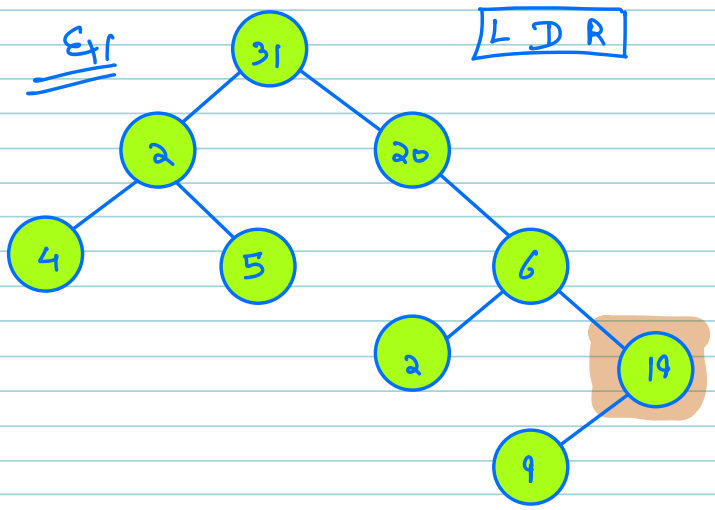
Trees 4 : LCA + Morris Inorder Traversal

Today's Content

- Morris Inorder traversal
- K^{th} Largest Element in a BST
- LCA in BST (Binary Search Tree)
- LCA in BT (Binary Tree)

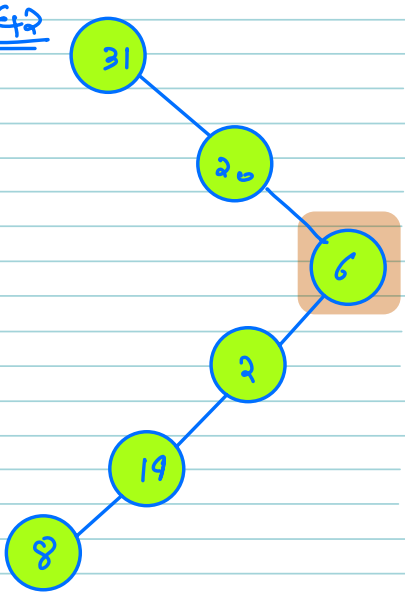
Question :- With Inorder on a Tree, Print the last Node traversed.

Ex1



L D R

Ex2



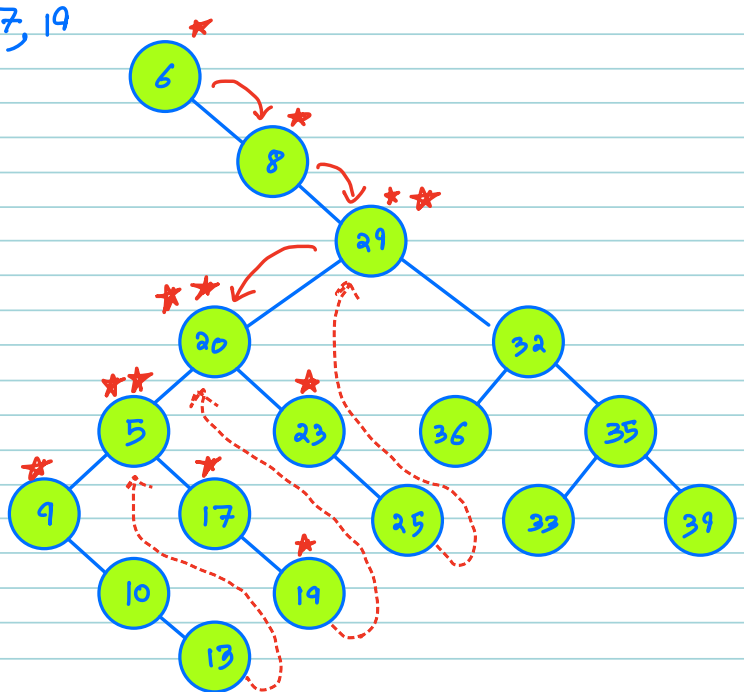
Claim :- In Inorder Traversal of BT last Node will always be right most Node of root.

