Session - 17 :-

Inorder : 3 -1 14 4 2 17 8 15 6 4 9

Preorder : 8 -1 3 17 4 14 2 4 6 15 9

Given inorder & Preorder print the Post order

Root Ele is the first ele in Pre Order in inorder

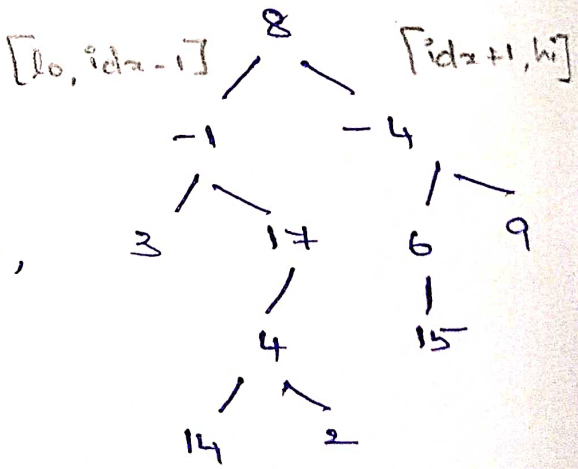
We need to find the Root and divide left and Right Recursively

(Extract the current Ele)

→ We need to have a global Variable that- the Preorder

→ We need to find the index of ele in inorder

Every thing from lo, idx-1 are left and idx+1, to high are Right



Constructing the tree.

```
int idx=0;
Node Construct(int in[], int Pre[],
               int lo, int hi) {
  if(lo > hi) return null;
  int curr = Pre[idx++];
  int cidx = find( );   ⟶ index of curr in inorder.
  Node nn = new Node(curr);
  nn.left = Construct(in, Pre, lo, cidx-1);
  nn.right = Construct(in, Pre, cidx+1, hi);
  return nn;
}
```

Construct

TC: N + N + N
populate map using in[]
Post-order

SC: N + N
map Tree

2nd Sol:-

No need to Construct a tree

```
Void Post(int in[], int Pre[], int lo, int hi) {
    if (lo > hi) return;
    int curr = Pre[idx++];
    int cid = find(---);
    Post(in, Pre, lo, cidx-1);
    Post(in, Pre, cidx+1, hi);
    Print(in[cidx]);
    return nn;
}
```

Print(in[cidx]);

Post

TC: N + N
Populate map using in[]

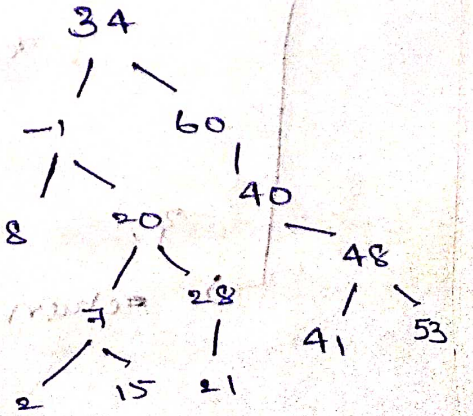
SC: N
map.

→ Given a Binary tree. Check if it is a BST

all(L) < root < all(R)

Sol1:

→ By checking Root node with max etc Value of left subtree and with min Value of Right Subtree



Solutions

1) $N^2, 1$ → Brute force
Check all max(L) < root < min(R) for all nodes

2) N+N, N

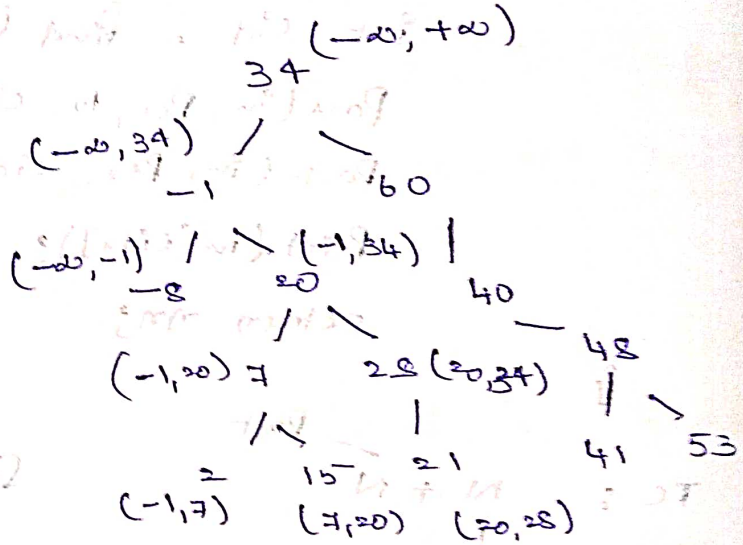
3) N, 1

4) N, 1

2nd Sol:
Inorder tree store in an array, check if sorted

3rd Sol:
no need of array check with previous

4th Sol:



Top Down approach.

