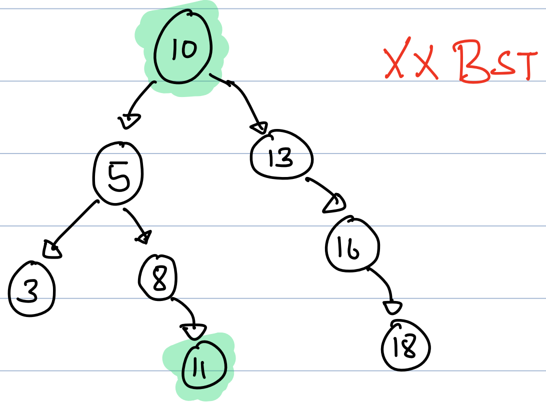
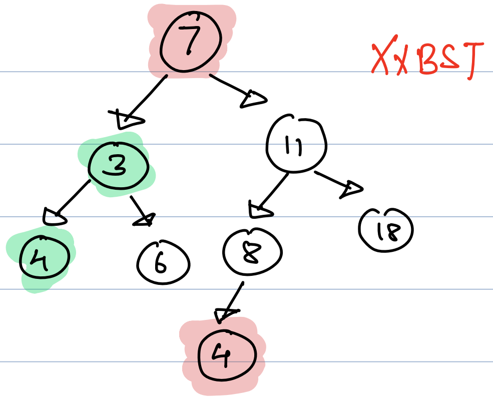
BST [Binary Search Tree]

A Binary tree is BST if.

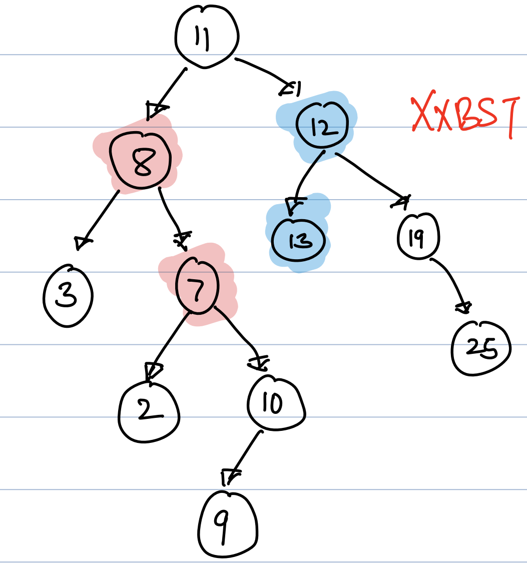
For all nodes : All Elements in LST < node < All Elements in RST

