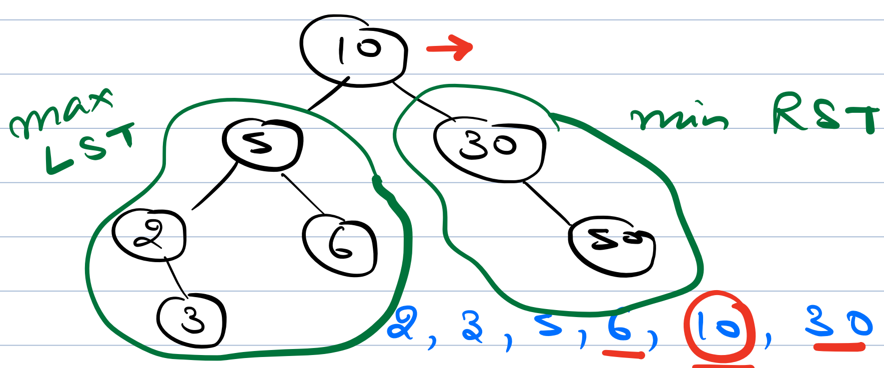
Inorder - LNR (BST)

T.C. = O(N)

S.C. = Stack space = O(H) = O(N)

O(1)



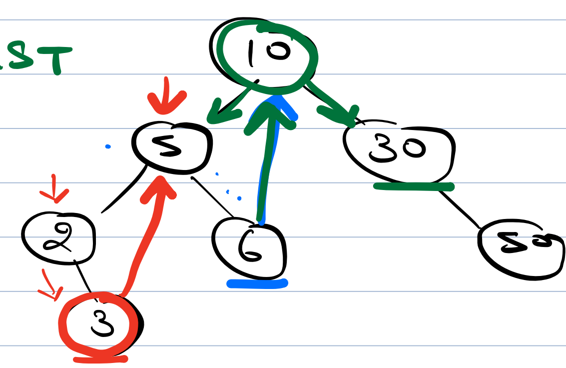
2, 3, 5, 6, 10, 30, 50

Inorder predecessor

Inorder Successor.

LST, 10, RST

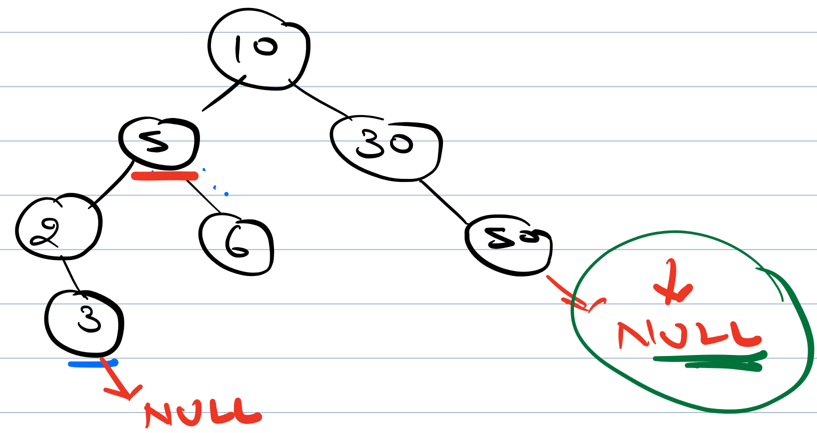
..... 6



(Creating a back link from inorder predecessor)

2, 3, 5, 6, 10, 30, 50

⇒ ∀ nodes, we try to find the inorder predecessor. & update it's right to the node.

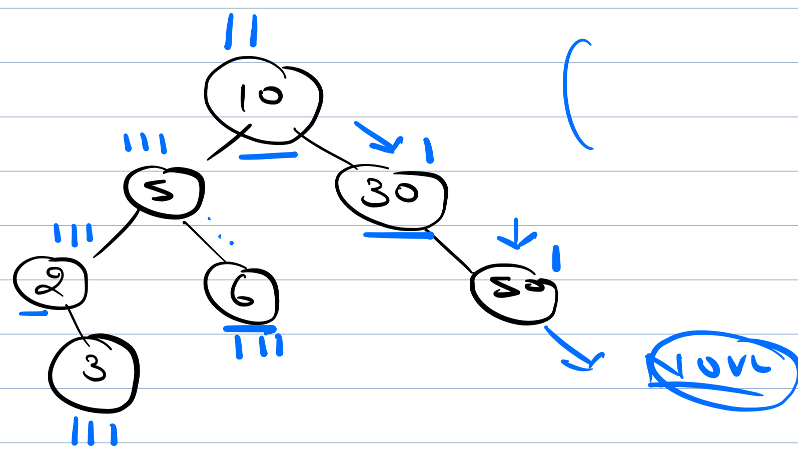


2, 3, 5, 6, 10, 30, 50

Code

```
curr = root;
while (curr != null) {
    if (curr.left == null) {
            print curr.value;
            curr = curr.right;
    }

    else {
```

```
pred = findPredecessor(curr);
if (pred.right == null) {
        pred.right = curr
        curr = curr.left;
}
else {
        pred.right = null;
        print(curr.value);
        curr = curr.right;
}
    }

}
```



