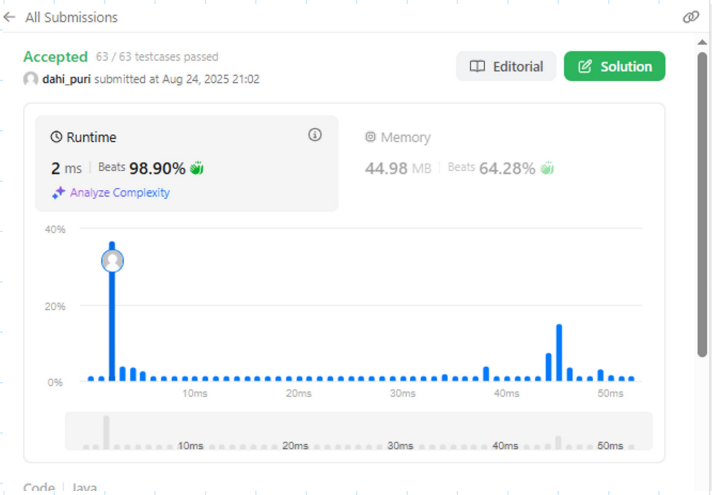
Submission of code

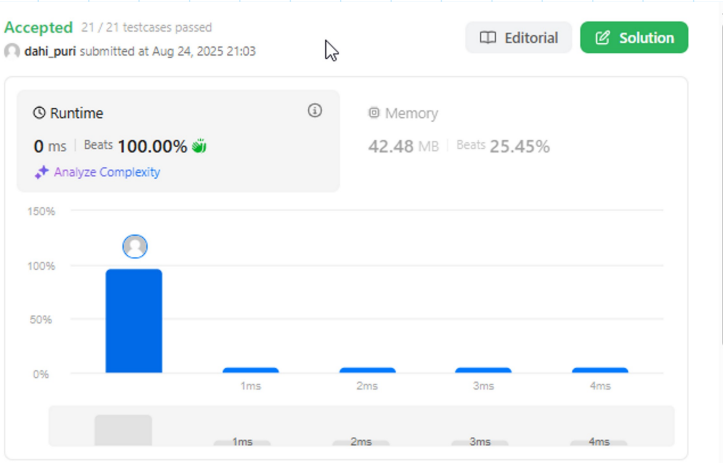
24 August 2025 21:00

Two sum




```java
class Solution {
    public int[] twoSum(int[] nums, int target) {
        //brute force
        // int n=nums.length;
        // for(int i=0;i<n;i++){
        //     for(int j=i+1;j<n;j++){
        //         if(nums[i]+nums[j]==target) return new int[]{i,j};
        //     }
        // }
        // return new int[]{-1,-1};

        //optimal
        int n=nums.length;
        HashMap<Integer,Integer> map=new HashMap<>();
        for(int i=0;i<n;i++){
            if(map.containsKey(target-nums[i])) return new int[]{i,map.get(target-nums[i])};
            map.put(nums[i],i);
        }
        return new int[]{-1,-1};

    }
}
```

Rotate image




```java
class Solution {
    public void rotate(int[][] matrix) {
        int n = matrix.length;


        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
                int temp = matrix[i][j];
                matrix[i][j] = matrix[j][i];
                matrix[j][i] = temp;
            }
        }


        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n / 2; j++) {
                int temp = matrix[i][j];
                matrix[i][j] = matrix[i][n - 1 - j];
                matrix[i][n - 1 - j] = temp;
            }
        }
    }
}
```

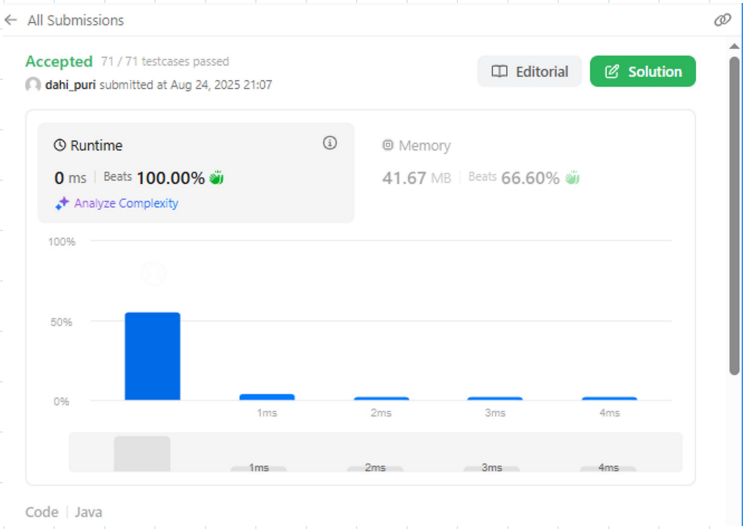
Saved

Inorder


```
*         this.right = right;
*     }
* }
*/
class Solution {
    public List<Integer> inorderTraversal(TreeNode root) {
        List<Integer> ans = new ArrayList<>();
        Deque<TreeNode> stack = new ArrayDeque<>();
        TreeNode cur = root;

        while (cur != null || !stack.isEmpty()) {

            while (cur != null) {
                stack.push(cur);
                cur = cur.left;
            }

            cur = stack.pop();
            ans.add(cur.val);

            cur = cur.right;
        }

        return ans;
    }
}
```

Saved Ln

Postorder




```
Java  Auto
*         this.left = left;
*         this.right = right;
*     }
* }
*/
class Solution {
    public List<Integer> postorderTraversal(TreeNode root) {
        LinkedList<Integer> res = new LinkedList<>();
        if (root == null) return res;

        Stack<TreeNode> stack = new Stack<>();
        stack.push(root);

        while (!stack.isEmpty()) {
            TreeNode node = stack.pop();
            res.addFirst(node.val);  // reverse order by adding at front
            if (node.left != null) stack.push(node.left);
            if (node.right != null) stack.push(node.right);
        }
        return res;
    }
}
```

Preorder

All Submissions

Accepted 71 / 71 testcases passed

dahi_puri submitted at Aug 24, 2025 21:08

Editorial | Solution

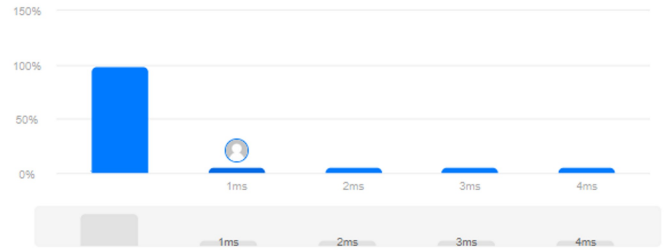
Runtime

1 ms | Beats 1.48%

Analyze Complexity

Memory

41.75 MB | Beats 45.38%



Code | Java

Java | Auto

```
*         this.right = right;
*     }
* }
*/
class Solution {
    public List<Integer> preorderTraversal(TreeNode root) {
        List<Integer> res = new ArrayList<>();
        if (root == null) return res;

        Stack<TreeNode> stack = new Stack<>();
        stack.push(root);

        while (!stack.isEmpty()) {
            TreeNode node = stack.pop();
            res.add(node.val);

            if (node.right != null) stack.push(node.right);
            if (node.left != null) stack.push(node.left);
        }
        return res;
    }
}
```