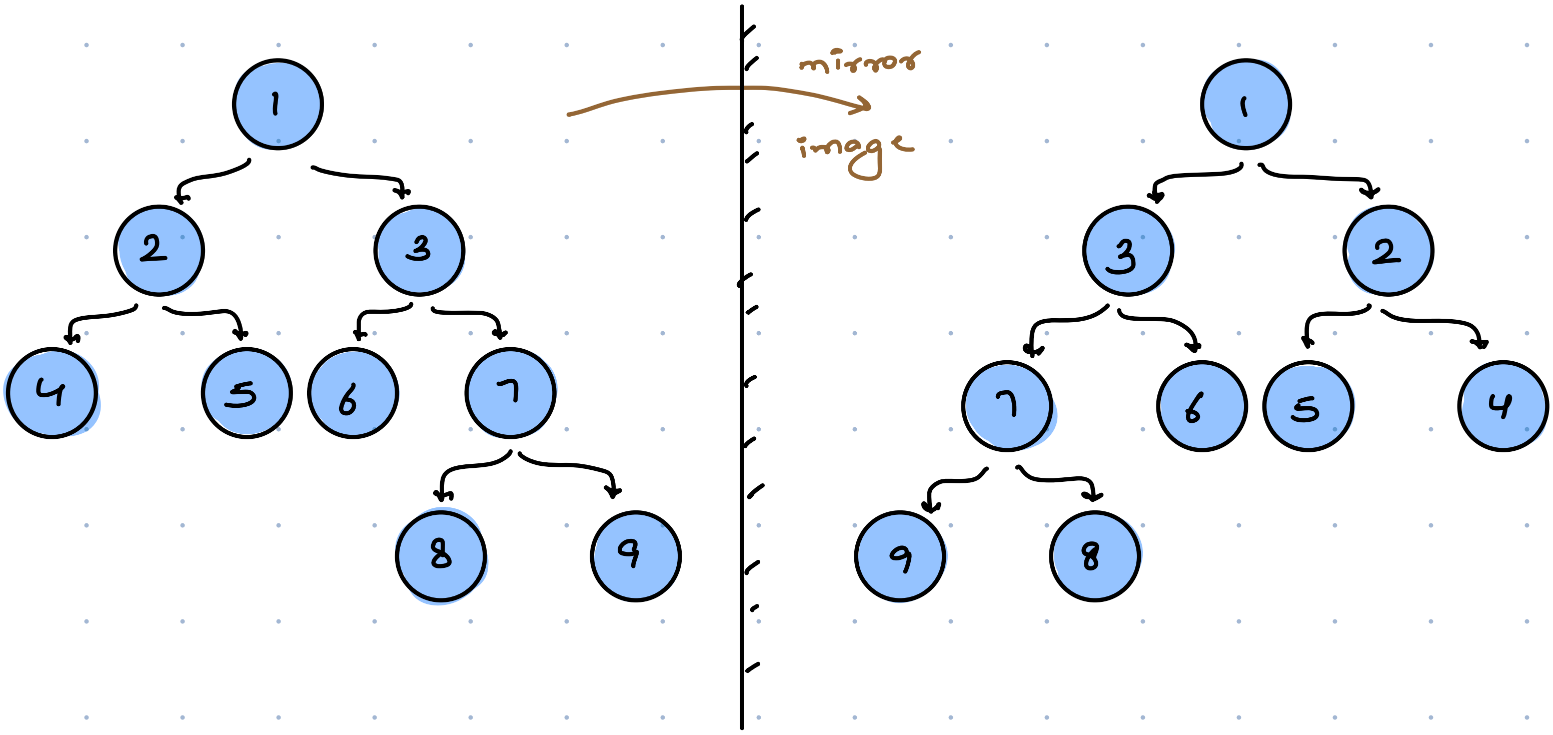
02. Equal tree partition

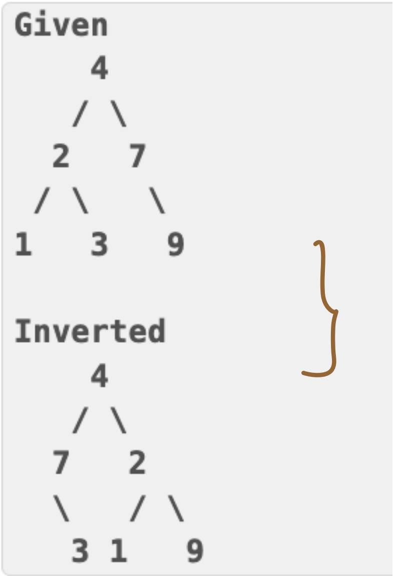
03 Next pointer in Binary tree

04. Root to leaf path sum = k

05. Diameter of Binary tree

* Invert the Binary Tree





```
Given
    4
   / \
  2   7
 / \   \
1   3   9

Inverted
    4
   / \
  7   2
   \  / \
   3 1   9
```