TC → O(N)

Inorder
- Recursively : O(H)
- Iteratively [using stack] : O(N)
- Morris traversal : O(1)

Question :- Morris Inorder Traversal {HARD}
↳ {No Extra Space}

6, 8, 9, 10, 13, 5, 17, 19
20, 23, 25, 29



⇒ We Need to Secure a way to come back when going Left in Inorder.

↳ Way to Secure

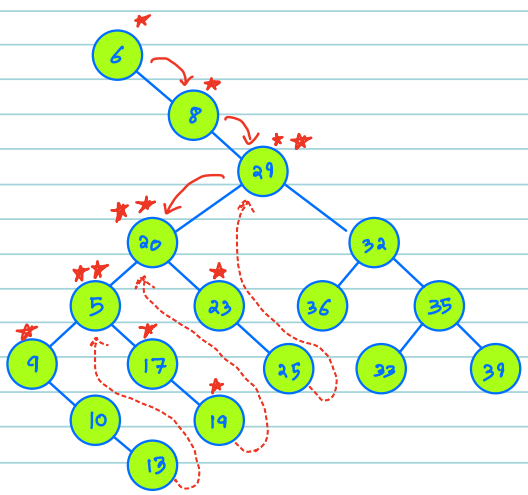
⇒ check the Last Node of Left Side where you are going i.e. last Node in Subtree starting from 20.

⇒ Connect the last Node's right of LST to current Node.
i.e. (25) ⟶ to (29)

Q If we have Made connections. Then I will come back on Member again after LST traversal So how I will figure out that I'm on a Node second time ?

↳ Check the last Node of LST, if its connected to you already, then we are going for the second time.

⇒ when on node for second time break the path of Last Node of LST to yourself.



PSEUDO CODE

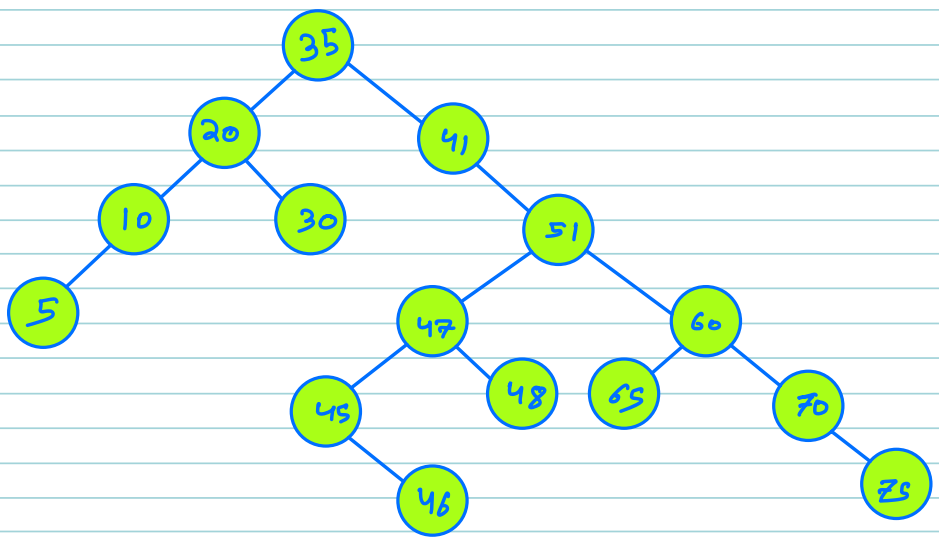
```
Inorder ( root) {
    curr = root;
    while ( curr != null ) {
        if ( curr. left == null) {
            Print (curr. data);
            curr = curr. right;
        } else {
            Node temp = curr. left;
            while (temp. right != null
                && temp. right != curr){
                temp = temp. right;
            }

            if ( temp. right == null){
                temp. right = curr;
                curr = curr. left;
            } else {
                temp. right = null;
                print (curr. data);
                curr = curr. right
            }
        }
    }
```

Quiz! :- what is the primary advantage of morris Traversal for binary Trees?