Ex1

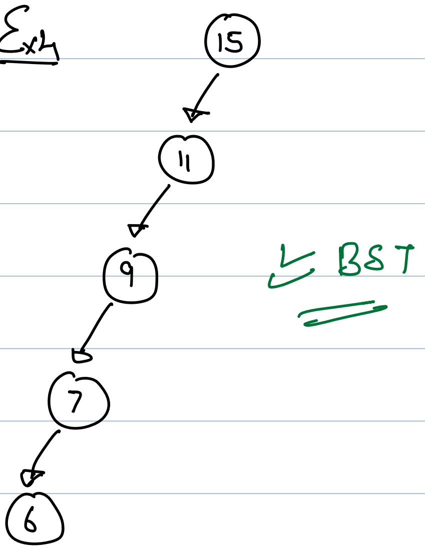




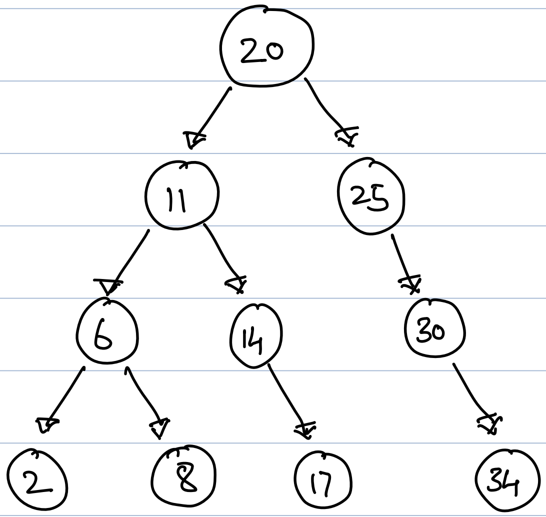
Ex3



Ex4



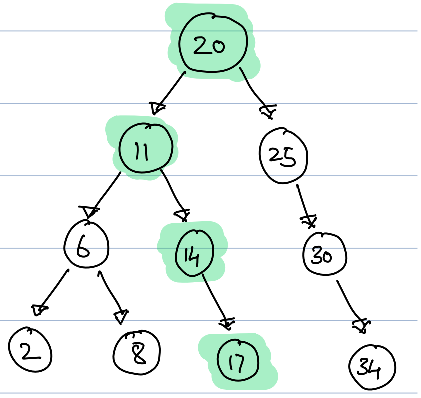
Interesting Property.



Inorder : 2 6 8 11 14 17 20 25 30 34

$\Rightarrow$ Sorted Order

Q1 Search for a node in BST. You are given root.



Ex : Search for $k=17$.

```
bool findNode (Node root, int k) {

    Node temp = root;

    while (temp != null) {

        if (temp.val == k) {
            return true;
        } else if (temp.val > k) {
            temp = temp.left;
        } else if (temp.val < k) {
            temp = temp.right
        }

    }

    return false;
}
```

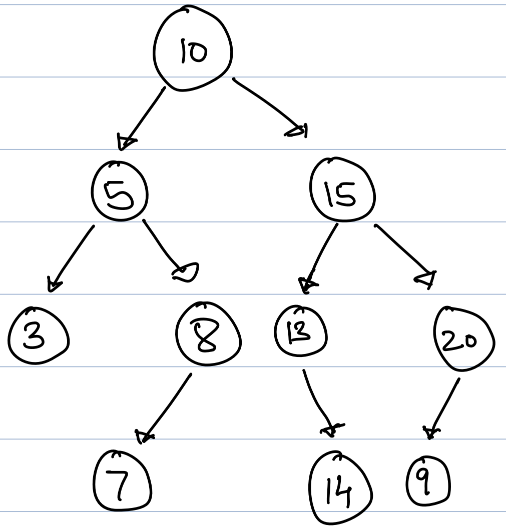
TC : O(H)

worst case : O(n)

Skewed tree

SC : O(1)

Q2 Given BT check BST or not .



Approach 1

Check inorder traversal if it is sorted or not.

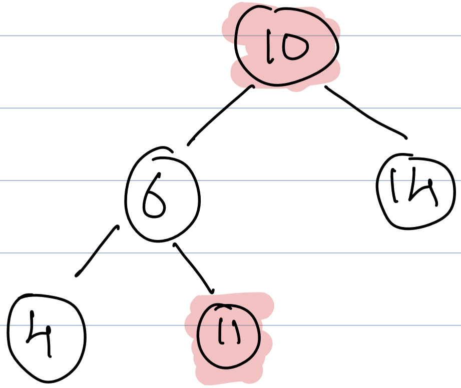
TC : O(n)
SC : O(n)

Approach 2 : max of lst < root < min of RST

Total n nodes

1^{st} node : ⇒ (n-1)
2^{nd} node : ⇒ (n-2)