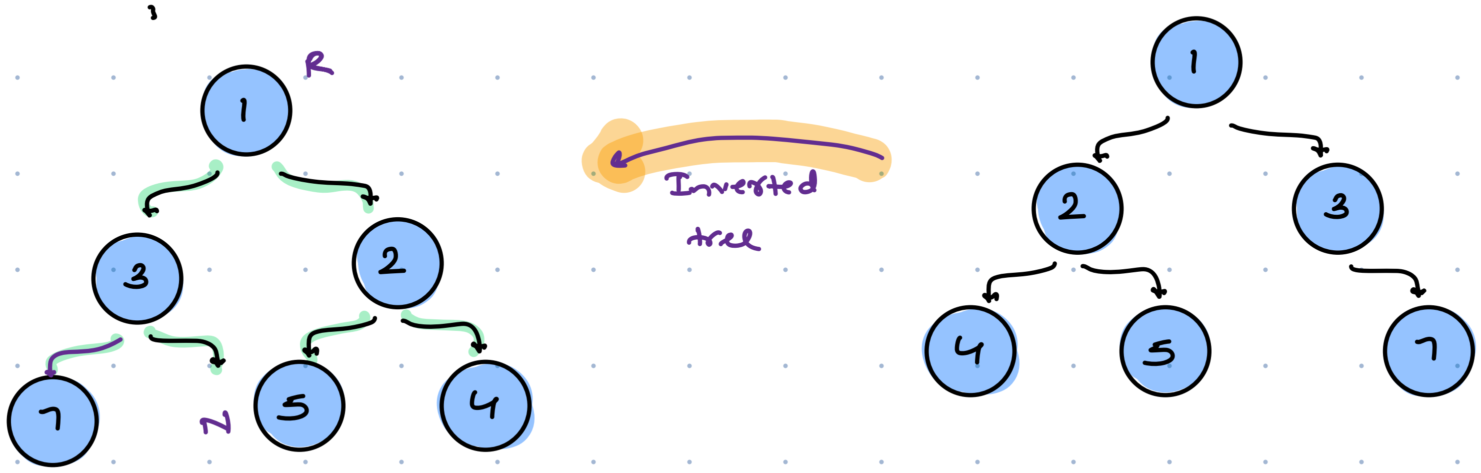
No, inverted tree is not correct mirror image of given tree.

Idea → For all the nodes, swap the left child & the right child.

```
void invert ( root) {
    if (root == null) return;

    invert (root.left)

    invert (root.right);

    Node temp = root.left;

    root.left = root.right;

    root.right = tmp;
}
```

