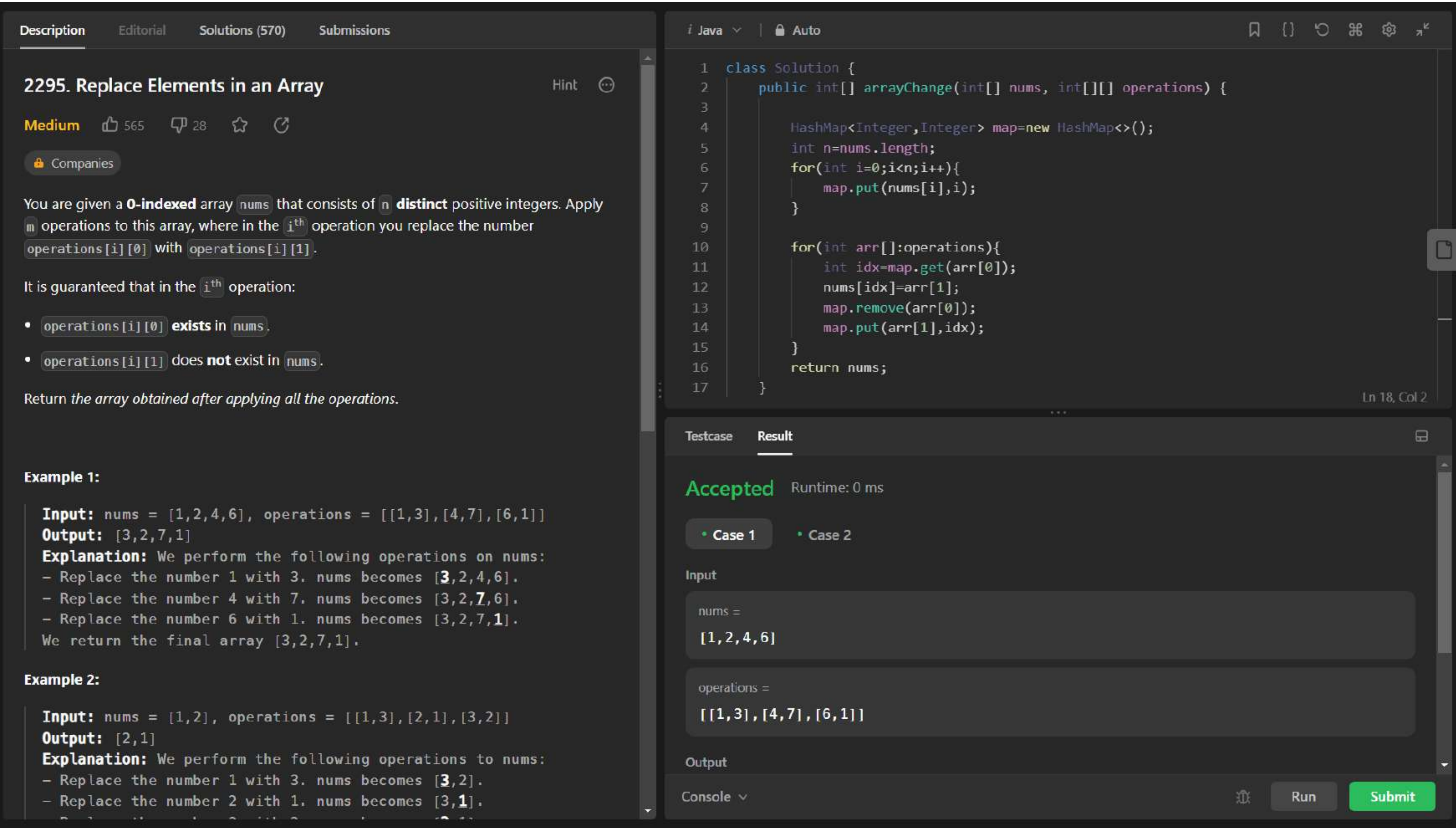


## 2295. Replace Elements in an Array

You are given a **0-indexed** array `nums` that consists of `n` **distinct** positive integers. Apply `m` operations to this array, where in the `i`th operation you replace the number `operations[i][0]` with `operations[i][1]`.

It is guaranteed that in the `i`th operation:

- `operations[i][0]` **exists** in `nums`.
- `operations[i][1]` does **not** exist in `nums`.

