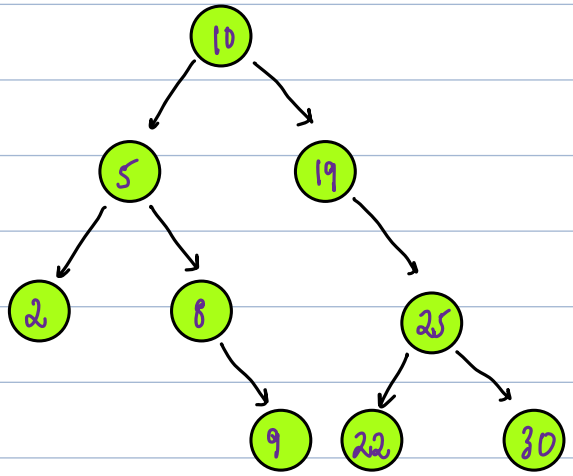
Todays Content

1. Trim BST
2. Check if given BT is BST or not

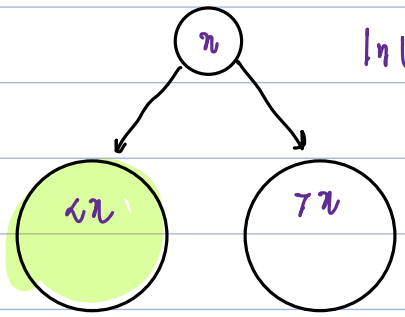
Given a BT, check if it's a BST or not.

Note: Assume all nodes are distinct

Eg1:



Ideal:



Inorder: L D R

$< n \quad n \quad > n$

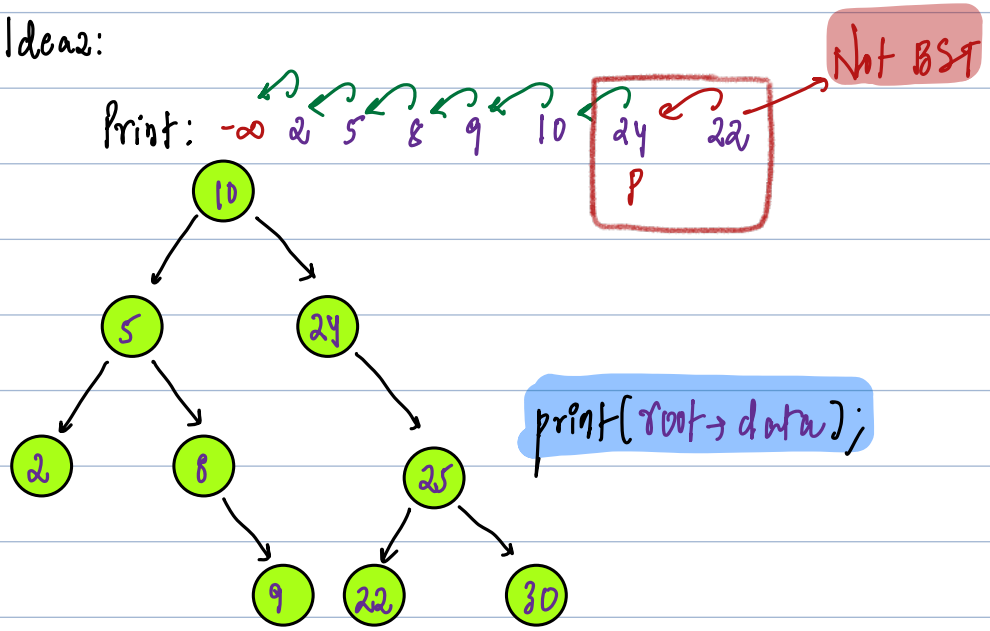
property BST: Inorder of BST is increasing order

Ideal: Store inorder in an array & check if array is in increasing order.

TC: $O(N+N) = O(N)$ SC: $O(N)$

↳ # storing in an array.

Idea2:

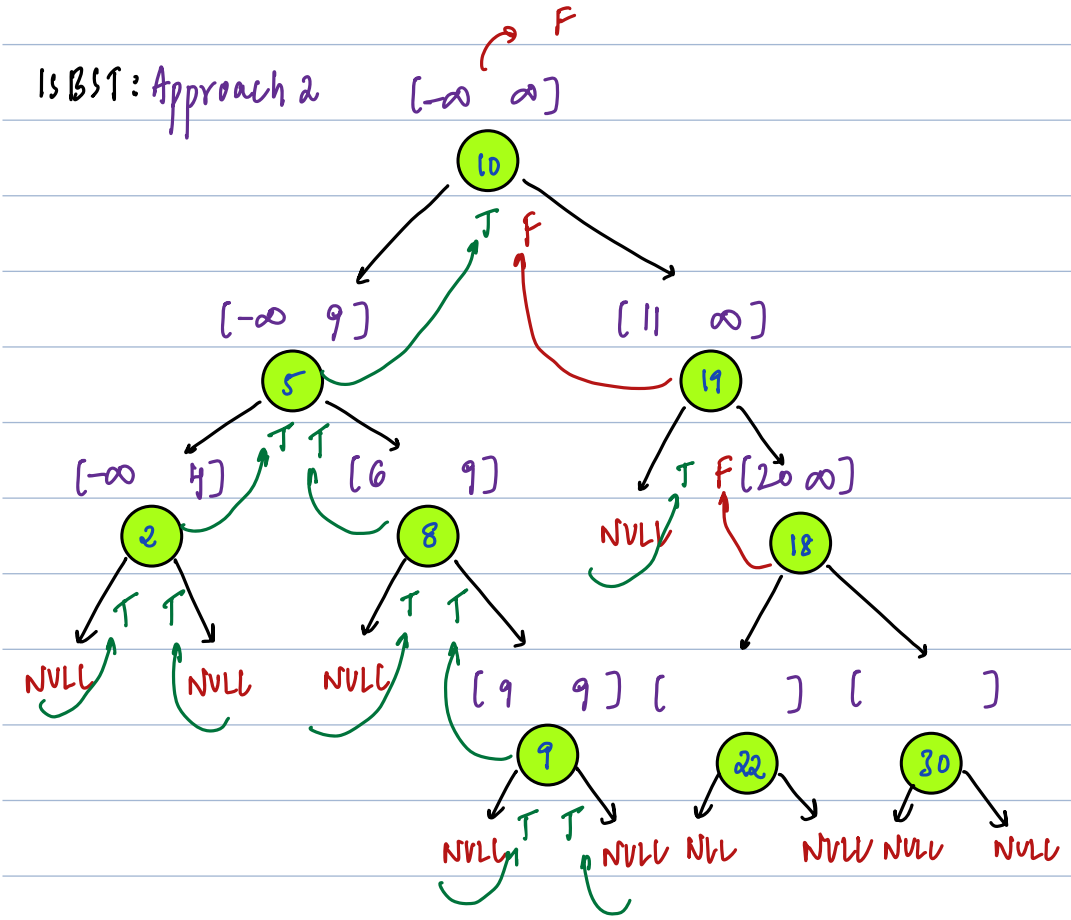



```
bool isBST=true; -> or global
int p=-∞;
void Inorder(Node* root, bool &isBST, int &p){ TC: O(N)  SC: O(H)
    if(root==nullptr){return;}
    Inorder(root->left, isBST, p);
    if(p > root->data){
        isBST=false;
    }
    p=root->data;
    Inorder(root->right, isBST, p);
}
```

pass by reference

```
bool isBST(Node *root){
    isBST=true;
    p=-∞;
    Inorder(root, isBST, p);
    return isBST;
}
```

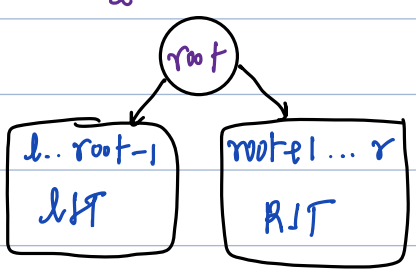
IsBST: Approach 2



Ans: Given root node & range check if entire BST is in range or not.

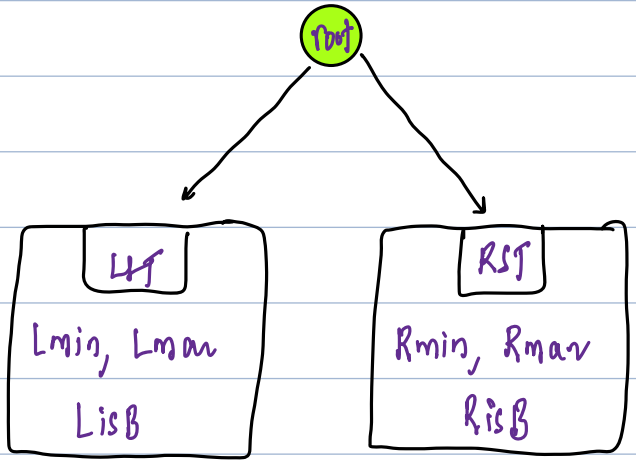
```
bool IsBST(Node *root, int l, int h){
    if(root==nullptr){
    }return true;

    return
        ((l<=root->data)&&(root->data<=r)
        && IsBST(root->left, l, root->data-1)
        && IsBST(root->right, root->data+1, r))
}
```




```
boolean solve(Node root){
    return IsBST(root, -∞, ∞);
}
```