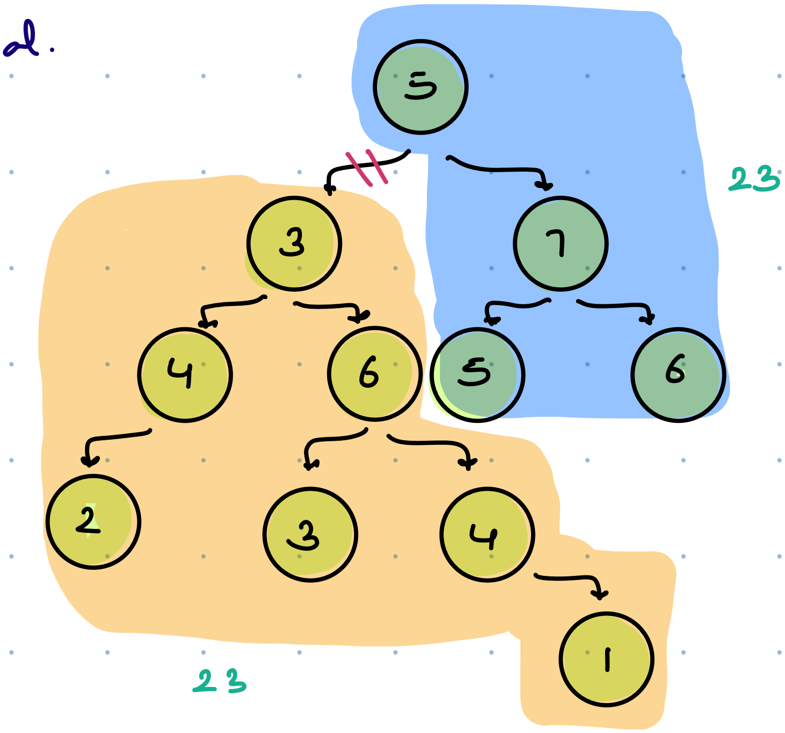
TC : O(n)

SC : O(ht)

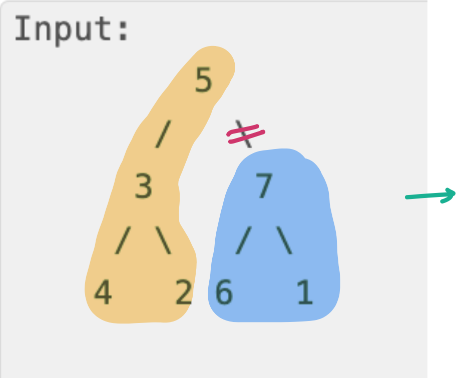


Equal tree partition

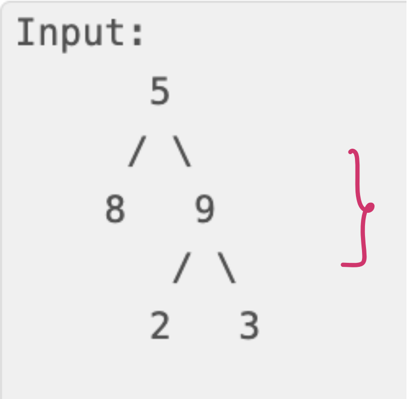
Q Check if it is possible to remove an edge from binary tree such that sum of resultant two trees is equal.



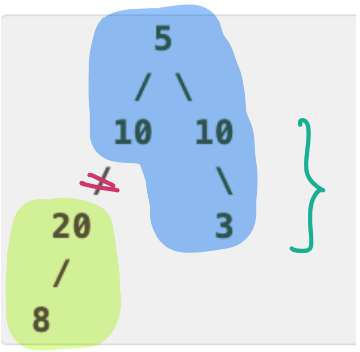
Ans= true


```
Input:
        5
       /
      3      7
     / \    / \
    4   2  6   1
```