TC : $O(n^2)$

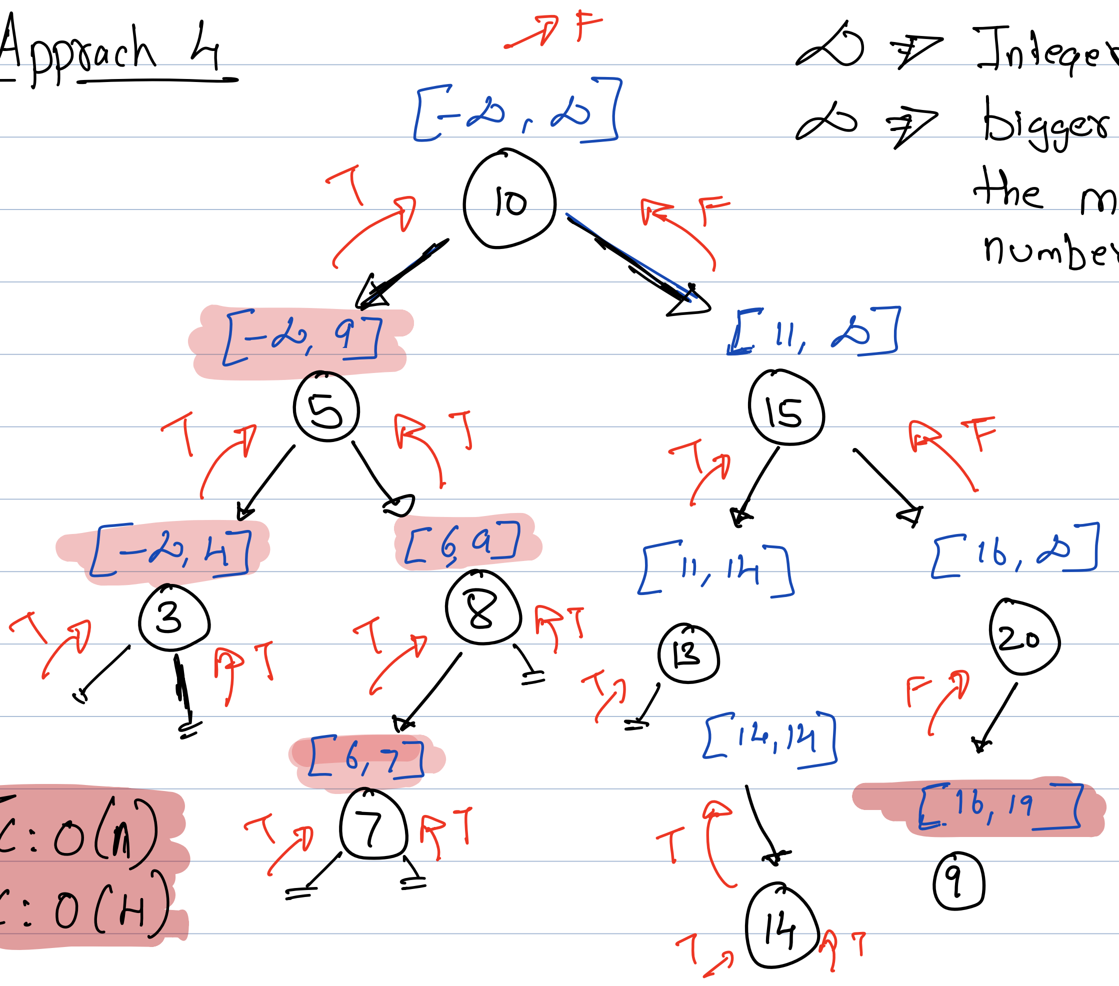


Approach 3 : left child value < root < right child value.

Apprach 4

∞ ⇒ Integer.max

∞ ⇒ bigger then the max number allowed.

