Rotate Array by One

Difficulty: **Basic** Accuracy: **69.6%** Submissions: **354K+** Points: **1** Average Time: **20m**

Given an array **arr**, rotate the array by one position in clockwise direction.

Examples:

Input: arr[] = [1, 2, 3, 4, 5]
Output: [5, 1, 2, 3, 4]
Explanation: If we rotate arr by one position in clockwise 5 come to the front and remaining those are shifted to the end.

Input: arr[] = [9, 8, 7, 6, 4, 2, 1, 3]
Output: [3, 9, 8, 7, 6, 4, 2, 1]
Explanation: After rotating clock-wise 3 comes in first position.

Constraints:

$1 <= arr.size() <= 10^5$

$0 <= arr[i] <= 10^5$

Try more examples

Expected Complexities

Topic Tags

Related Articles

Report An Issue

Java (21) Start Timer

```java
// // User function Template for Java

class Solution {
    public void rotate(int[] arr) {
        // code here
        int temp = arr[arr.length-1];
        //it should be initialise from the last element otherwise it overwr
        for( int i=arr.length-1; i>0; i--){
            arr[i] = arr[i-1];

        }
        arr[0] = temp;
    }
}
```

Custom Input Compile & Run Submit

Problem | Editorial | Submissions | Comments

Kadane's Algorithm

Difficulty: **Medium** Accuracy: **36.28%** Submissions: **1.2M** Points: **4** Average Time: **20m**

You are given an integer array **arr[]**. You need to find the **maximum** sum of a subarray (containing at least one element) in the array **arr[]**.

Note : A **subarray** is a continuous part of an array.

Examples:

Input: arr[] = [2, 3, -8, 7, -1, 2, 3]
Output: 11
Explanation: The subarray [7, -1, 2, 3] has the largest sum 11.

Input: arr[] = [-2, -4]
Output: -2
Explanation: The subarray [-2] has the largest sum -2.

Input: arr[] = [5, 4, 1, 7, 8]
Output: 25
Explanation: The subarray [5, 4, 1, 7, 8] has the largest sum 25.

Constraints:

$1 \leq arr.size() \leq 10^5$

$-10^4 \leq arr[i] \leq 10^4$

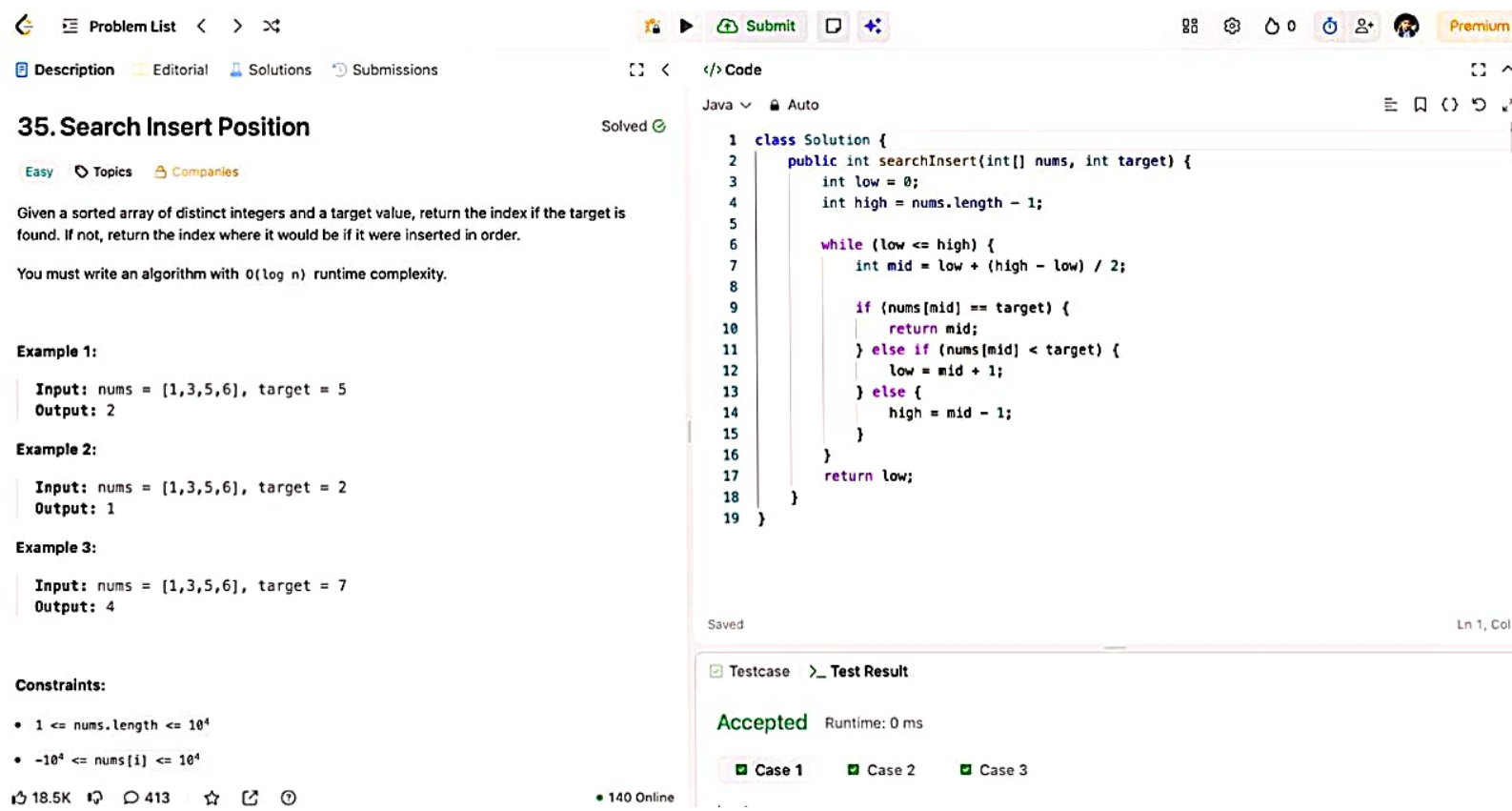
Try more examples

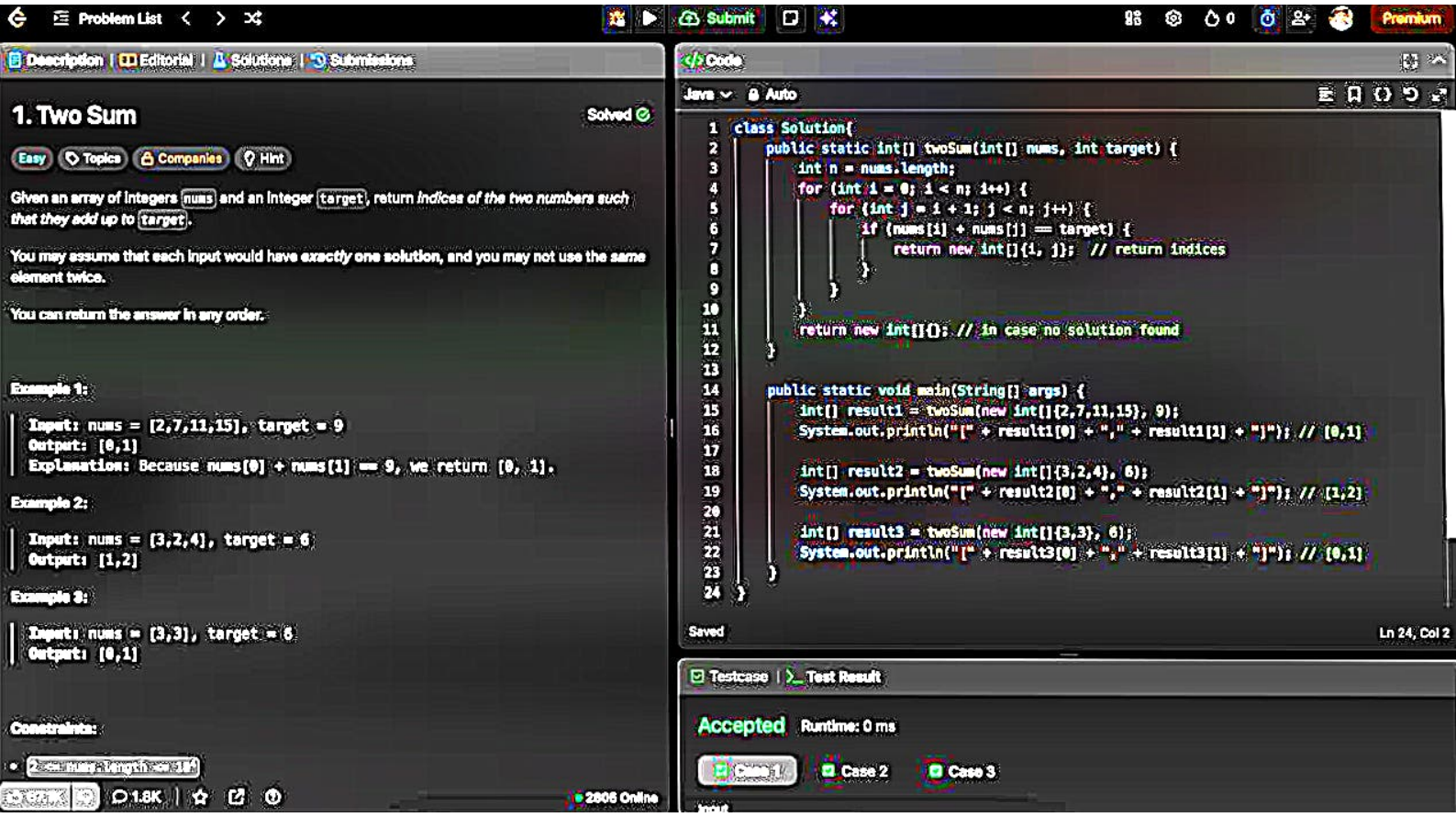
Expected Complexities

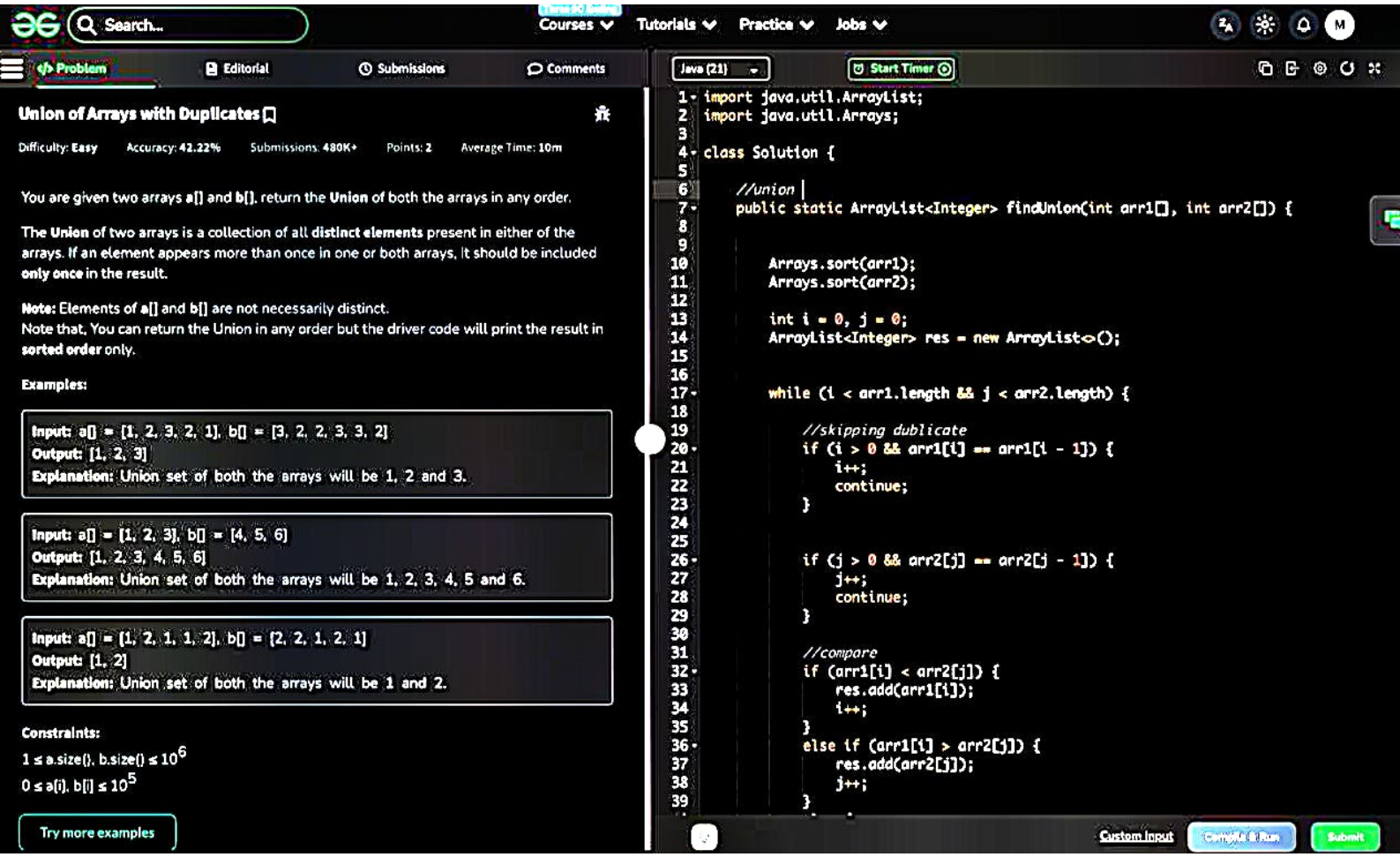
Java (21) Start Timer

```java
class Solution {
    int maxSubarraySum(int[] arr) {
        // Code here
        int sum =0;
        int max = arr[0];
        for(int i=0; i<arr.length; i++){
            sum +=arr[i];
            if(sum>max){
                max=sum;
            }
            if(sum<0){
                sum =0;
            }

        }
         return max;

    }
}