→ Ans = true


```
Input:
     5
    / \
   8   9
      / \
     2   3
```

} Ans = false



Ans = true

Observation →

01. If sum of entire tree is odd, we can't split it into two parts → return false

02. If sum is even → check

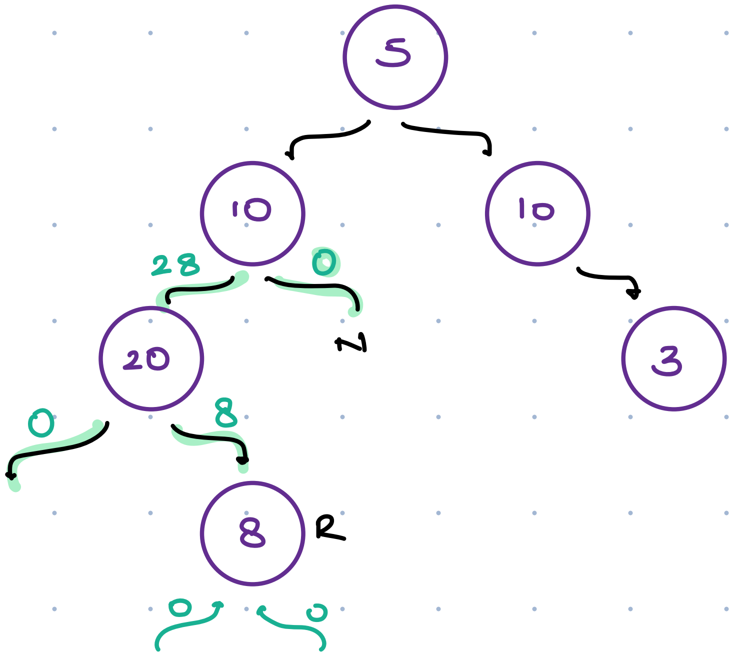
Total sum = s

if there is any subtree with sum = $s/2$

```
public int sum (root){
    if (root == null) return 0;
    return sum (root.left) + sum (root.right) + root.data;
}
public int solve ( Node root){
    int s = sum(root)
    if (s % 2 == 1 ) return false;
    find (root, s);
    return ans;
}
```

```
boolean ans = false;
public int find ( root , sum) {

    if (root==null) return 0;

    int l = find (root.left, sum);

    int r = find (root.right, sum);

    if ( l == sum/2 || r == sum/2 ) {
        ans = true;
    }

    return l + r + root.val;
}
```

* Travel & change

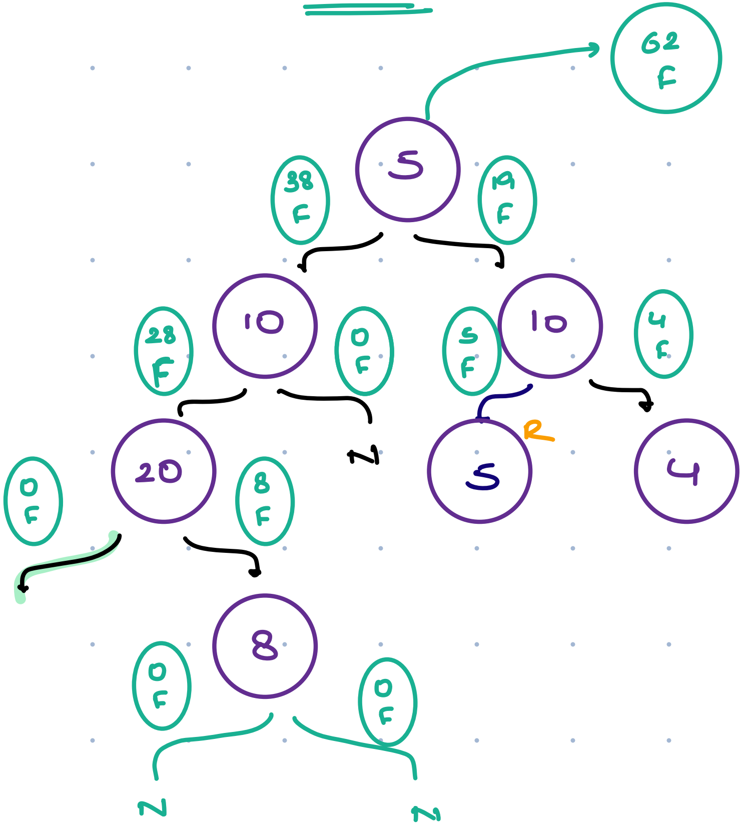


```
public class pair{
    int sum;
    boolean equal;
    pair ( int s , boolean e){
        sum = s;
        equal= e;
    }
}

public pair fun ( root , s ){

    if (root == null) return new pair (0, false);

    pair l = fun (root.left , s )
    pair r= fun (root.right , s )

    if ( l.equal == true || r.equal == true){
        return new pair ( 0, true);
    }
    else if ( l.sum = = s/2 || r.sum== s/2)
        return new pair ( 0, true);
    }
    else {
        return new pair ( l.sum + r.sum + root.val , false);
    }
}
```

