

## Output Window

### Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

### Problem Solved Successfully ✓

[Suggest Feedback](#)

Test Cases Passed

**1112 / 1112**

Attempts : Correct / Total

**1 / 1**

Accuracy : 100%

Points Scored ⓘ

**2 / 2**

Your Total Score: 39 ↑

Time Taken

**0.38**

### Solve Next

Sorted subsequence of size 3

Array Duplicates

Two Sum - Pair with Given Sum

### Stay Ahead With:

```
1 // User function Template for Java
2
3 class Solution {
4
5     public static int smallestSubwithSum(int x, int[] arr) {
6         // Your code goes here
7         int n = arr.length;
8         int left = 0;
9         int right = 0;
10        int sum = 0;
11        int minLen = Integer.MAX_VALUE;
12
13        while(right < n) {
14            sum += arr[right];
15
16            while(sum > x) {
17                int len = right-left+1;
18                minLen = Math.min(minLen, len);
19
20                sum -= arr[left];
21                left++;
22            }
23
24            right++;
25        }
26
27        if(minLen == Integer.MAX_VALUE) return 0;
28
29        return minLen;
30    }
31 }
```



Custom Input

Compile & Run

Submit

Ctrl + Enter

## Output Window

### Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

### Problem Solved Successfully ✓

[Suggest Feedback](#)

Test Cases Passed

**1112 / 1112**

Attempts : Correct / Total

**1 / 1**

Accuracy : 100%

Points Scored ⓘ

**2 / 2**

Your Total Score: 37 ↑

Time Taken

**0.77**

### Solve Next

Bubble Sort

Floor in a Sorted Array

Closest Triplet

### Stay Ahead With:

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3
4 class Solution {
5     public int findMinDiff(ArrayList<Integer> arr, int m) {
6         // Sort the array
7         Collections.sort(arr);
8
9         int start = 0;
10        int end = m - 1;
11        int ans = Integer.MAX_VALUE;
12
13        while (end < arr.size()) {
14            ans = Math.min(ans, arr.get(end) - arr.get(start));
15            start++;
16            end++;
17        }
18
19        return ans;
20    }
21 }
22
23
```



Custom Input

Compile & Run

Ctrl + Enter

Submit

All Submissions

Accepted 172 / 172 testcases passed

SHREYA RAJ submitted at Feb 17, 2026 13:43

Editorial
 Solution



Code
 Java

```

1 class Solution {
2     public int[][] merge(int[][] intervals) {
3         Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
4
5         List<int[]> merged = new ArrayList<>();
    
```

Code

Java
 Auto

```

1 public int[][] merge(int[][] intervals) {
2     Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
3
4     List<int[]> merged = new ArrayList<>();
5     int[] prev = intervals[0];
6
7     for (int i = 1; i < intervals.length; i++) {
8         int[] interval = intervals[i];
9         if (interval[0] <= prev[1]) {
10             prev[1] = Math.max(prev[1], interval[1]);
11         } else {
12             merged.add(prev);
13             prev = interval;
14         }
15     }
16     merged.add(prev);
17     return merged.toArray(new int[merged.size()][]);
18 }
    
```

Saved
 Ln 22, Col 2

Testcase
 Test Result

Accepted Runtime: 1 ms

Case 1 Case 2 Case 3

Input

intervals =  
 [[1,3],[2,6],[8,10],[15,18]]

Output

[[1,6],[8,10],[15,18]]

Expected

Description Editorial Solutions Submissions

Follow up:

- How can we prove that at least one duplicate number must exist in `nums`?
- Can you solve the problem in linear runtime complexity?

over  
re

Seen this question in a real interview before? 1/5

Yes No

Accepted 2,475,173 / 3.9M Acceptance Rate 64.0%

Topics

Companies

Similar Questions

Discussion (498)

Copyright © 2026 LeetCode. All rights reserved.

25.3K 498

165 Online

<

</> Code

Java Auto

