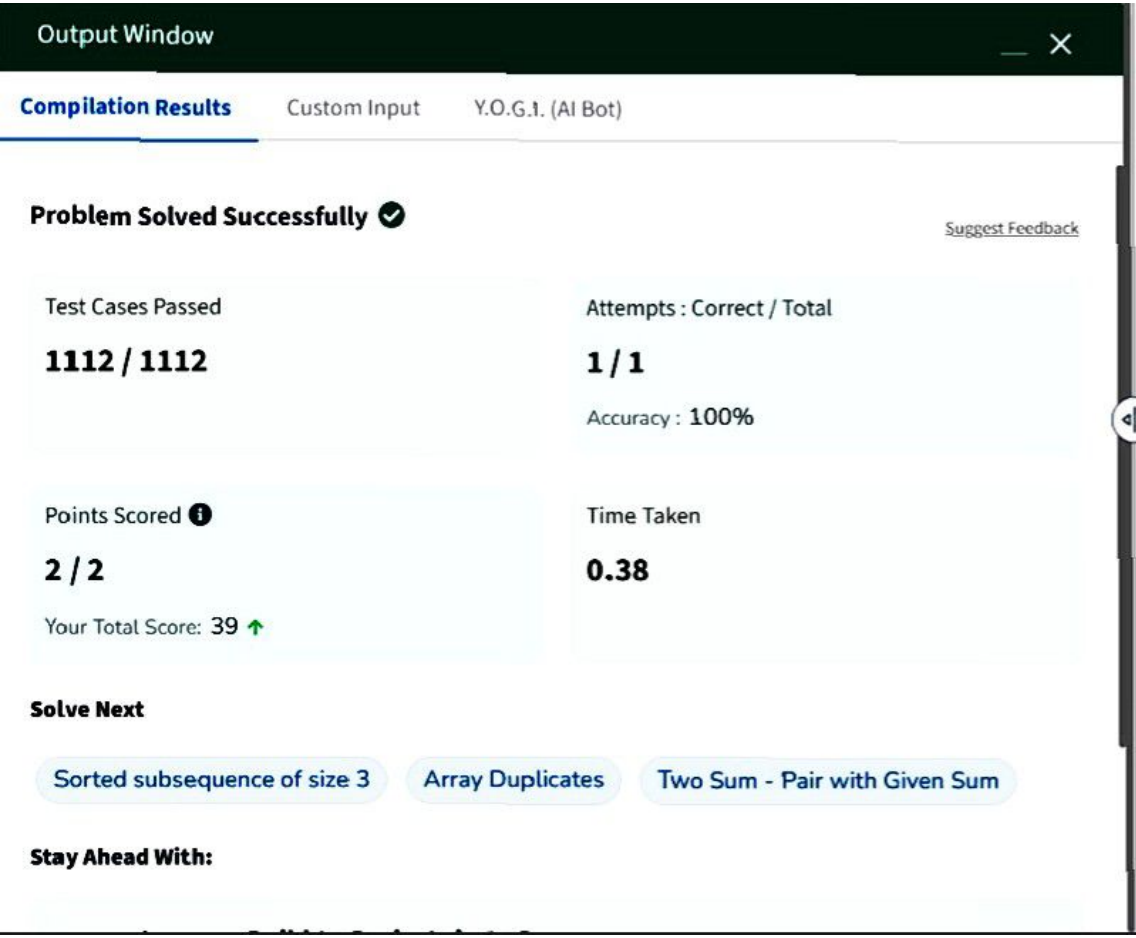
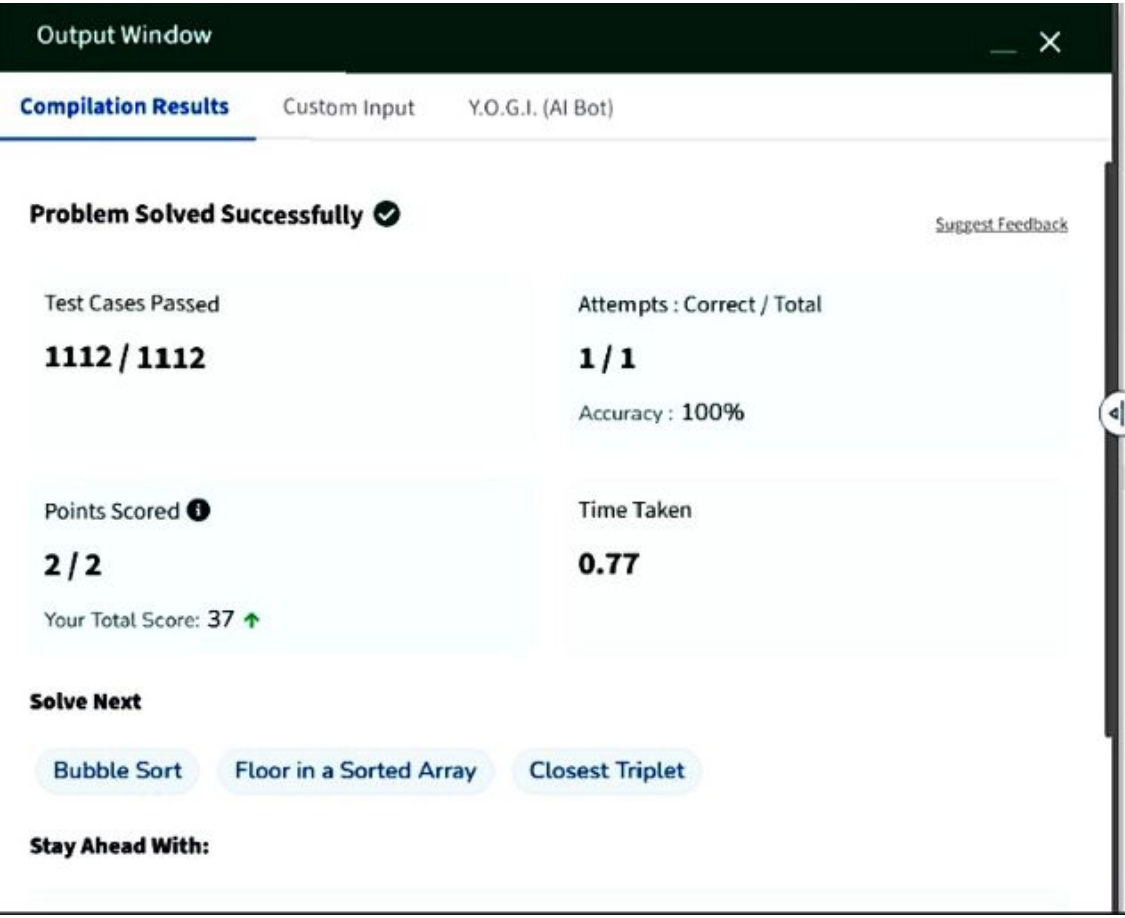



```java
// User function Template for Java

class Solution {

    public static int smallestSubWithSum(int x, int[] arr) {
        // Your code goes here
        int n = arr.length;
        int left = 0;
        int right = 0;
        int sum = 0;
        int minLen = Integer.MAX_VALUE;

        while(right < n) {
            sum += arr[right];

            while(sum > x) {
                int len = right-left+1;
                minLen = Math.min(minLen,len);

                sum -= arr[left];
                left++;
            }

            right++;
        }

        if(minLen == Integer.MAX_VALUE) return 0;

        return minLen;
    }
}
```




```java
import java.util.ArrayList;
import java.util.Collections;

class Solution {
    public int findMinDiff(ArrayList<Integer> arr, int m) {
        // Sort the array
        Collections.sort(arr);

        int start = 0;
        int end = m - 1;
        int ans = Integer.MAX_VALUE;

        while (end < arr.size()) {
            ans = Math.min(ans, arr.get(end) - arr.get(start));
            start++;
            end++;
        }

        return ans;
    }
}
```