TC : O(n)

SC : O(H)

```
public boolean solve( root){
    int s = sum ( root);
    if (s % 2 == 1) return false
    return fn(root,s). equal;
}
```



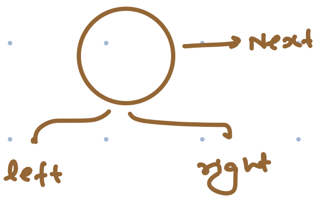
sum = 62

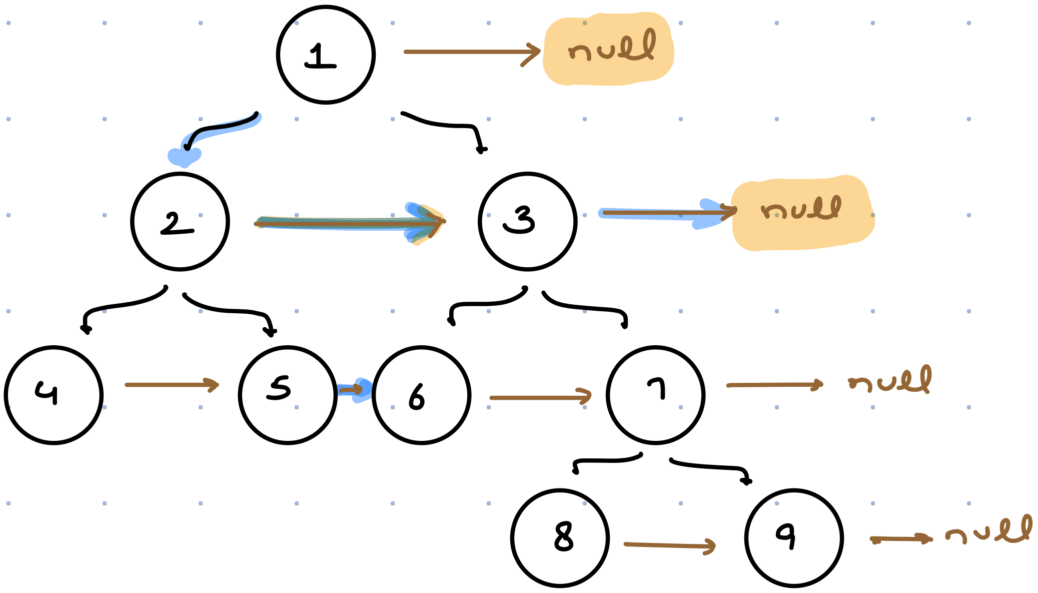
sum/2 = 31

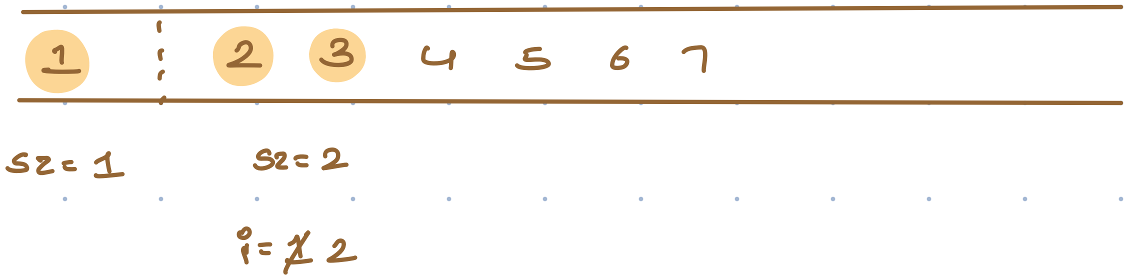
10:21 pm → 10:31 pm

Next Pointer in Binary Tree

For all the nodes in tree, next ptr points to null. Update next ptr to point to the next node in same level. (left to right)







```
Queue < Node > q = new LinkedList <>();

q. add (root);
```

```
while ( q.size () > 0) {
    int sz= q.size();
    for ( i=1; i≤ sz ; i++){
        Node rem= q.remove ();
        if (i== sz) rem.next = null;
        else  rem.next = q.peek();
        if (rem.left != null)  q.add (rem.left).
        if (rem.right != null)  q.add (rem.right);
    }
}
```

