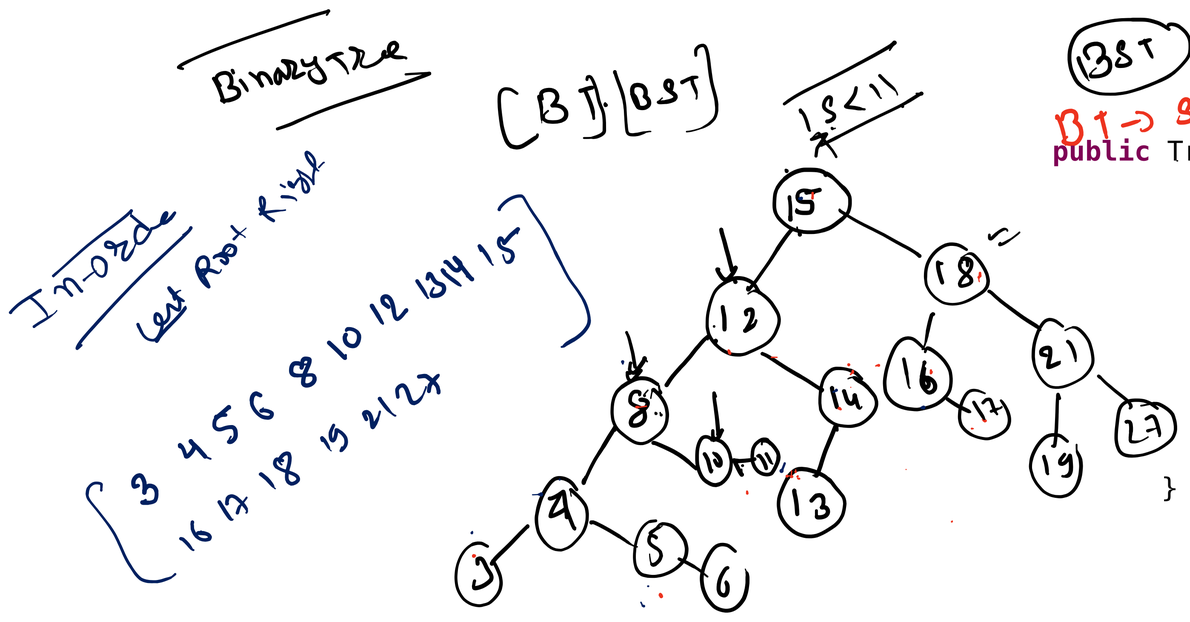
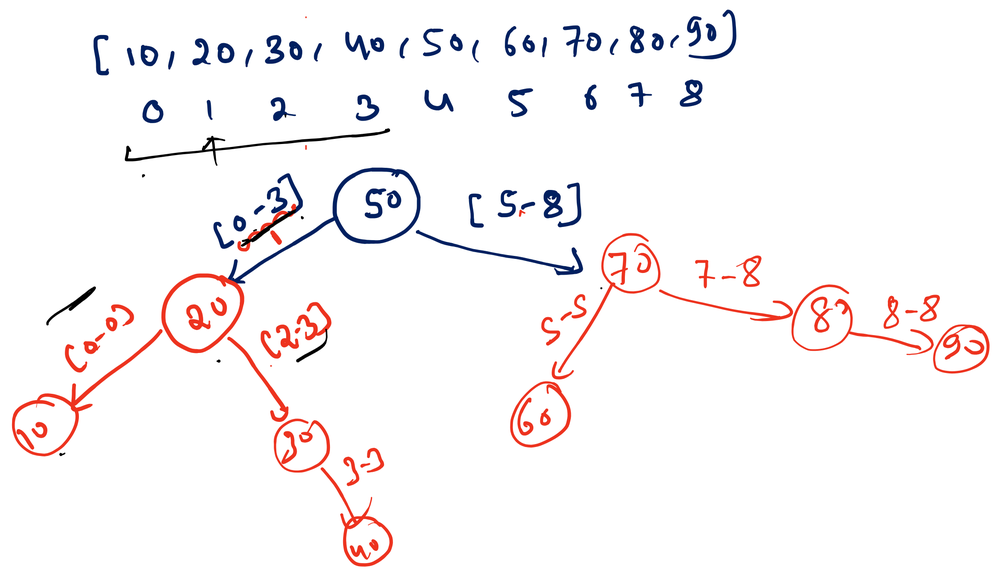
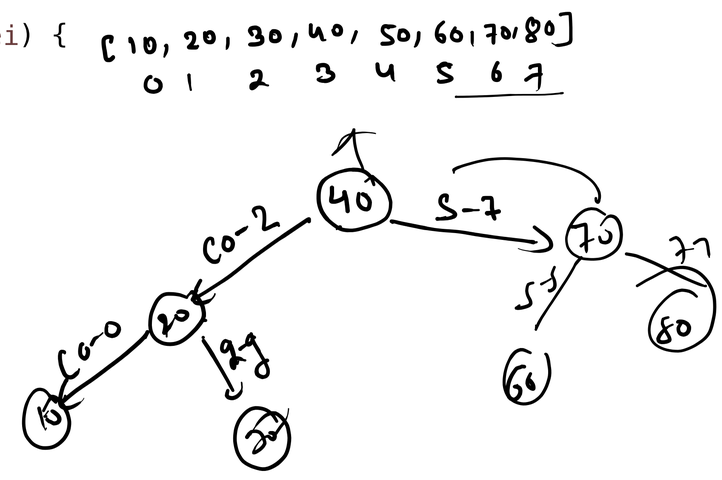
Lec-32

14/09/24 5:08 PM

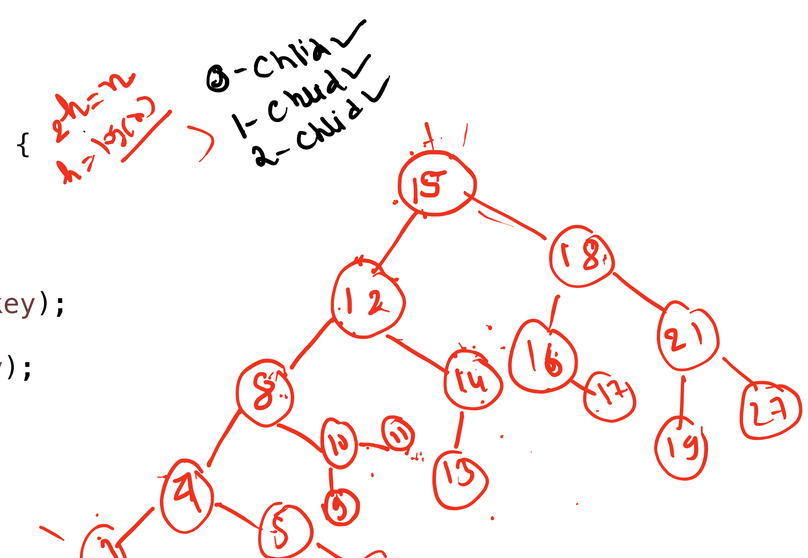




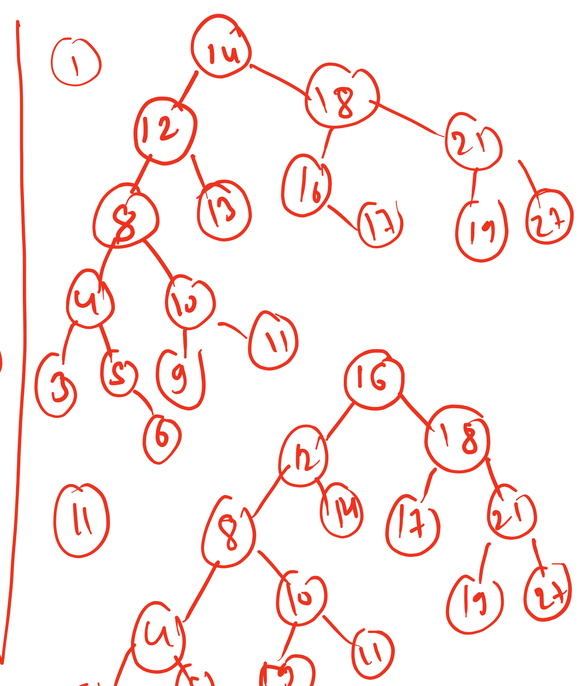

```
private Node CreateTree(int[] in, int si, int ei) {
        if (si > ei) {
            return null;
        }
        int mid = (si + ei) / 2;
        Node nn = new Node();
        nn.val = in[mid];
        nn.left = CreateTree(in, si, mid - 1);
        nn.right = CreateTree(in, mid + 1, ei);
        return nn;

    }
```



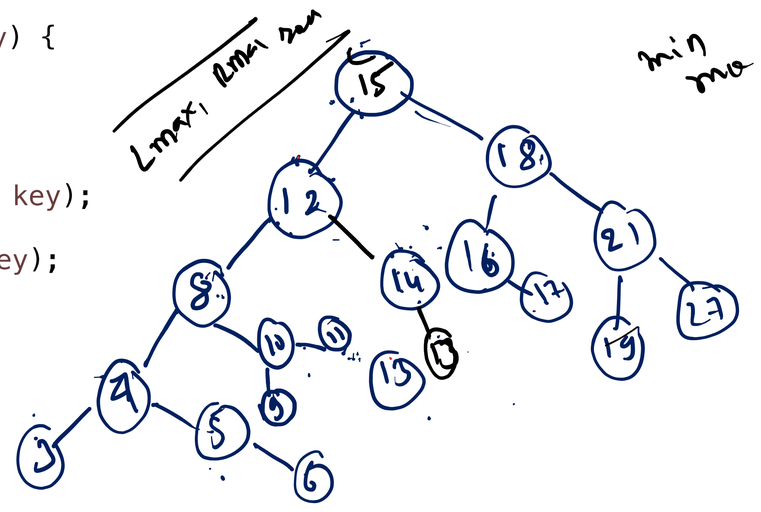

```
public TreeNode deleteNode(TreeNode root, int key) {
        if (root == null) {
            return null;
        }
        if (root.val < key) {
            root.right = deleteNode(root.right, key);
        } else if (root.val > key) {
