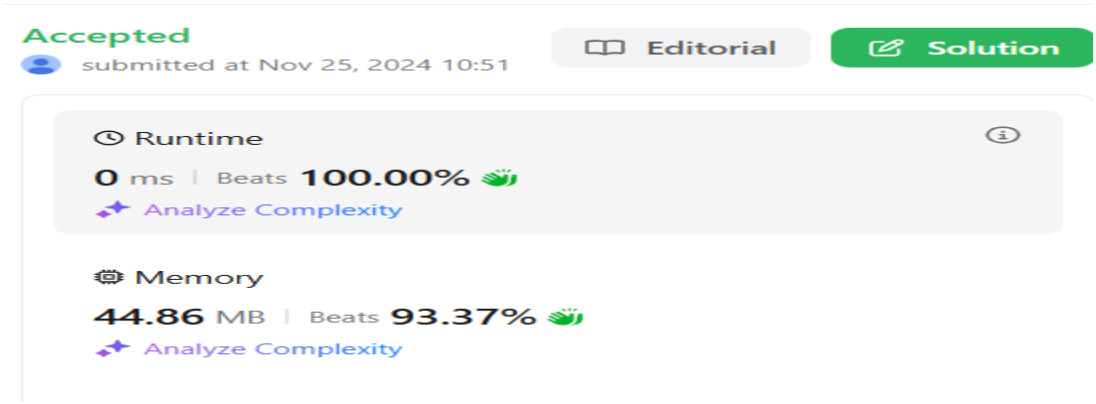
BINARY SEARCH TREE

25/11/2024

1. Search in BST:

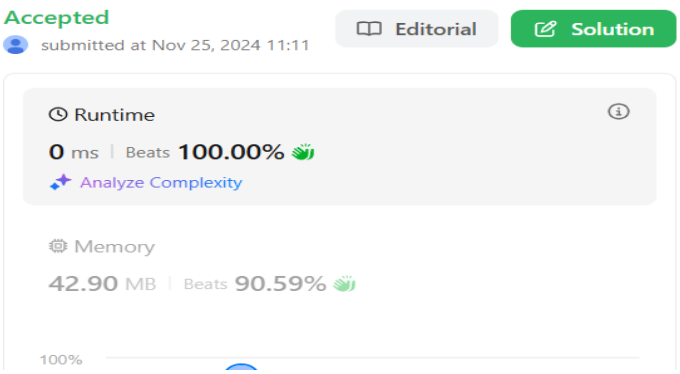
```
class Solution {
    public TreeNode searchBST(TreeNode root, int val) {
        if(root== null) return null;
        if (root.val==val) return root;
        if(root.val<val){
            return searchBST(root.right,val);
        }else{
            return searchBST(root.left,val);
        }
    }
}
```



2. VALIDATE BINARY SEARCH TREE:

```
class Solution {
    public boolean isValidBST(TreeNode root) {
        return validate(root, Long.MIN_VALUE, Long.MAX_VALUE);
    }

    private boolean validate(TreeNode node, long min, long max) {
        if (node == null) return true;
        if (node.val <= min || node.val >= max) return false;
        return validate(node.left, min, node.val) && validate(node.right,
node.val, max);
    }
}
```



3.BINARY SERACH TREE :

```
class Node {
    int value;
    Node left, right;

    Node(int value) {
        this.value = value;
        this.left = null;
        this.right = null;
    }
}

class BST {
    Node root;

    BST() {
        root = null;
    }

    public void insert(int value) {
        root = insertRec(root, value);
    }

    private Node insertRec(Node root, int value) {
        if (root == null) {
            root = new Node(value);
            return root;
        }
        if (value < root.value) {
            root.left = insertRec(root.left, value);
        } else if (value > root.value) {
            root.right = insertRec(root.right, value);
        }
```

```
        return root;
    }

    public boolean isValidBST() {
        return validate(root, Long.MIN_VALUE, Long.MAX_VALUE);
    }

    private boolean validate(Node node, long min, long max) {
        if (node == null) {
            return true;
        }
        if (node.value <= min || node.value >= max) {
            return false;
        }
        return validate(node.left, min, node.value) && validate(node.right, node.value,
max);
    }
}

public class Main {
    public static void main(String[] args) {
        BST tree = new BST();
        tree.insert(10);
        tree.insert(5);
        tree.insert(15);
        tree.insert(3);
        tree.insert(7);

        System.out.println(tree.isValidBST());
    }

}
```

