

## Submission of code

24 August 2025 21:00

### Two sum

← All Submissions

Accepted 63 / 63 testcases passed

dahi\_puri submitted at Aug 24, 2025 21:02

Editorial

Solution

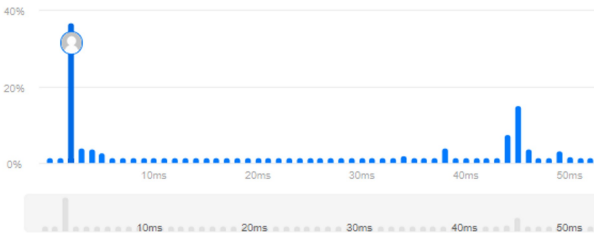
Runtime

2 ms | Beats 98.90%

Analyze Complexity

Memory

44.98 MB | Beats 64.28%



Code | Java

Java Auto

```
1 class Solution {
2     public int[] twoSum(int[] nums, int target) {
3         //brute force
4         // int n=nums.length;
5         // for(int i=0;i<n;i++){
6         //     for(int j=i+1;j<n;j++){
7         //         if(nums[i]+nums[j]==target) return new int[]{i,j};
8         //     }
9         // }
10        // return new int[]{-1,-1};
11
12        //optimal
13        int n=nums.length;
14        HashMap<Integer,Integer> map=new HashMap<>();
15        for(int i=0;i<n;i++){
16            if(map.containsKey(target-nums[i])) return new int[]{i,map.get(target-nums[i])};
17            map.put(nums[i],i);
18        }
19        return new int[]{-1,-1};
20    }
21 }
22 }
```

### Rotate image

Accepted 21 / 21 testcases passed

dahi\_puri submitted at Aug 24, 2025 21:03

Editorial

Solution

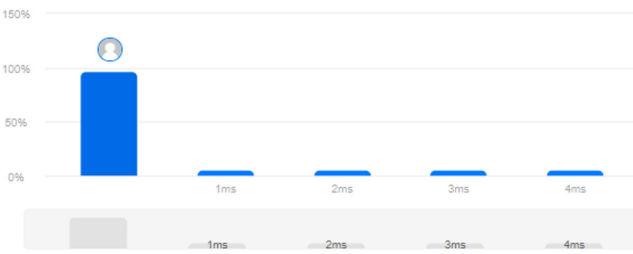
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

42.48 MB | Beats 25.45%



Code | Java

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

```
1 class Solution {
2     public void rotate(int[][] matrix) {
3         int n = matrix.length;
4
5
6         for (int i = 0; i < n; i++) {
7             for (int j = i; j < n; j++) {
8                 int temp = matrix[i][j];
9                 matrix[i][j] = matrix[j][i];
10                matrix[j][i] = temp;
11            }
12        }
13
14        for (int i = 0; i < n; i++) {
15            for (int j = 0; j < n / 2; j++) {
16                int temp = matrix[i][j];
17                matrix[i][j] = matrix[i][n - 1 - j];
18                matrix[i][n - 1 - j] = temp;
19            }
20        }
21    }
22 }
23 }
24 }
```

Saved

### Inorder

Accepted 71 / 71 testcases passed  
dahi\_puri submitted at Aug 24, 2025 21:06

Editorial

Solution

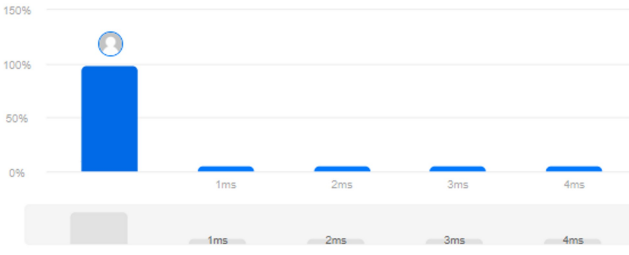
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

41.56 MB | Beats 83.37%



Code | Java

```
/**  
 * Definition for a binary tree node.
```

```
12 *         this.right = right;  
13 *     }  
14 * }  
15 */  
16 class Solution {  
17     public List<Integer> inorderTraversal(TreeNode root) {  
18         List<Integer> ans = new ArrayList<>();  
19         Deque<TreeNode> stack = new ArrayDeque<>();  
20         TreeNode cur = root;  
21  
22         while (cur != null || !stack.isEmpty()) {  
23  
24             while (cur != null) {  
25                 stack.push(cur);  
26                 cur = cur.left;  
27             }  
28  
29             cur = stack.pop();  
30             ans.add(cur.val);  
31  
32             cur = cur.right;  
33         }  
34  
35         return ans;  
36     }  
37 }
```

Saved

## Postorder

All Submissions

Accepted 71 / 71 testcases passed  
dahi\_puri submitted at Aug 24, 2025 21:07

Editorial

Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

41.67 MB | Beats 66.60%



Code | Java

```
11 *         this.left = left;  
12 *         this.right = right;  
13 *     }  
14 * }  
15 */  
16 class Solution {  
17     public List<Integer> postorderTraversal(TreeNode root) {  
18         LinkedList<Integer> res = new LinkedList<>();  
19         if (root == null) return res;  
20  
21         Stack<TreeNode> stack = new Stack<>();  
22         stack.push(root);  
23  
24         while (!stack.isEmpty()) {  
25             TreeNode node = stack.pop();  
26             res.addFirst(node.val); // reverse order by adding at front  
27             if (node.left != null) stack.push(node.left);  
28             if (node.right != null) stack.push(node.right);  
29         }  
30         return res;  
31     }  
32 }  
33 }
```

Accepted 71 / 71 testcases passed

dahi\_puri submitted at Aug 24, 2025 21:08

Editorial

Solution

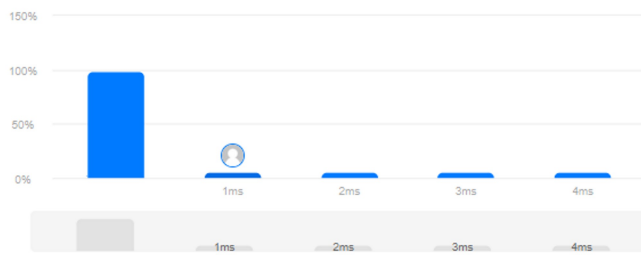
## Runtime

1 ms | Beats 1.48%

[Analyze Complexity](#)

## Memory

41.75 MB | Beats 45.38%



Code | Java

```
12 *         this.right = right;
13 *     }
14 * }
15 */
16 class Solution {
17     public List<Integer> preorderTraversal(TreeNode root) {
18         List<Integer> res = new ArrayList<>();
19         if (root == null) return res;
20
21         Stack<TreeNode> stack = new Stack<>();
22         stack.push(root);
23
24         while (!stack.isEmpty()) {
25             TreeNode node = stack.pop();
26             res.add(node.val);
27
28             if (node.right != null) stack.push(node.right);
29             if (node.left != null) stack.push(node.left);
30         }
31         return res;
32     }
33 }
34
```