TC : O(n)

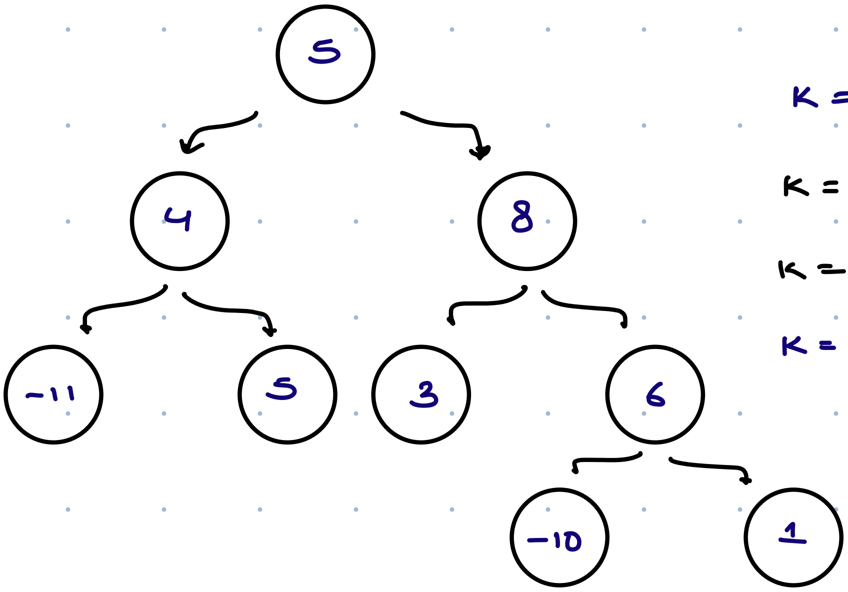
SC : O(n)