Output: true

4.top view of binary search tree:

```
class Solution {
   static ArrayList<Integer> topView(Node root) {
      ArrayList<Integer> result = new ArrayList<>();
      if (root == null) return result;

      TreeMap<Integer, Integer> hdMap = new TreeMap<>();
      Queue<Pair> queue = new LinkedList<>();
      queue.add(new Pair(root, 0));

      while (!queue.isEmpty()) {
```

```
            Pair current = queue.poll();
            Node node = current.node;
            int hd = current.hd;

            if (!hdMap.containsKey(hd)) {
                hdMap.put(hd, node.data);
            }

            if (node.left != null) {
                queue.add(new Pair(node.left, hd - 1));
            }
            if (node.right != null) {
                queue.add(new Pair(node.right, hd + 1));
            }
        }

        result.addAll(hdMap.values());
        return result;
    }
}

class Pair {
    Node node;
    int hd;

    Pair(Node node, int hd) {
        this.node = node;
        this.hd = hd;
    }
}
```

Problem Solved Successfully ✔ Suggest Feedbac

Test Cases Passed	Attempts : Correct / Total
1111 / 1111	**1 / 1**
	Accuracy : 100%
Points Scored ⓘ	Time Taken
4 / 4	**0.57**
Your Total Score: 120 ↑	

5.bottom view binary search tree:

```
class Solution {
    static ArrayList<Integer> bottomView(Node root) {
```

```
        ArrayList<Integer> result = new ArrayList<>();
        if (root == null) return result;

        TreeMap<Integer, Integer> hdMap = new TreeMap<>();
        Queue<Pair> queue = new LinkedList<>();
        queue.add(new Pair(root, 0));

        while (!queue.isEmpty()) {
            Pair current = queue.poll();
            Node node = current.node;
            int hd = current.hd;

            // Update the node data for this horizontal distance
            hdMap.put(hd, node.data);

            if (node.left != null) {
                queue.add(new Pair(node.left, hd - 1));
            }
            if (node.right != null) {
                queue.add(new Pair(node.right, hd + 1));
            }
        }

        result.addAll(hdMap.values());
        return result;
    }
}

class Pair {
    Node node;
    int hd;

    Pair(Node node, int hd) {
        this.node = node;
        this.hd = hd;
    }
}
```

Test Cases Passed

1115 / 1115

Attempts : Correct / Total

2 / 2

Accuracy : 100%

Time Taken

1.23

6.LEFT VIEW OF BINARY SEARCH TREE:

```
class Solution {
    static ArrayList<Integer> leftView(Node root) {
        ArrayList<Integer> result = new ArrayList<>();
        if (root == null) return result;

        Queue<Node> queue = new LinkedList<>();
        queue.add(root);

        while (!queue.isEmpty()) {
            int levelSize = queue.size();

            for (int i = 0; i < levelSize; i++) {
                Node current = queue.poll();

                if (i == 0) {
                    result.add(current.data);
                }

                if (current.left != null) {
                    queue.add(current.left);
                }
                if (current.right != null) {
                    queue.add(current.right);
                }
            }
        }

        return result;
    }
}
```

Problem Solved Successfully

Suggest Feedbac

Test Cases Passed	Attempts : Correct / Total
1115 / 1115	**1 / 1**
	Accuracy : 100%
Points Scored	Time Taken
2 / 2	**0.77**
Your Total Score: 126 ↑	