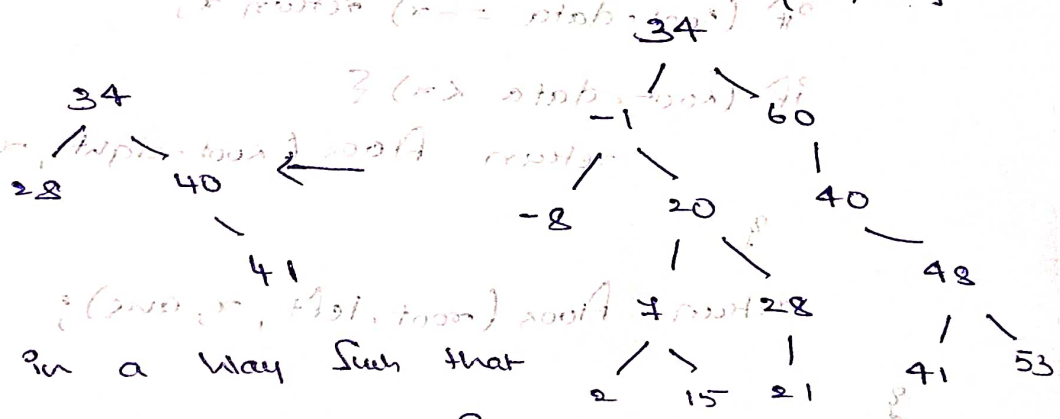
```
bool isBST (Node root, int lo, int hi) {
    if (root == null) return True;
    if (root.data >= hi || root.data <= lo)
        return false;
    return isBST(root.left, lo, root.data) &&
           isBST(root.right, root.data, hi);
}
return (root.data > lo && root.data < hi) &&
       isBST (root.left, lo, root.data) &&
       isBST (root.right, root.data, hi);
```

Code for 3rd Solution :-

```
bool ans = true;
Void inorder (Node root) {
    if (root == null) return;
    inorder(root.left);
    if (root.data > Prev) ans = false;
    Prev = root.data;
    inorder(root.right);
}
```

→ You are given a BST, Trim the BST all the elements in BST should be [a, b].

[24, 43]



Trim BST in a way such that only the range elements should present in BST.

Solutions

Brute force ← 1. N*(H), 1

date search & delete

2. N, 1

2nd Sol :-

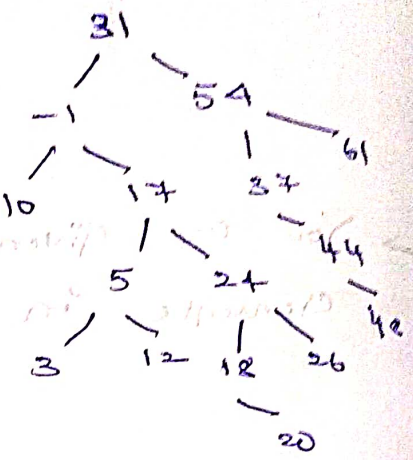
```
Node trim(Node root, int a, int b) {
    if (root == null) return null;
    if (root.data < a) return trim(root.right, a, b);
    if (root.data > b) return trim(root.left, a, b);
    root.left = trim(root.left, a, b);
    root.right = trim(root.right, a, b);
```

return root;

}

TC : N SC : 1

→ You are given BST You need to find the floor(x) in BST.



x = 28 max(ele) <= x

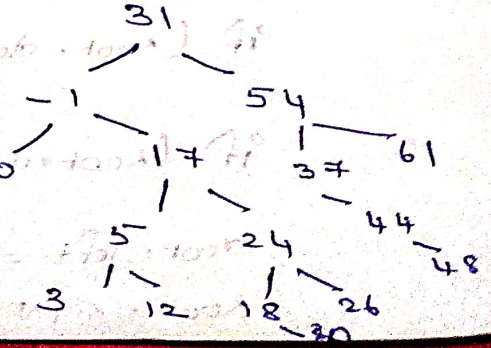
```
int floor(Node root, int x, int ans) {
    if (root == null) return ans;
    if (root.data == x) return x;
    if (root.data < x) {
        return floor(root.right, x, root.data);
    }
    return floor(root.left, x, ans);
}
```

TC : O(H) SC : O(1)

Todo: Similarly implement ceil

→ You are given 2 numbers, find their least-common ancestor in a given BST.

Ex: For 20, 26. least-common ancestor is 24



```
int LCAinBST( Node root, int a, int b) {

    if (root.data < a && root.data < b) {
    return LCAinBST (root.right, a, b);
     }
    else if (root.data > a && root.data
                                > b) {
    return LCAinBST( root.left , a , b);
     }
    else
        return root.data;
}
```