```
1 class Solution {
2     public int findDuplicate(int[] nums) {
3         int slow = nums[0];
4         int fast = nums[0];
5
6         while (true) {
7             slow = nums[slow];
8             fast = nums[nums[fast]];
9
10            if (slow == fast) {
```

Saved

Ln 24, Col 2

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =  
[1,3,4,2,2]

Output

2

Expected

## Output Window

### Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

### Problem Solved Successfully

[Suggest Feedback](#)

Test Cases Passed

**1111 / 1111**

Attempts : Correct / Total

**1 / 1**

Accuracy : 100%

Points Scored 

**4 / 4**

Your Total Score: 35 

Time Taken

**0.79**

### Solve Next

Median of 2 Sorted Arrays of Different Sizes

Nth Natural Number

Smallest Positive Integer that can not be represented as Sum

Can Ahead With:

```
1 class Solution {
2     public void mergeArrays(int a[], int b[]) {
3         int n = a.length;
4         int m = b.length;
5
6         PriorityQueue<Integer> pq = new PriorityQueue<>();
7
8         for(int i = 0; i < m; i++){
9             pq.add(b[i]);
10        }
11
12        for(int i = 0; i < n; i++){
13            if(pq.peek() < a[i]){
14                pq.add(a[i]);
15                a[i] = pq.poll();
16            }
17        }
18
19        for(int i = 0 ; i < m; i++){
20            b[i] = pq.poll();
21        }
22    }
23 }
24
25
```



Custom Input

Compile & Run

Submit

Ctrl + Enter

## Output Window

### Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

### Problem Solved Successfully ✓

[Suggest Feedback](#)

Test Cases Passed

**1215 / 1215**

Attempts : Correct / Total

**2 / 4**

Accuracy : 50%

Time Taken

**5.36**



You get marks only for the first correct submission if you solve the problem without viewing the full solution.

### Solve Next

Two Repeated Elements

Sorted and Rotated Minimum

Sorted Insert Position

```
1 class Solution {
2     // Function to find common elements in three arrays.
3     public List<Integer> commonElements(List<Integer> arr1, List<Integer> arr2,
4                                         List<Integer> arr3) {
5         Map<Integer,Integer> mp = new TreeMap<>();
6
7         HashSet<Integer> h1 = new HashSet<>(arr1);
8         HashSet<Integer> h2 = new HashSet<>(arr2);
9         HashSet<Integer> h3 = new HashSet<>(arr3);
10        for(int i :h1){
11            mp.put(i,mp.getDefault(i,0)+1);
12        }
13        for(int i :h2){
14            mp.put(i,mp.getDefault(i,0)+1);
15        }
16
17        for(int i :h3){
18            mp.put(i,mp.getDefault(i,0)+1);
19        }
20
21        List<Integer> res= new ArrayList<>();
22        for(Map.Entry<Integer,Integer> entry: mp.entrySet()){
23            if(entry.getValue()==3){
24                res.add(entry.getKey());
25            }
26        }
27        return res;
28    }
29 }
30
31
```



Custom Input

Compile & Run

Ctrl + Enter

Submit



## Output Window

### Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

### Problem Solved Successfully

[Suggest Feedback](#)

Test Cases Passed

**1114 / 1114**

Attempts : Correct / Total

**1 / 1**

Accuracy : 100%

Points Scored 

**1 / 1**

Your Total Score: 29 

Time Taken

**0.71**

### Solve Next

Counting elements in two arrays

Union of 2 Sorted Arrays

Left most and right most index

```
1
2 class Solution {
3     public boolean isSubset(int a[], int b[]) {
4         // Your code here
5         Arrays.sort(a);
6         Arrays.sort(b);
7         int ai = 0, bi = 0;
8         while(ai < a.length && bi < b.length){
9             if(a[ai] == b[bi])
10                bi++;
11            ai++;
12        }
13        if(bi == b.length)
14            return true;
15        return false;
16    }
17 }
```

Ctrl + Enter

Custom Input

Compile & Run

Submit

## Output Window

### Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

### Problem Solved Successfully ✓

[Suggest Feedback](#)

Test Cases Passed

**1111 / 1111**

Attempts : Correct / Total

**1 / 1**

Accuracy : 100%

Points Scored ⓘ

**4 / 4**

Your Total Score: 28 ↑

Time Taken

**0.24**

### Solve Next

Sort Elements by Decreasing Frequency

Zero Sum Subarrays

Triplets with Smaller Sum