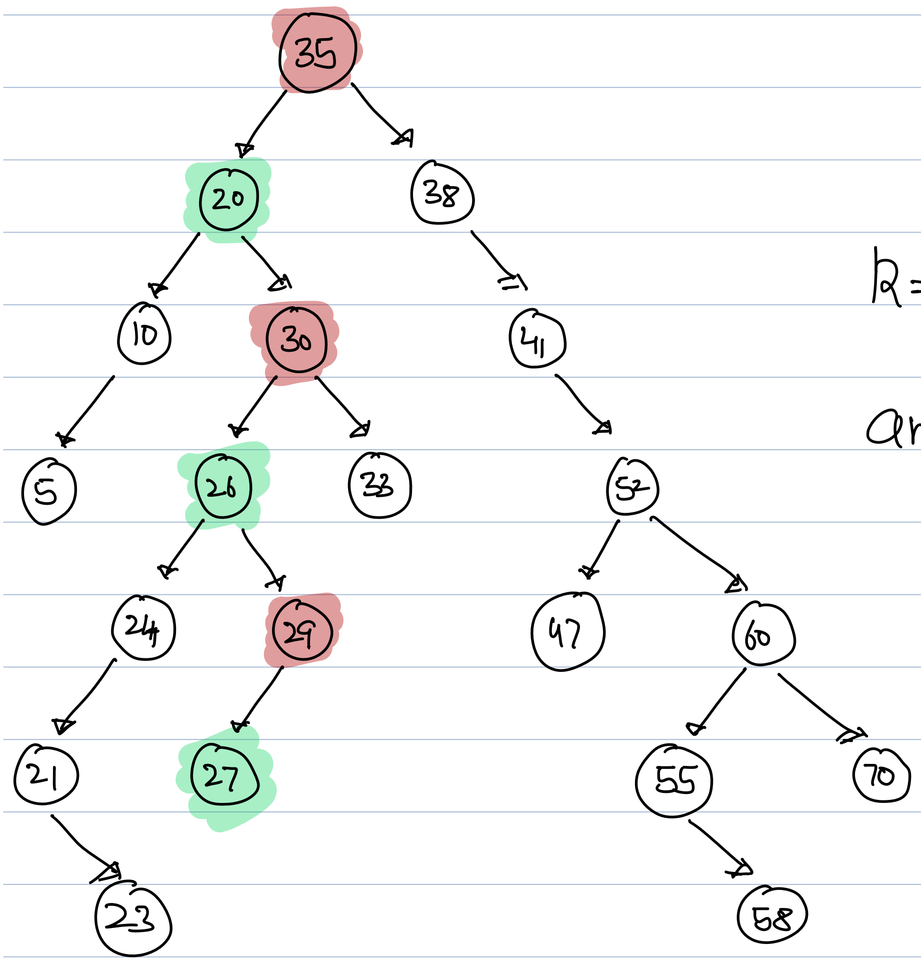


TC: O(n)
SC: O(H)

```
bool isBST ( Node root , int l, int r) {
    if ( root == null) return true;

    if (root.val ≥ l && root.val ≤ r) {

        return isBST (root.left, l, root.val -1 )
               &&
               isBST (root.right, root.val +1, r );
    } else
        return false;
}
```

Q3 Given K, find floor of K.