↳ memmory efficient

Question :- K^{th} smallest element in a BST



K = 1 ⟶ 5

K = 2 ⟶ 10

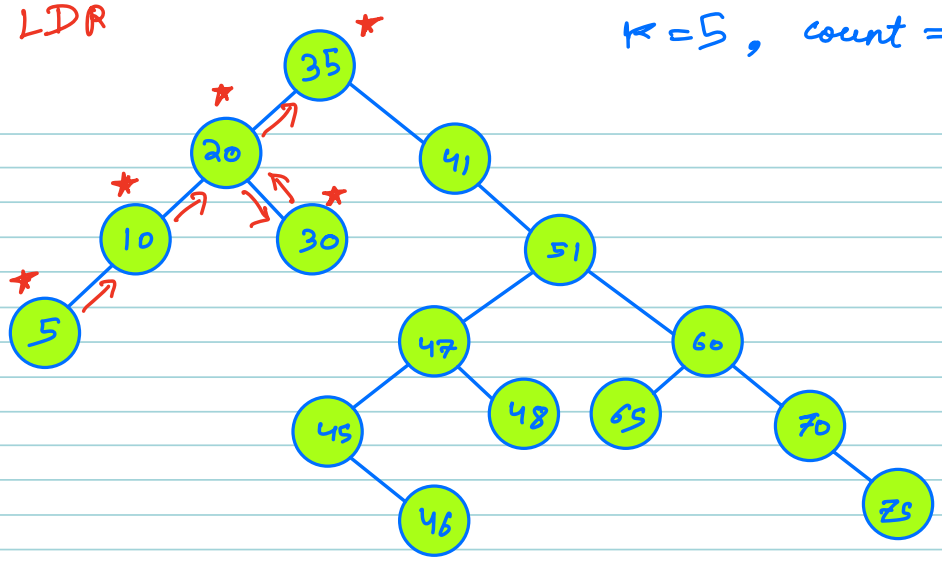
K = 3 ⟶ 20

K = 4 ⟶ 30

K = 5 ⟶ 35

Brute force :- Do inorder traversal & store element in list, return ans[K-1] element.

LDR



k = 5 , count = ~~0 1 2 3~~ 4 5

PSEUDO CODE

```
ans = -1
public void inorder ( Node root ){
    if ( root == null ) { return; }
    inorder ( root.left );
    count ++;
    if ( count == k){
        ans = root.data;
    }
    inorder ( root.right );
}
```

TC → O(N)

SC → O(H)

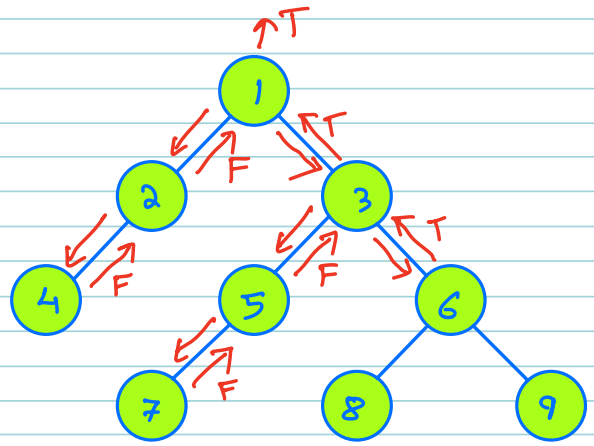
↳ OPTIMIZE

↳ use Morris traversal for Inorder.