Description | Editorial | Solutions | **Accepted** × | Submissions

← All Submissions

Accepted 172 / 172 testcases passed

SHREYA RAJ submitted at Feb 17, 2026 13:43

Editorial | Solution

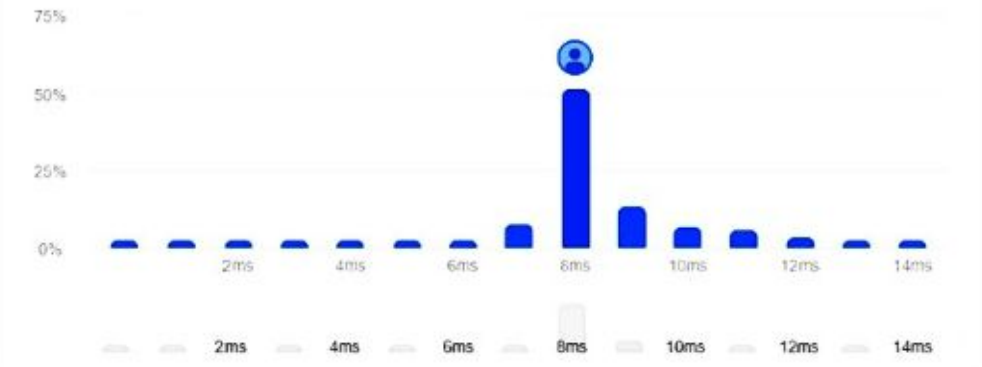
Runtime

8 ms | Beats **90.44%**

Analyze Complexity

Memory

49.22 MB | Beats 39.66%



Code Java

```java
class Solution {
    public int[][] merge(int[][] intervals) {
        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));

        List<int[]> merged = new ArrayList<>();
```

</> Code

Java | Auto

```java
        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));

        List<int[]> merged = new ArrayList<>();
        int[] prev = intervals[0];

        for (int i = 1; i < intervals.length; i++) {
            int[] interval = intervals[i];
            if (interval[0] <= prev[1]) {
                prev[1] = Math.max(prev[1], interval[1]);
```

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
ore

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)