Ideas:




```
vector<int> isBST(Node *root){
    if(root==null){
        vector<int> ans(3,0);
        ans[0]=1; ans[1]=+∞; #Min  ans[2]=-∞; #Max
        return ans;                                  0    1    2
    vector<int> l = isBST(root->left);   # l: isB  min  max
    vector<int> r = isBST(root->right);  # r: isB  min  max
    vector<int> ans(3,0);

    if( (l[0]==1) && (r[0]==1) && (root->data > l[2]) && (root->data < r[1]) ){
        ans[0]=1;
    ans[1]= min(l[1], r[1], root->data);   #Note we compare with root->data,
    ans[2]= max(l[2], r[2], root->data);    because if both left & right are
    return ans;                             null, then data should be
}                                           compared with root

bool solve(Node *root){
    vector<int> ans = isBST(root);
    return ans[0]==1;
}
```

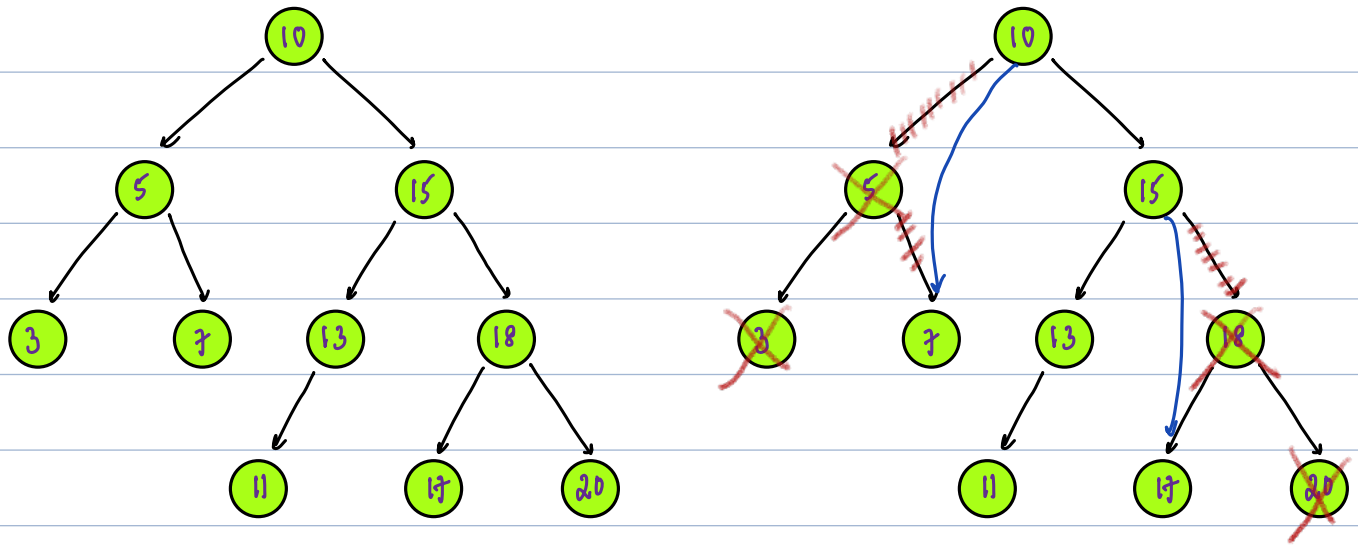
Trim BST:

Given a BST & low & high.