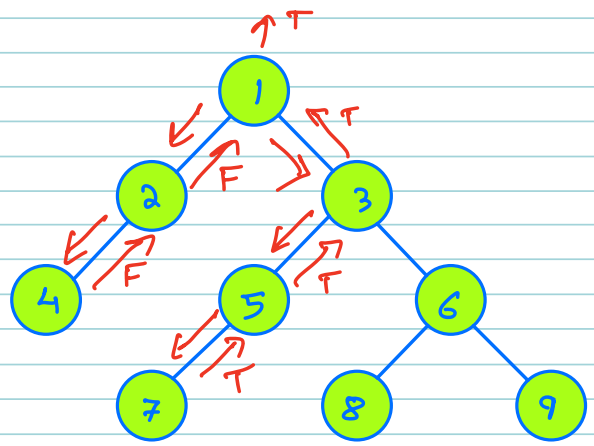
TC — O(N)
SC — O(1)

Question :— Search K in given BT.

K = 6



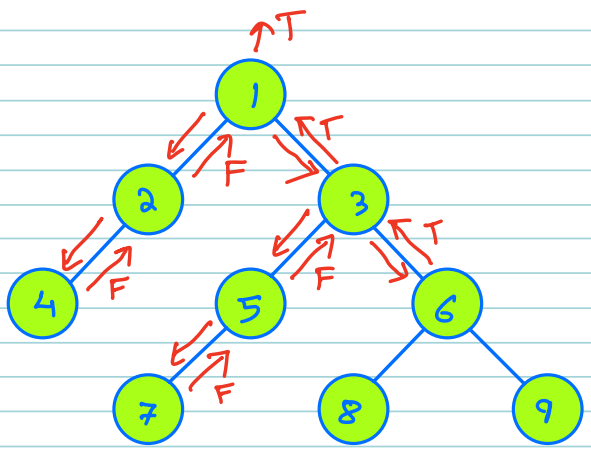
K = 7

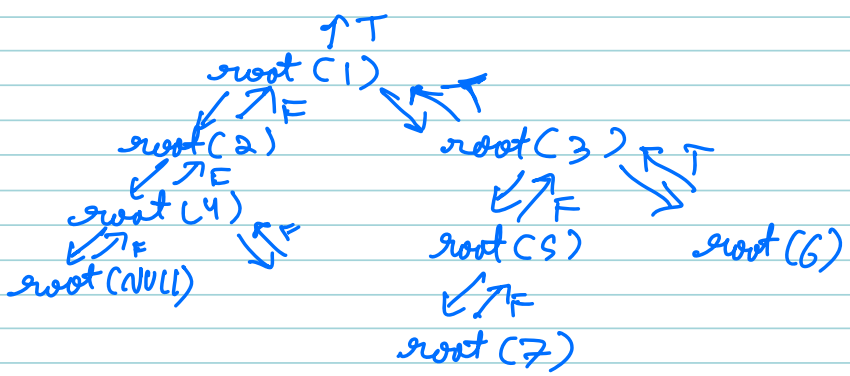


PSEUDO CODE

```
boolean Search ( Node root, K ) {
    if ( root == Null ) { return false; }
    if ( root.data == K ) {
        return True;
    }
    return Search ( root.Left, K) ||
           Search ( root.right, K);
}
```