25.3K 498 165 Online

Code

Java Auto

```java
class Solution {
    public int findDuplicate(int[] nums) {
        int slow = nums[0];
        int fast = nums[0];

        while (true) {
            slow = nums[slow];
            fast = nums[nums[fast]];

            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

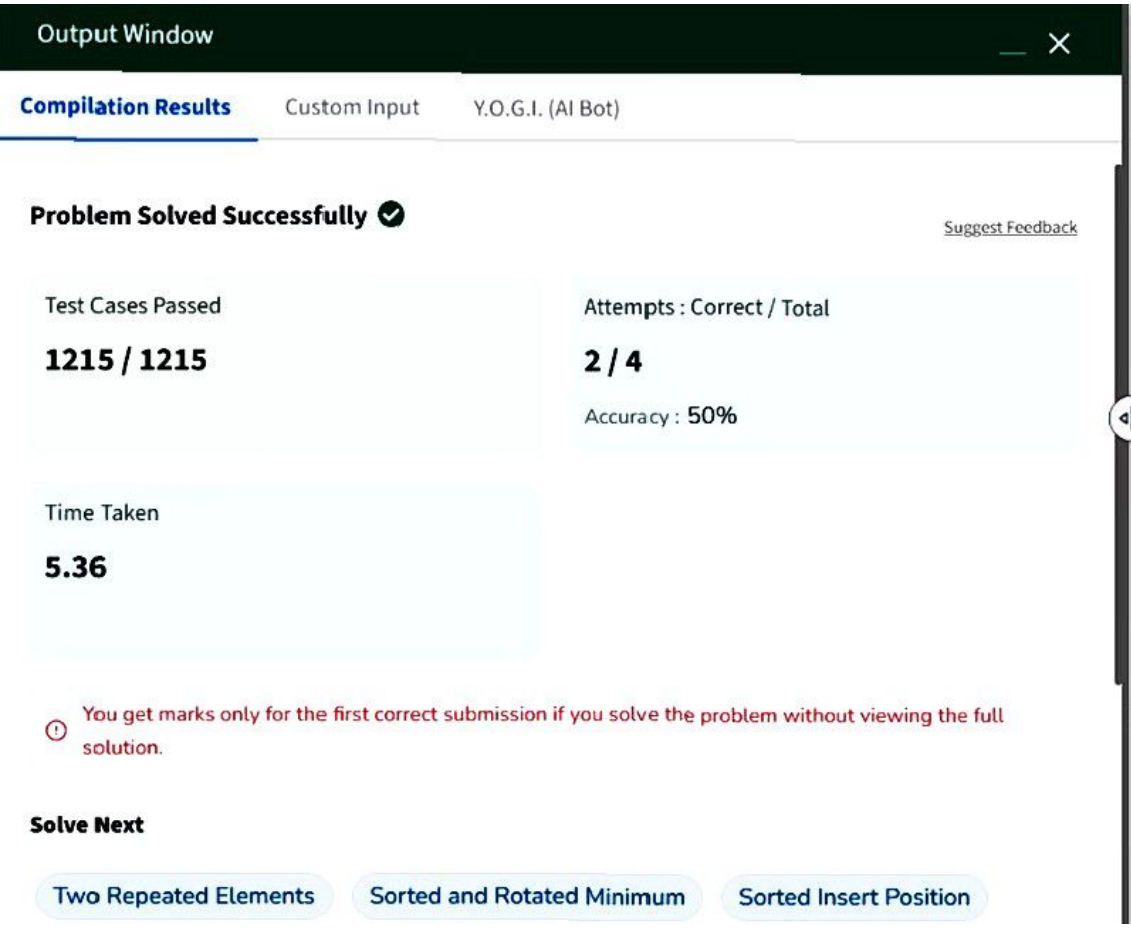
Median of 2 Sorted Arrays of Different Sizes | Nth Natural Number

Smallest Positive Integer that can not be represented as Sum

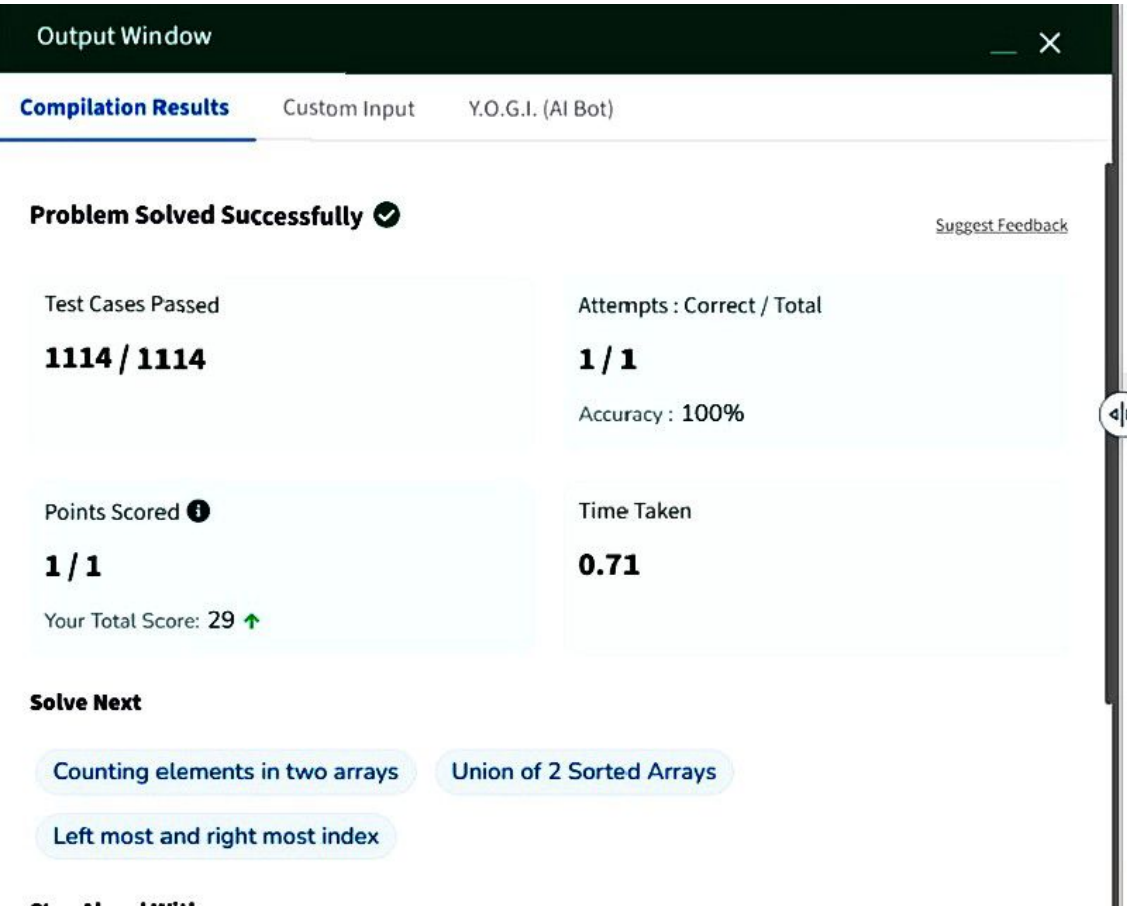
```java
class Solution {
    public void mergeArrays(int a[], int b[]) {
        int n = a.length;
        int m = b.length;