greatest element ≤ K in BST

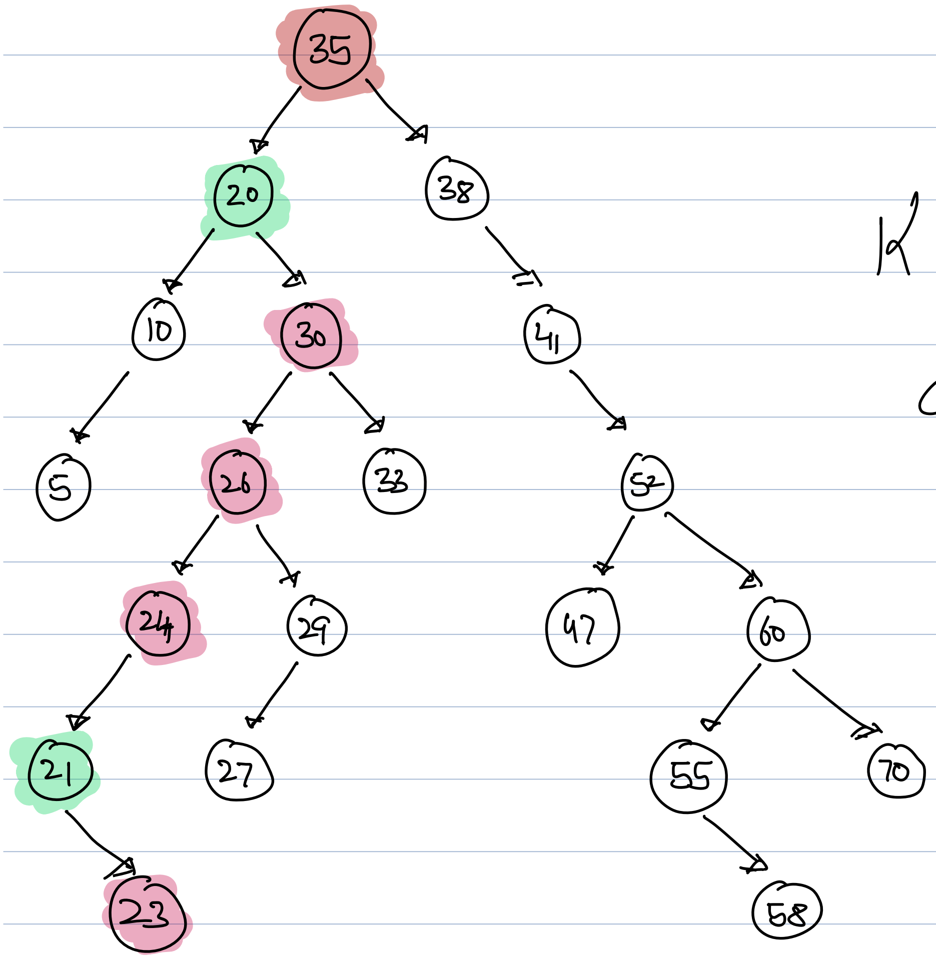


K=28

ans = 27

TC: O(H)

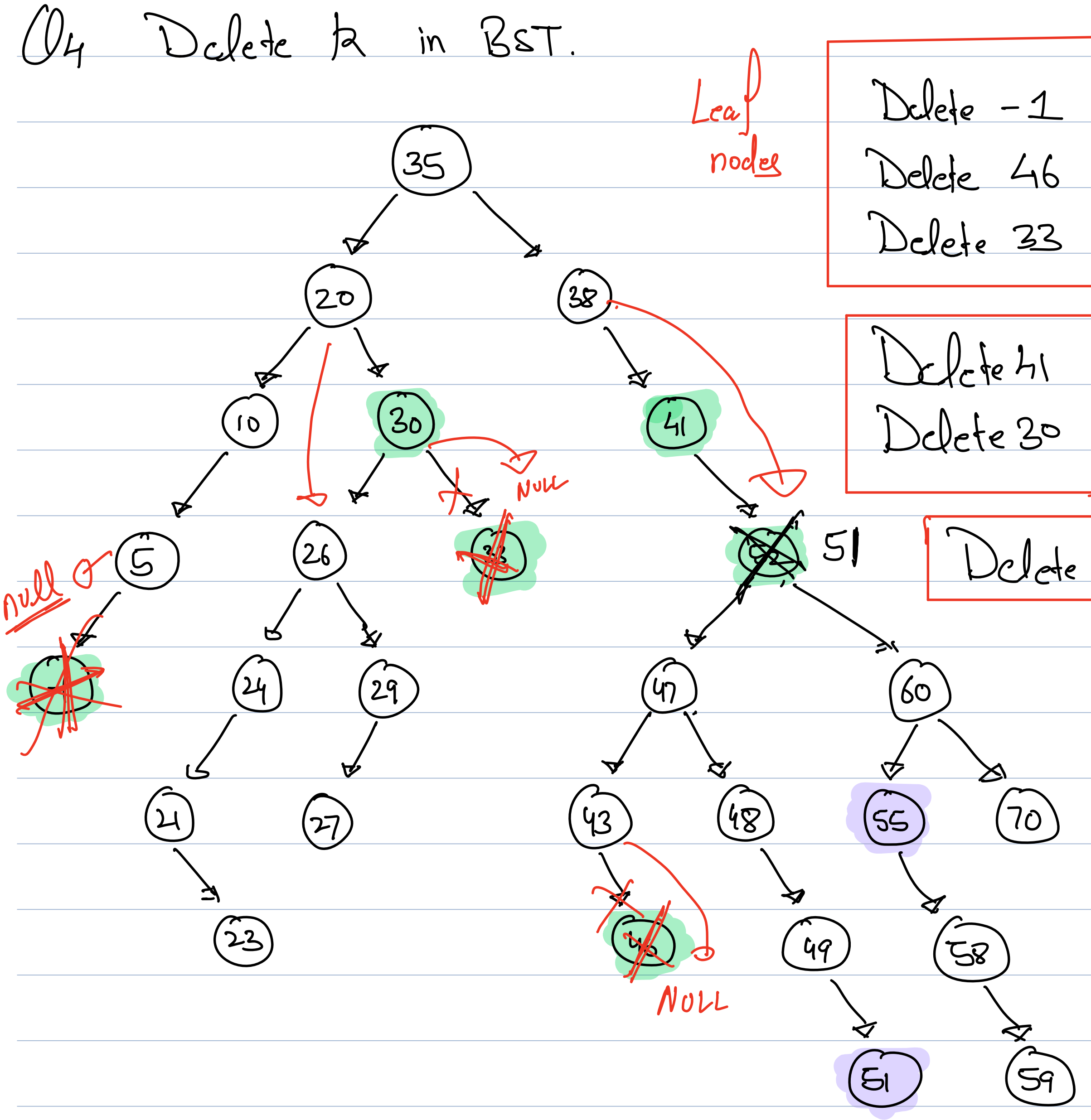
SC: O(1)