K = 6

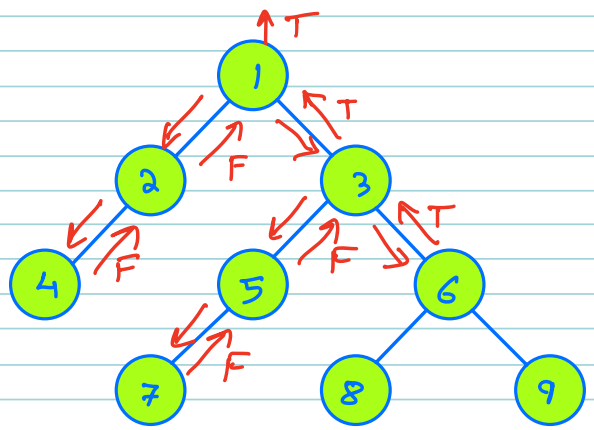




Question :- Root to Node path

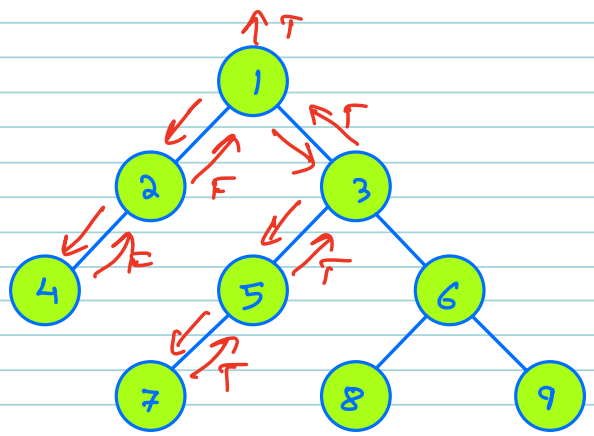
K = 6

[1, 3, 6]



K = 7

[1, 3, 5, 7]



OBSERVATION :- All the Nodes in Path is returning. So can we say wherever I was returning True. we just Need to store data.

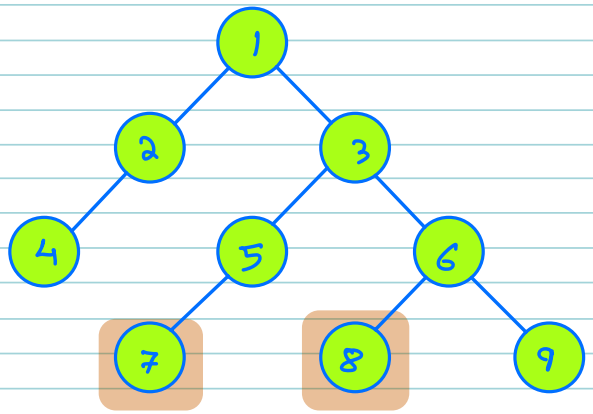
PSEUDO CODE

```
boolean Search ( Node root, K, List<int> ans){
    if ( root == Null ) { return false;}
    if ( root.data == K) {
        ans.add( root.data);
        return True;
    }
    if ( Search ( root.Left, k, List ))){
        ans.add ( root.data );
        return True;
    }
    if ( Search ( root.right, k, List)) {
        ans.add (root.data );
        return True;
    }
    return False.
}
```

* Just Need to reverse ans in end.

TC → O(N)
SC → O(H)

Question :- Lowest Common Ancestor



LCA(7,8) → 3

LCA(8,1) → 6

LCA(7,6) → 3

LCA(3,7) → 3

LCA(6,9) → 6

IDEA 1

LCA(7,8)

↳ Root to Node path : 1 3 5 7

↳ Root to Node path : 1 3 6 8

LCA(8,9)

↳ Root to Node path : 1 3 6 8

↳ Root to Node path : 1 3 6 9

LCA(3,7)

↳ Root to Node path : 1 3

↳ Root to Node path : 1 3 5 7

// Generalize

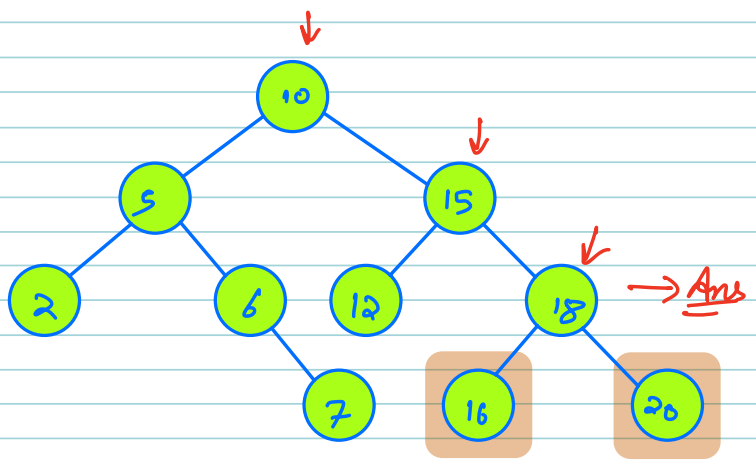
LCA(x, y) → Find root to node path of x & y

Find Last common value in both arrays.