        PriorityQueue<Integer> pq = new PriorityQueue<>();

        for(int i = 0; i < m; i++){
            pq.add(b[i]);
        }

        for(int i = 0; i < n; i++){
            if(pq.peek() < a[i]){
                pq.add(a[i]);
                a[i] = pq.poll();
            }
        }

        for(int i = 0 ; i < m; i++){
            b[i] = pq.poll();
        }
    }
}
```

Ctrl + Enter

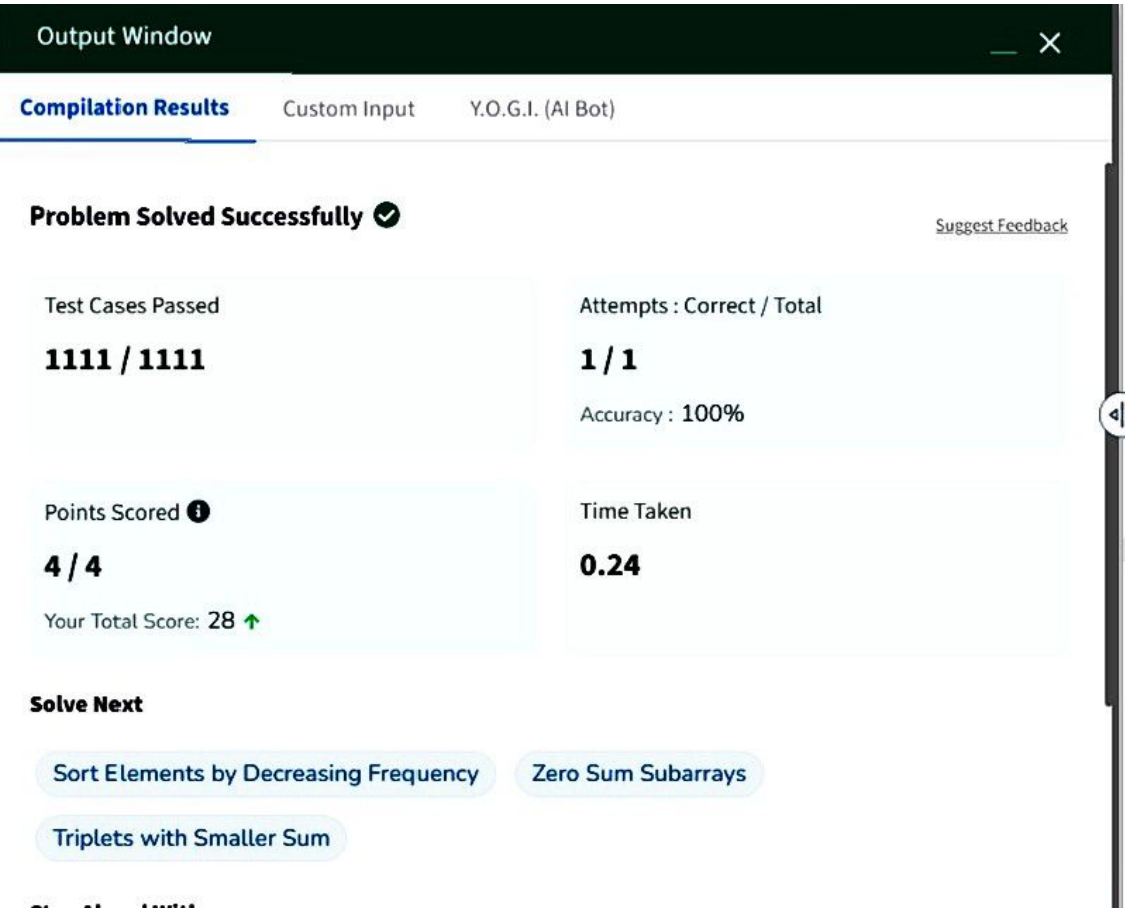
Custom Input | Compile & Run | Submit