2, 3, 5, 6, 10, 30, 50

T.C. = O(3N) = O(N)

S.C. = O(1)

```
findPredecessor (root) {
        curr = root;
        curr = curr.left;

        while ((curr.right != null) &&
                        (curr.right != root)) {
                curr = curr.right;
        }

        return curr;
}
```

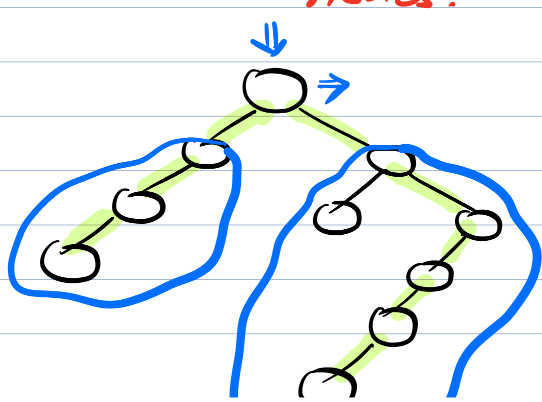
Q

Amazon
MS
Adobe
Google.

Given a BT.

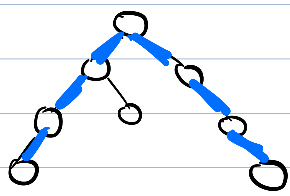
Find the diameter of the tree

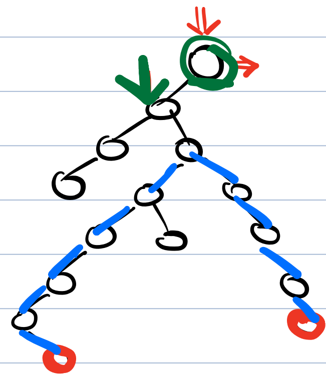
Length of the longest path b/w any 2 nodes.



$\Rightarrow$ 8

Height (LST) + Height (RST) + 2





HLST + HRST
+2

Code

```
int dia (root) {
    if (root == null) {
        ret -1;
    }

    int lh = height (root.left);
    int rh = height (root.right);
    ans = lh + rh + 2

    int ld = dia (root.left);
    int rd = dia (root.right);
    return max( ld, rd , ans);
}
```

$T.C. = \underline{O(N^2)}$

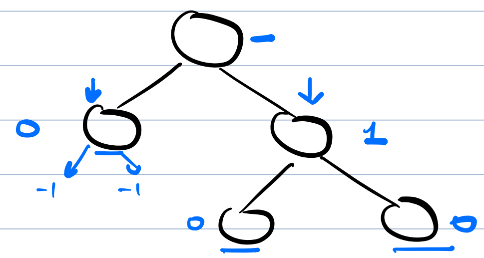
```
class TreeInfo {
    int height;
    int dia;
    constructor() ....
}

TreeInfo dia (root) {
    if (root == NULL) {
        return new TreeInfo (-1, -1);
    }

L   TreeInfo l = dia(root.left);
R   TreeInfo r = dia (root.right);

N   return new TreeInfo ( max( l.height,
            r.height) + 1,
      max( l.diameter, r.diameter,
            ( l.height + r.height) + 2))
}
```

Height Balanced Tree

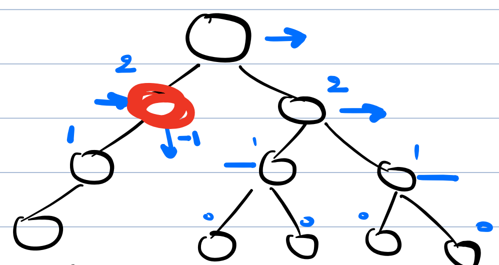
$$\text{Height}(\text{Height Balanced Tree}) = \log_2 N$$

$$|\,ht(LST) - ht(RST)\,| <= 1$$

$\forall$ nodes of the tree.



$1-(-1) = 2$



height　　isBalanced.

TreeInfo isBalanced(root) {

H.W. Code