            root.left = deleteNode(root.left, key);
        } else {
            // 1 child
            if (root.left == null) {
                return root.right;
            } else if (root.right == null) {
                return root.left;
            }

        }

    }
```




```
public int max(TreeNode root) {
        if (root == null) {
            return Integer.MIN_VALUE;
        }

        int r = max(root.right);
        return Math.max(root.val, r);

    }
```



```
public TreeNode deleteNode(TreeNode root, int key) {
        if (root == null) {
            return null;
        }
        if (root.val < key) {
            root.right = deleteNode(root.right, key);
        } else if (root.val > key) {
            root.left = deleteNode(root.left, key);
        } else {
            // 0 Or 1 child
            if (root.left == null) {
                return root.right;
            } else if (root.right == null) {
                return root.left;
            } else {
                int max = max(root.left);
                root.left = deleteNode(root.left, max);
                root.val = max;
            }

        }
        return root;
    }
```




```
public BstPair isValid(TreeNode root) {
        if (root == null) {
            return new BstPair();
        }
        BstPair lbp = isValid(root.left);
        BstPair rbp = isValid(root.right);
        BstPair sbp = new BstPair();
        sbp.max = Math.max(root.val, Math.max(lbp.max, rbp.max));
        sbp.min = Math.min(root.val, Math.min(lbp.min, rbp.min));
        sbp.isbst = lbp.isbst && rbp.isbst && lbp.max < root.val && rbp.min > root.val;
        return sbp;

    }
```