```java
class Solution {
    // Function to find common elements in three arrays.
    public List<Integer> commonElements(List<Integer> arr1, List<Integer> arr2,
                                        List<Integer> arr3) {
        Map<Integer,Integer> mp = new TreeMap<>();

        HashSet<Integer> h1 = new HashSet<>(arr1);
        HashSet<Integer> h2 = new HashSet<>(arr2);
        HashSet<Integer> h3 = new HashSet<>(arr3);
         for(int i :h1){
             mp.put(i,mp.getOrDefault(i,0)+1);
         }
         for(int i :h2){
             mp.put(i,mp.getOrDefault(i,0)+1);
         }

         for(int i :h3){
             mp.put(i,mp.getOrDefault(i,0)+1);
         }

         List<Integer> res= new ArrayList<>();
         for(Map.Entry<Integer,Integer> entry: mp.entrySet()){
             if(entry.getValue()==3){
                 res.add(entry.getKey());
             }
         }
         return res;
    }
}

```




```java
class Solution {
    public boolean isSubset(int a[], int b[]) {
        // Your code here
        Arrays.sort(a);
        Arrays.sort(b);
        int ai = 0, bi = 0;
        while(ai < a.length && bi < b.length){
            if(a[ai] == b[bi])
                bi++;
            ai++;
        }
        if(bi == b.length)
            return true;
        return false;
    }
}
```




```java
class Solution {
    public boolean hasTripletSum(int arr[], int target) {

        Arrays.sort(arr);

        for(int i=0;i<arr.length-2;i++){
            int left=i+1;
            int right=arr.length-1;

            while(left<right){
                int sum=arr[i]+arr[left]+arr[right];
                if(sum==target){
                    return true;
                }else if(sum<target){
                    left++;
                }else{
                    right--;
                }
            }
        }
        return false;