Note → This can be solved without extra space

⇓

You need to have Perfect Binary Tree

04. Given binary tree & an integer k. Determine if there exist a Root to leaf path with sum = k

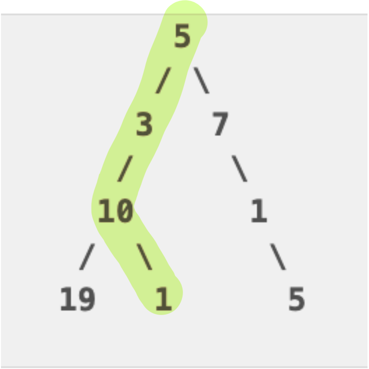


K = 16 → True

K = 10 → False

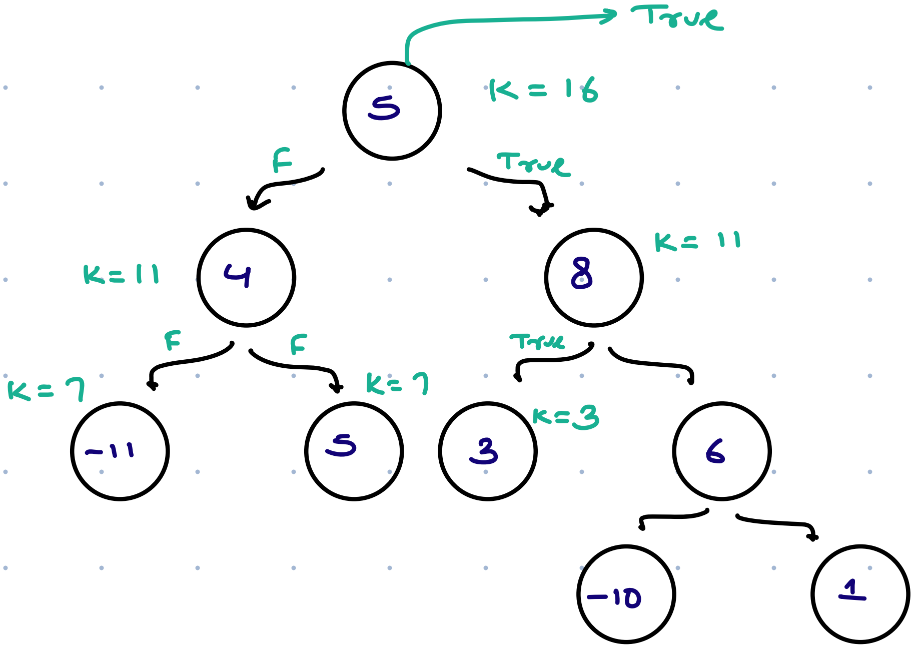
K = 9 → True

K = -2 → True



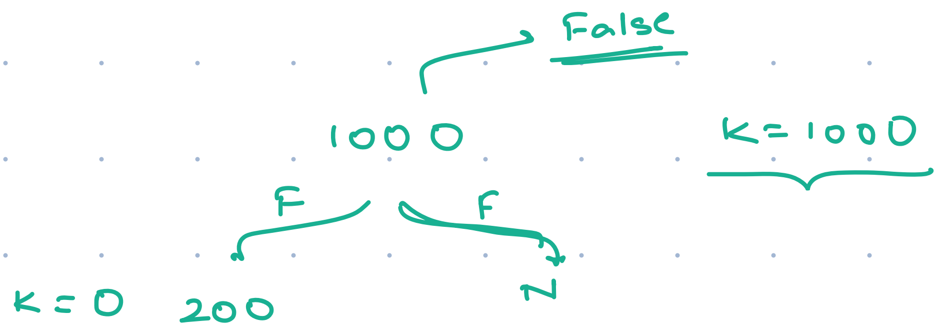
```
      5
     / \
    3   7
   /     \
  10      1
 /  \      \
19   1      5
```

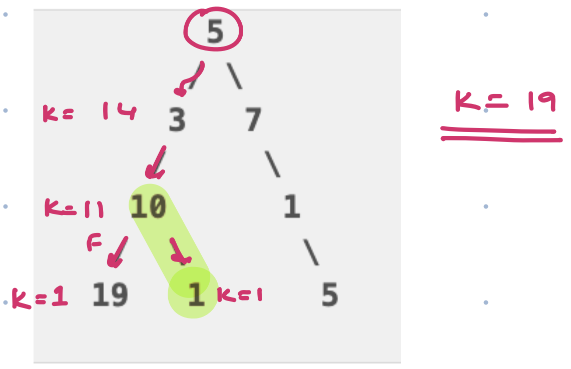
K = 19 → True



```
boolean check ( root , k ) {
    if (root == null) return false;

    if (root.left == null && root.right == null){
        return (root.val == k)
    }
    return check(root.left, k - root.val) ||
           check(root.right, k - root.val);
}
```

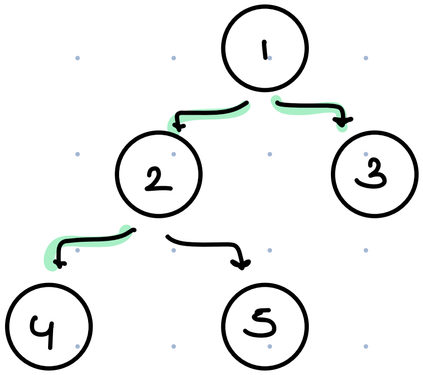
- * Path Sum 1
- * Path Sum 2
- * Path Sum 3