```
1 class Solution {
2     public boolean hasTripletSum(int arr[], int target) {
3
4         Arrays.sort(arr);
5
6         for(int i=0;i<arr.length-2;i++){
7             int left=i+1;
8             int right=arr.length-1;
9
10            while(left<right){
11                int sum=arr[i]+arr[left]+arr[right];
12                if(sum==target){
13                    return true;
14                }else if(sum<target){
15                    left++;
16                }else{
17                    right--;
18                }
19            }
20        }
21        return false;
22    }
23 }
24 }
```



Custom Input

Compile & Run

Ctrl + Enter

Submit



## Output Window

### Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

### Problem Solved Successfully ✓

[Suggest Feedback](#)

Test Cases Passed

**1111 / 1111**

Attempts : Correct / Total

**1 / 2**

Accuracy : 50%

Points Scored ⓘ

**8 / 8**

Your Total Score: 24 ↑

Time Taken

**0.25**

### Solve Next

[Longest Arithmetic Subsequence](#)

[Rod Cutting](#)

[Jump Game](#)

### Stay Ahead With:

```
1 • class Solution {
2 •     public int maxWater(int a[]) {
3
4         int n=a.length;
5         if(n==0)return 0;
6         int res=0;
7         int[] lm=new int[n];
8         lm[0]=a[0];
9         for(int i=1;i<n;i++){
10             lm[i]=Math.max(lm[i-1],a[i]);
11         }
12         int rm[]=new int[n];
13         rm[n-1]=a[n-1];
14         for(int i=n-2;i>=0;i--){
15             rm[i]=Math.max(rm[i+1],a[i]);
16         }
17         for(int i=0;i<n;i++){
18             res+=Math.min(lm[i],rm[i])-a[i];
19         }
20         return res;
21     }
22 }
```



[Custom Input](#)

[Compile & Run](#)

[Submit](#)

Ctl + Enter

## Output Window

### Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

### Problem Solved Successfully

[Suggest Feedback](#)

Test Cases Passed

**1111 / 1111**

Attempts : Correct / Total

**1 / 3**

Accuracy : 33%

Points Scored 

**4 / 4**

Your Total Score: 16 

Time Taken

**0.66**

### Solve Next

Large Factorial

Number following a pattern

Rank The Permutations

### Stay Ahead With:

```
1 import java.math.BigInteger;
2
3 class Solution {
4     public static ArrayList<Integer> factorial(int n) {
5         ArrayList<Integer> list = new ArrayList<>();
6         BigInteger fac = fact(n);
7         String s = fac.toString();
8
9         for(char c : s.toCharArray()){
10             list.add(c - '0');
11         }
12         return list;
13     }
14
15     public static BigInteger fact(int n){
16         BigInteger res = BigInteger.ONE;
17         for(int i=2; i<=n; i++){
18             res = res.multiply(BigInteger.valueOf(i));
19         }
20         return res;
21     }
22 }
```



Custom Input

Compile & Run

Ctrl + Enter

Submit

## Output Window

[Compilation Results](#) [Custom Input](#) [Y.O.G.I. \(AI Bot\)](#)

Problem Solved Successfully 

[Suggest Feedback](#)

Test Cases Passed

1120 / 1120

Attempts : Correct / Total

1 / 14

Accuracy : 7%

Points Scored 

4 / 4

Time Taken

0.69

Your Total Score: 12 

Stay Ahead With:



### Build 21 Projects in 21 Days

Build real-world ML, Deep Learning & Gen AI projects

[Register Now](#)

```
1 class Solution {
2     public int minJumps(int[] arr) {
3         int n = arr.length;
4         if (n <= 1) return 0;
5         if (arr[0] == 0) return -1;
6
7         int maxReach = arr[0];
8         int steps = arr[0];
9         int jumps = 1;
10
11         for (int i = 1; i < n; i++) {
12             if (i == n - 1) return jumps;
13
14             maxReach = Math.max(maxReach, i + arr[i]);
15             steps--;
16
17             if (steps == 0) {
18                 jumps++;
19                 if (i >= maxReach) return -1;
20                 steps = maxReach - i;
21             }
22         }
23         return -1;
24     }
25 }
```



Custom Input

Compile & Run

Ctrl + Enter

Submit