    }
}
```

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

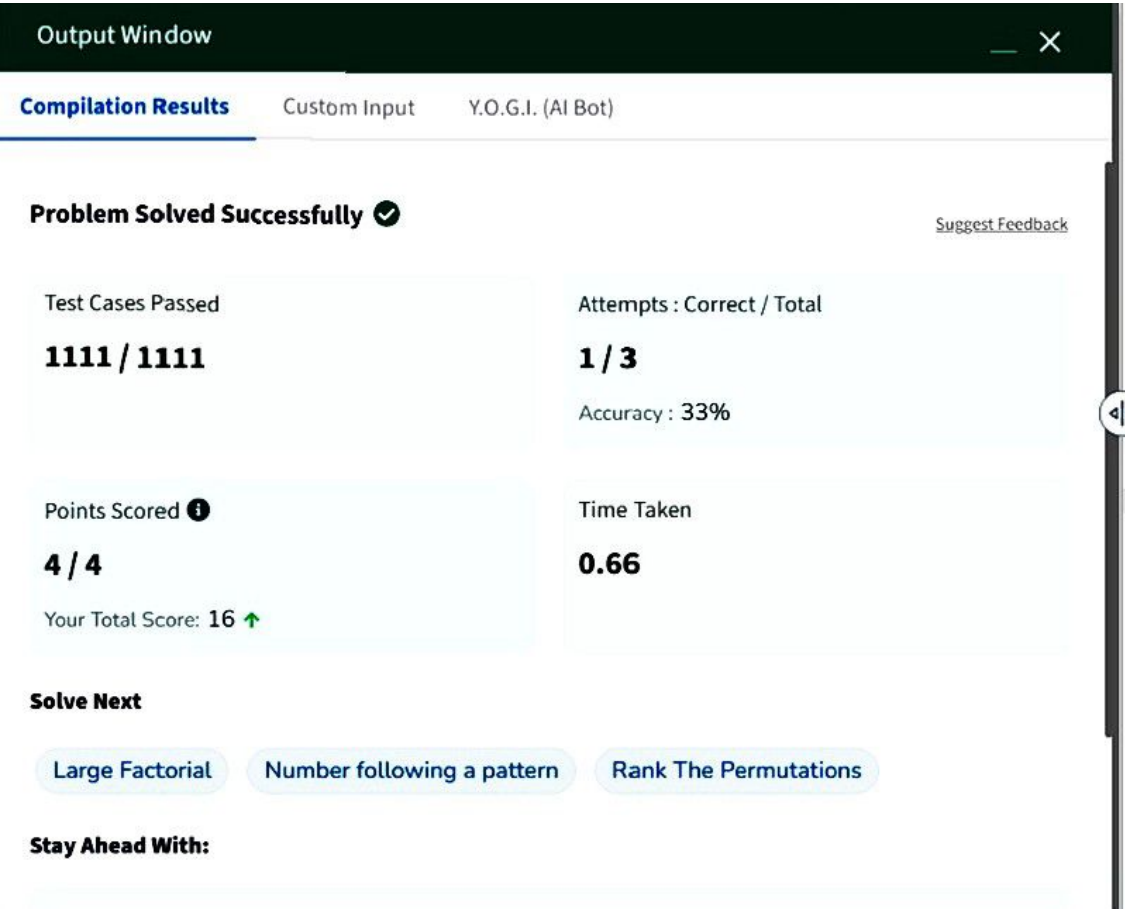
Longest Arithmetic Subsequence | Rod Cutting | Jump Game

Stay Ahead With:

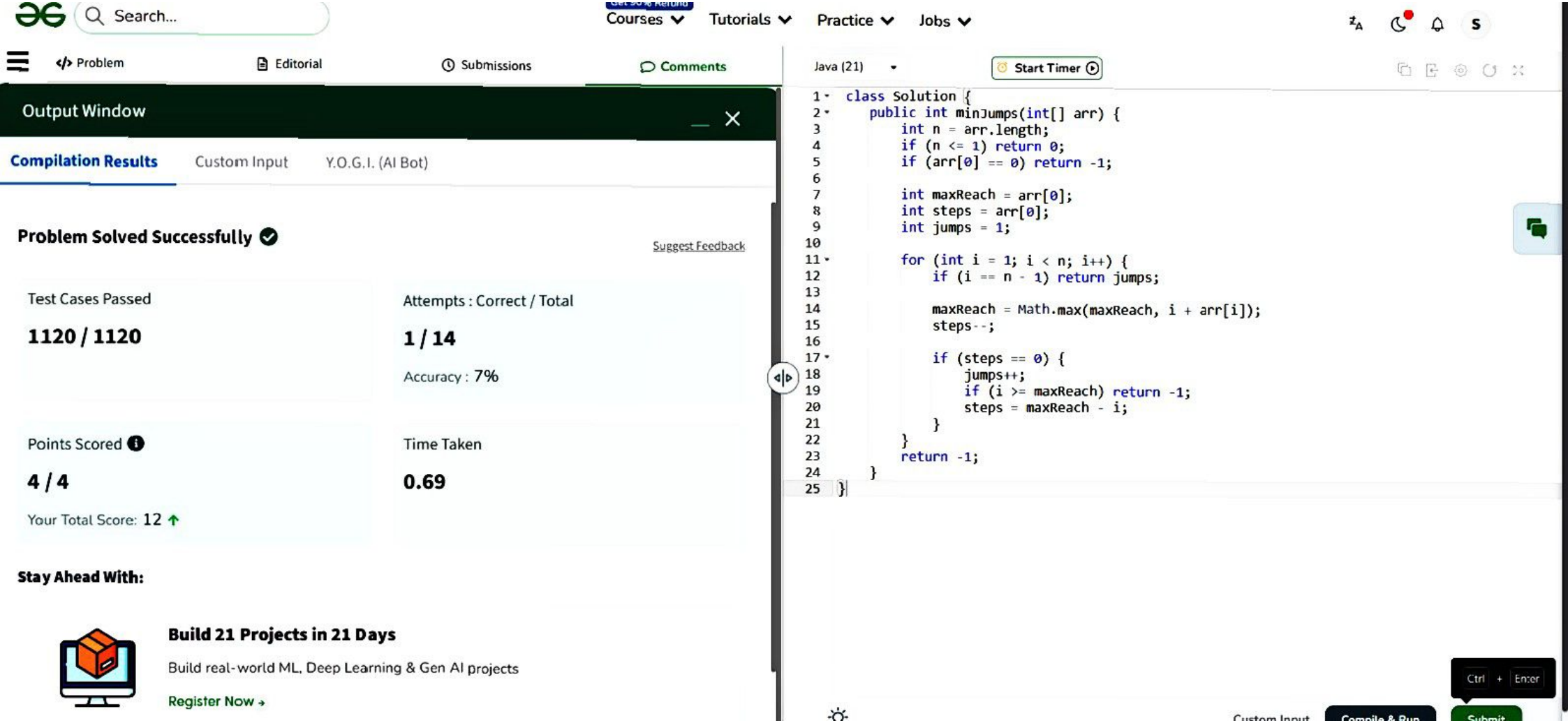
```java
class Solution {
    public int maxWater(int a[]) {