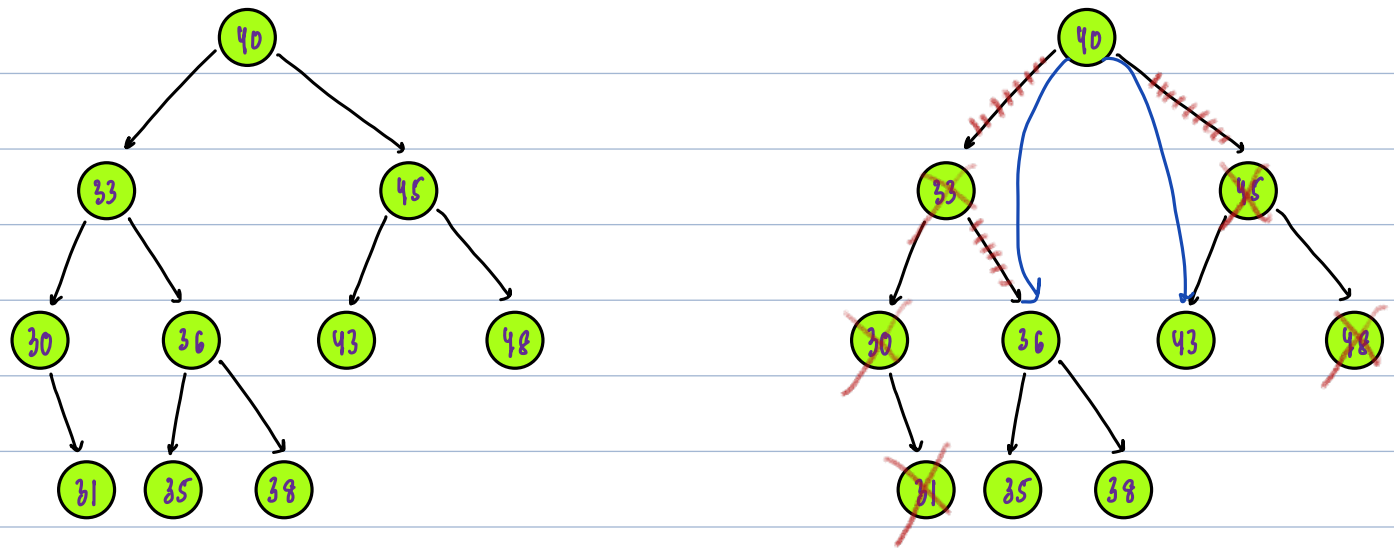
Delete all nodes not in range & make sure relative order maintained.

tc1

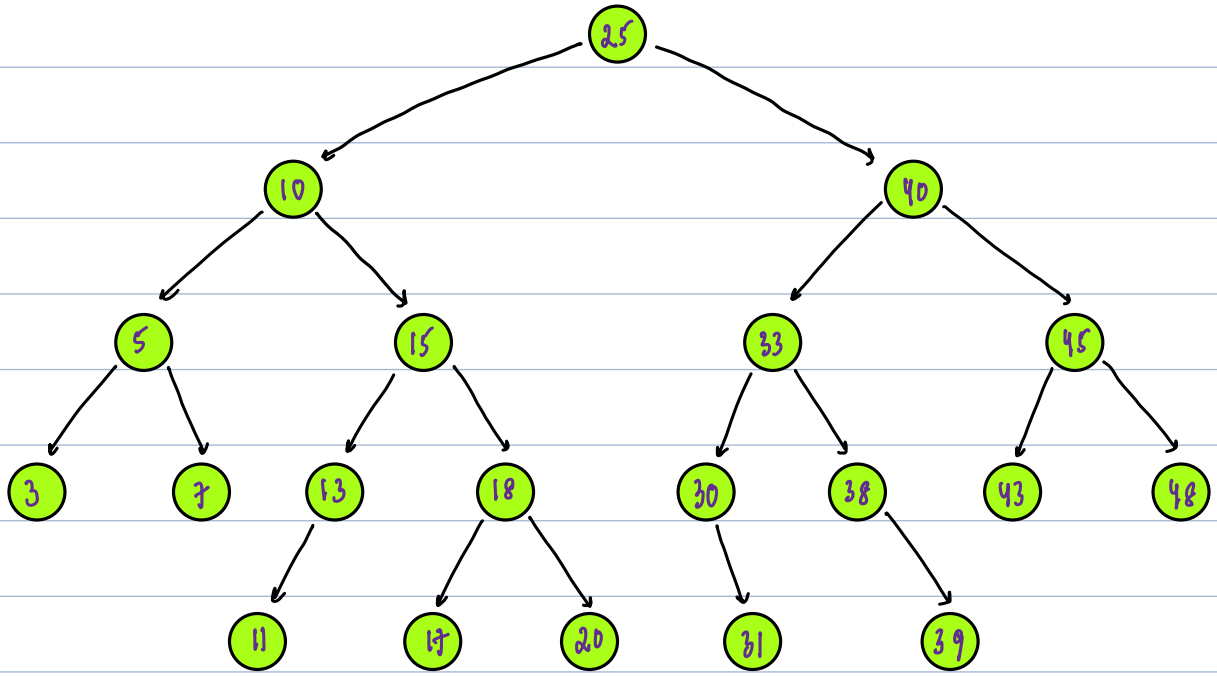
{6 - 17}



tc2 {35 - 44}



{12-37}



Ass: Given root of BST & l & h, delete all nodes not in range & return root of BST

```
Node Trim(Node h, int l, int r){
    if(root==nullptr){return nullptr;}
    root->left = Trim(root->left, l, r);
    root->right = Trim(root->right, l, r);
    if(l<=root->data && root->data<=r){
        return root;
    }
    else if(root->data<l){
        return root->right
    }
    else{
        return root->left
    }
}
```