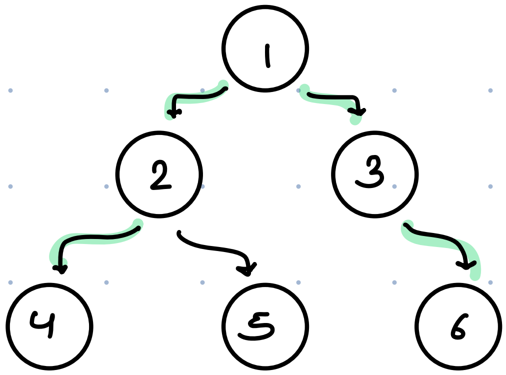


* Diameter of Binary Tree

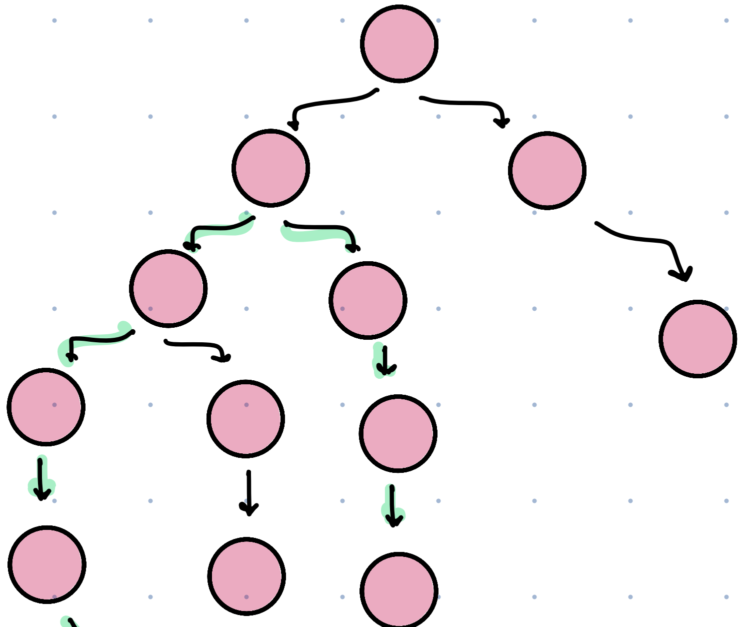
→ No. of the edges along the longest path between any two leaf nodes in the tree.



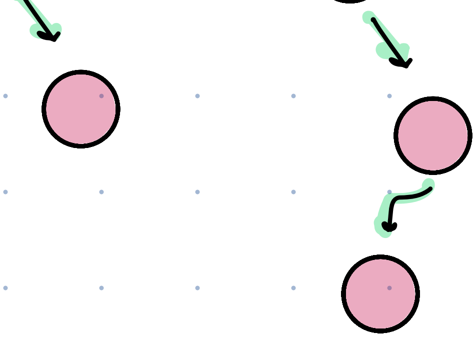
Ans = 3

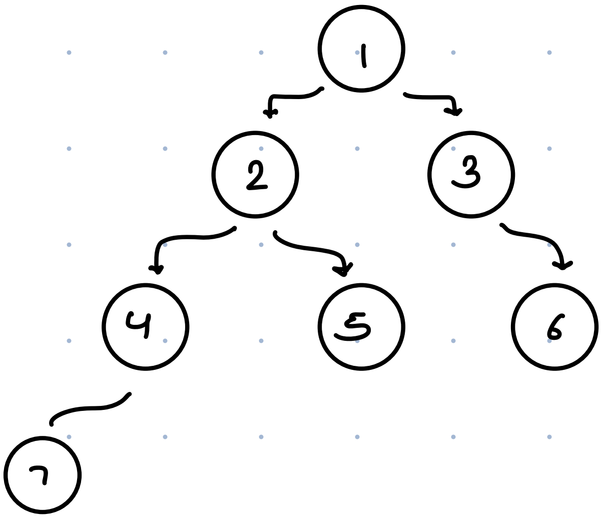


Ans = 4



Ans = 9





Dia = lh + rh + 2

* 3 possibilities

→ Dia can be LHS

→ Dia can be on RHS

→ Dia can pass through root.

```
public int diameter ( root) {
    int lh = height (root.left);
    int rh = height (root.right)
  return max ( diameter (root.left), diameter (root.right),
                         lh + rh + 2);
}
```

```
public int height ( root ) {
    if (root == null) return -1;
    int lh = height(root.left)
    int rh = height (root.right);
    return Math.max (lh, rh) + 1;
}
```

* Travel & Change

```
int dia = 0
int height ( root ) {
    if (root == null) return -1;
    int lh = height(root.left)
    int rh = height (root.right);
    dia = Max (dia, lh+rh+2);
    return Math.max (lh, rh) + 1;
}
```