 int n=a.length;
 if(n==0)return 0;
 int res=0;
 int[]lm=new int[n];
 lm[0]=a[0];
 for(int i=1;i<n;i++){
     lm[i]=Math.max(lm[i-1],a[i]);
 }
 int rm[]=new int[n];
 rm[n-1]=a[n-1];
 for(int i=n-2;i>=0;i--){
     rm[i]=Math.max(rm[i+1],a[i]);
 }
 for(int i=0;i<n;i++){
     res+=Math.min(lm[i],rm[i])-a[i];
 }
 return res;
    }
}
```

Ctrl + Enter

Custom Input | Compile & Run | Submit




```java
import java.math.BigInteger;

class Solution {
    public static ArrayList<Integer> factorial(int n) {
        ArrayList<Integer> list = new ArrayList<>();
        BigInteger fac = fact(n);
        String s = fac.toString();

        for(char c : s.toCharArray()){
            list.add(c - '0');
        }
        return list;
    }

    public static BigInteger fact(int n){
        BigInteger res = BigInteger.ONE;
        for(int i=2; i<=n; i++){
            res = res.multiply(BigInteger.valueOf(i));
        }
        return res;
    }
}
```




```java
class Solution {
    public int minJumps(int[] arr) {
        int n = arr.length;
        if (n <= 1) return 0;
        if (arr[0] == 0) return -1;

        int maxReach = arr[0];
        int steps = arr[0];
        int jumps = 1;

        for (int i = 1; i < n; i++) {
            if (i == n - 1) return jumps;

            maxReach = Math.max(maxReach, i + arr[i]);
            steps--;

            if (steps == 0) {
                jumps++;
                if (i >= maxReach) return -1;
                steps = maxReach - i;
            }
        }
        return -1;
    }
}
```