Follow up :- LCA in BST

LCA (12, 16)

LCA (16, 20)



IDEA:- The lowest point where the No are Splitting into different branches will be the Ans

PSEUDO CODE

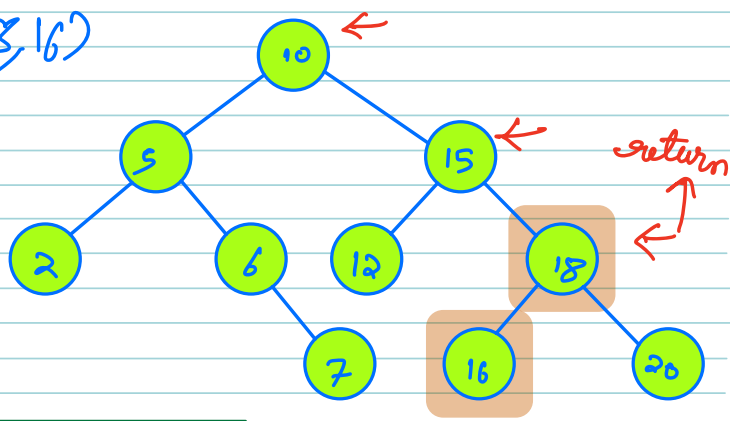
```
curr = root;

while ( curr != null ) {

    if ( curr.data > x && curr.data > y ) {
        curr = curr.left;
    } else if ( curr.data < x &&
                curr.data < y ) {
        curr = curr.right;
    } else {
        return curr;
    }
}

return null;
```

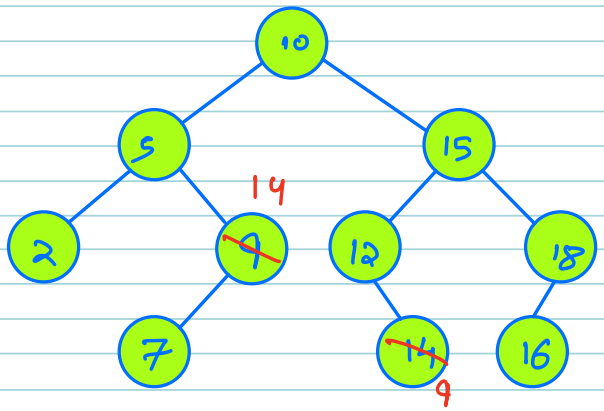
LCA(18, 16)



TC → O(H)

SC → O(1)

Question :- Recover BST



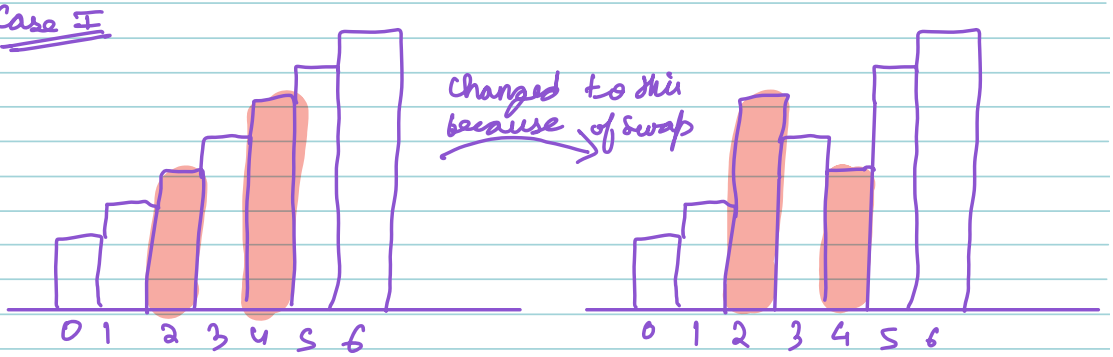
Two Nodes of a BST are swapped, Find the two Nodes.

IDEA :- Inorder of a BST is Sorted.