K = 23

ans = 21

Q4 Delete k in BST.



Delete -1
Delete 46
Delete 33

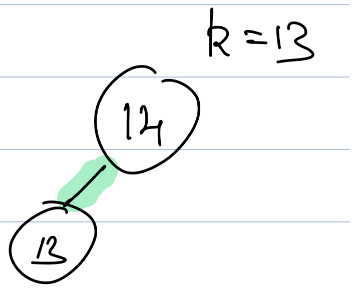
Delete 41
Delete 30

Delete 52

```
Node deleteNode ( Node root, int k) {
     if( root == null)
          return root;
   if (root.val == k ) {
      if ( root.left == null && root.right = null){
             return null;
      }

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

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

     int x => max (root.left)
     root.val => x

     root.left => deleteNode (root.left, x);

   return root;
}
```

leaf node

One Child

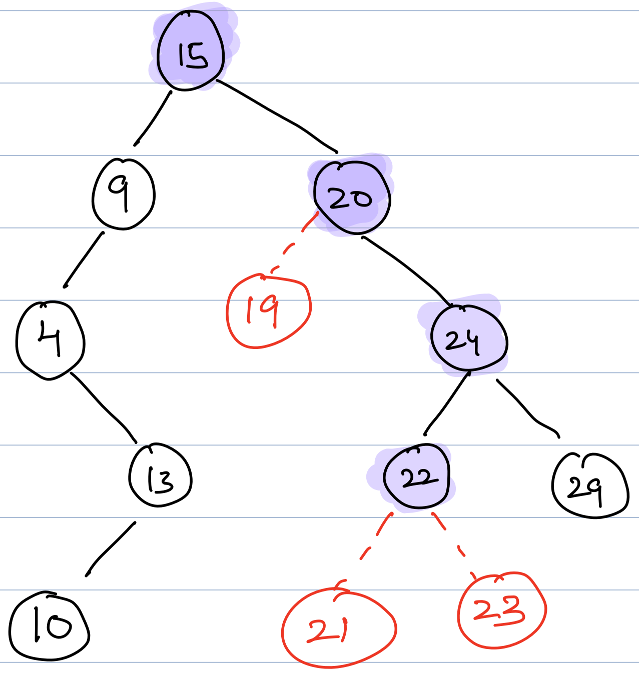


```
if (root.val > K) {
     root.left => delete (root.left , k);
} else if (root.val < K)
     root.right => delete (root.right, k);
}

   return root;

}
```

Tc : O(H)

Sc : O(H)

Q4 Insert in BST



Add 19

Add 23

Add 21

TC : O(H)

SC: O(1)