```

Custom Input Compile & Run Submit


Problem List
Submit
Premium
Description
Editorial
Solutions
Submissions
35. Search Insert Position
Solved
Easy
Topics
Companies
Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.
You must write an algorithm with O(log n) runtime complexity.
Example 1:
Input: nums = [1,3,5,6], target = 5
Output: 2
Example 2:
Input: nums = [1,3,5,6], target = 2
Output: 1
Example 3:
Input: nums = [1,3,5,6], target = 7
Output: 4
Constraints:
1 <= nums.length <= 10^4
-10^4 <= nums[i] <= 10^4
18.5K
413
140 Online
Code
Java
Auto
class Solution {
public int searchInsert(int[] nums, int target) {
int low = 0;
int high = nums.length - 1;
while (low <= high) {
int mid = low + (high - low) / 2;
if (nums[mid] == target) {
return mid;
} else if (nums[mid] < target) {
low = mid + 1;
} else {
high = mid - 1;
}
}
return low;
}
}
Saved
Ln 1, Col
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3


Problem List
Submit
Premium
Description | Editorial | Solutions | Submissions
1. Two Sum
Solved
Easy
Topics
Companies
Hint
Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.
You may assume that each input would have exactly one solution, and you may not use the same element twice.
You can return the answer in any order.
Example 1:
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
Example 2:
Input: nums = [3,2,4], target = 6
Output: [1,2]
Example 3:
Input: nums = [3,3], target = 6
Output: [0,1]
Constraints:
1.8K
2806 Online
Code
Java
Auto
1 class Solution{
2 public static int[] twoSum(int[] nums, int target) {
3 int n = nums.length;
4 for (int i = 0; i < n; i++) {
5 for (int j = i + 1; j < n; j++) {
6 if (nums[i] + nums[j] == target) {
7 return new int[]{i, j}; // return indices
8 }
9 }
10 }
11 return new int[]{}; // in case no solution found
12 }
13
14 public static void main(String[] args) {
15 int[] result1 = twoSum(new int[]{2,7,11,15}, 9);
16 System.out.println("[" + result1[0] + "," + result1[1] + "]"); // [0,1]
17
18 int[] result2 = twoSum(new int[]{3,2,4}, 6);
19 System.out.println("[" + result2[0] + "," + result2[1] + "]"); // [1,2]
20
21 int[] result3 = twoSum(new int[]{3,3}, 6);
22 System.out.println("[" + result3[0] + "," + result3[1] + "]"); // [0,1]
23 }
24 }
Saved
Ln 24, Col 2
Testcase | Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3


Search...
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Union of Arrays with Duplicates
Difficulty: Easy
Accuracy: 42.22%
Submissions: 480K+
Points: 2
Average Time: 10m
You are given two arrays a[] and b[], return the Union of both the arrays in any order.
The Union of two arrays is a collection of all distinct elements present in either of the arrays. If an element appears more than once in one or both arrays, it should be included only once in the result.
Note: Elements of a[] and b[] are not necessarily distinct.
Note that, You can return the Union in any order but the driver code will print the result in sorted order only.
Examples:
Input: a[] = [1, 2, 3, 2, 1], b[] = [3, 2, 2, 3, 3, 2]
Output: [1, 2, 3]
Explanation: Union set of both the arrays will be 1, 2 and 3.
Input: a[] = [1, 2, 3], b[] = [4, 5, 6]
Output: [1, 2, 3, 4, 5, 6]
Explanation: Union set of both the arrays will be 1, 2, 3, 4, 5 and 6.
Input: a[] = [1, 2, 1, 1, 2], b[] = [2, 2, 1, 2, 1]
Output: [1, 2]
Explanation: Union set of both the arrays will be 1 and 2.
Constraints:
1 ≤ a.size(), b.size() ≤ 10^6
0 ≤ a[i], b[i] ≤ 10^5
Try more examples
Java (21)
Start Timer
1 import java.util.ArrayList;
2 import java.util.Arrays;
3
4 class Solution {
5
6 //union
7 public static ArrayList<Integer> findUnion(int arr1[], int arr2[]) {
8
9
10 Arrays.sort(arr1);
11 Arrays.sort(arr2);
12
13 int i = 0, j = 0;
14 ArrayList<Integer> res = new ArrayList<>();
15
16
17 while (i < arr1.length && j < arr2.length) {
18
19 //skipping dublicate
20 if (i > 0 && arr1[i] == arr1[i - 1]) {
21 i++;
22 continue;
23 }
24
25
26 if (j > 0 && arr2[j] == arr2[j - 1]) {
27 j++;
28 continue;
29 }
30
31 //compare
32 if (arr1[i] < arr2[j]) {
33 res.add(arr1[i]);
34 i++;
35 }
36 else if (arr1[i] > arr2[j]) {
37 res.add(arr2[j]);
38 j++;
39 }
Custom Input
Compile & Run
Submit