2 5 7 14 10 12 9 15 16 18

↑

Case I



Ideal Scenario for all Nodes/Number

```
arr[i] > arr[i-1]
```

Exist Point

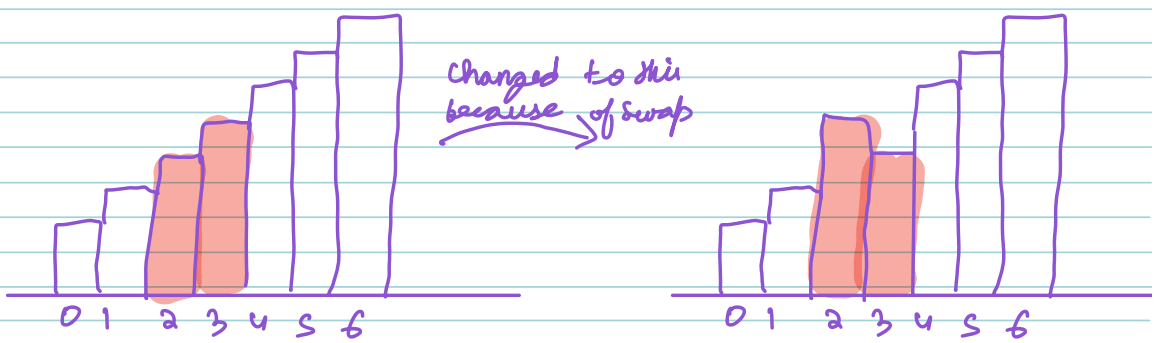
↳ Prob1 = arr[i-1]

Second Point

↳ Prob2 = arr[i]

Case II

2 5 7 10 4 12 14 15 16 18



FIX:-

Exist point

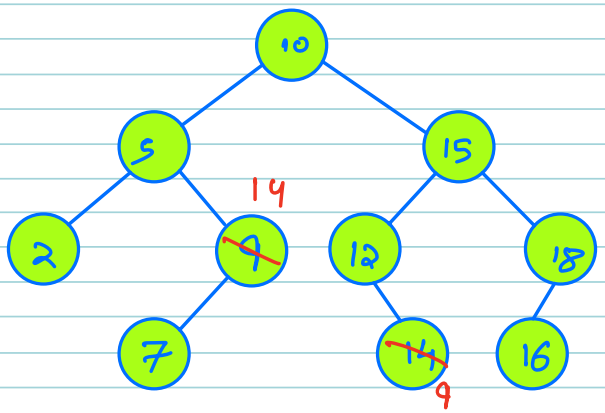
Prob1 = arr[i-1]

Prob2 = arr[i]

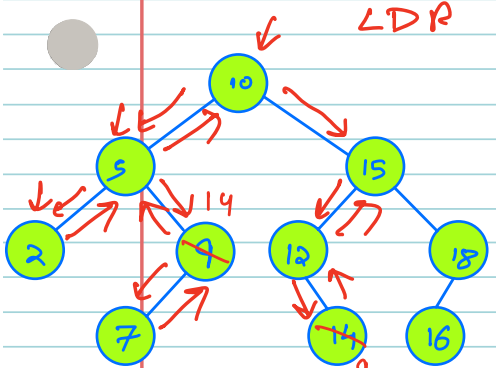
Second point

Prob2 = arr[i]

Implementation IDEA :-



PSEUDO CODE




```
Node P1, P2, Prev = NULL;
Inorder (Node root){
    Inorder (root.left);
    if (Prev != null &&
            root.data < p.data){
        if (P1 == NULL){
            P1 = Prev;
            P2 = root;
        } else {
            P2 = root;
        }
    }
    Prev = root;
    Inorder (root.right);
}
```

P1 = ~~NULL~~ 14

P2 = ~~NULL~~ ~~10~~ 4

Prev = ~~NULL~~ ~~2~~ ~~5~~ ~~7~~ ~~14~~ ~~10~~ ~~12~~ ~~15~~ ~~16~~ ~~18~~

$TC \rightarrow O(N)$

$SC \rightarrow O(H)$

Q How can we do it better?

↳ Using Morris traversal.