Return *the array obtained after applying all the operations.*

**Example 1:**

```
Input: nums = [1,2,4,6], operations = [[1,3],[4,7],[6,1]]
Output: [3,2,7,1]
Explanation: We perform the following operations on nums:
- Replace the number 1 with 3. nums becomes [3,2,4,6].
- Replace the number 4 with 7. nums becomes [3,2,7,6].
- Replace the number 6 with 1. nums becomes [3,2,7,1].
We return the final array [3,2,7,1].
```

**Example 2:**

```
Input: nums = [1,2], operations = [[1,3],[2,1],[3,2]]
Output: [2,1]
Explanation: We perform the following operations to nums:
- Replace the number 1 with 3. nums becomes [3,2].
- Replace the number 2 with 1. nums becomes [3,1].
```

```java
class Solution {
    public int[] arrayChange(int[] nums, int[][] operations) {

        HashMap<Integer,Integer> map=new HashMap<>();
        int n=nums.length;
        for(int i=0;i<n;i++){
            map.put(nums[i],i);
        }

        for(int arr[]:operations){
            int idx=map.get(arr[0]);
            nums[idx]=arr[1];
            map.remove(arr[0]);
            map.put(arr[1],idx);
        }
        return nums;
    }
```

Ln 18, Col 2

Accepted Runtime: 0 ms

Case 1 Case 2

Input

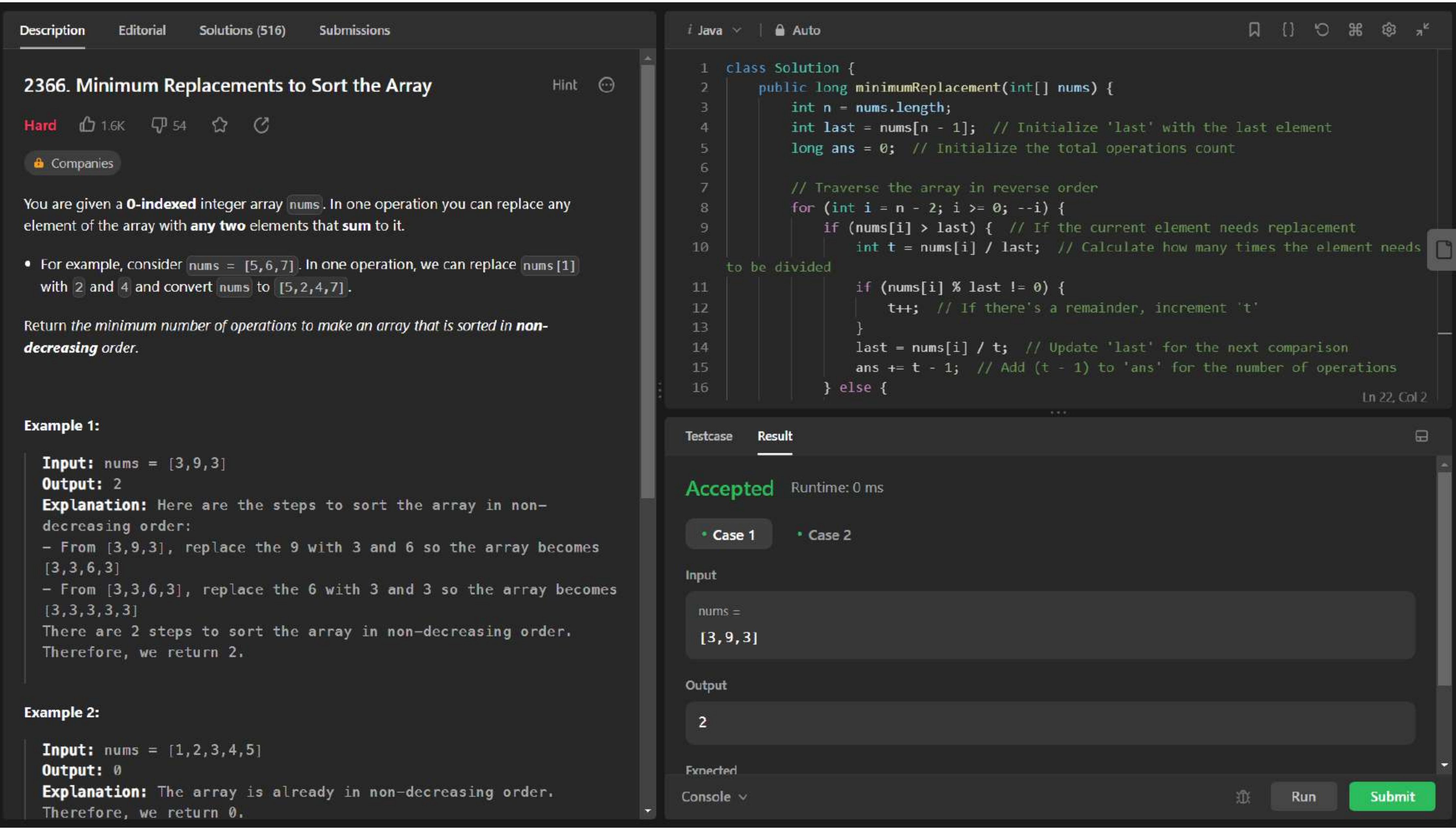
nums =

```
[1,2,4,6]
```

operations =

```
[[1,3],[4,7],[6,1]]
```

Output



Description Editorial Solutions (516) Submissions

## 2366. Minimum Replacements to Sort the Array

Hint

Hard 1.6K 54

Companies

You are given a **0-indexed** integer array `nums`. In one operation you can replace any element of the array with **any two** elements that **sum** to it.

- For example, consider `nums = [5,6,7]`. In one operation, we can replace `nums[1]` with `2` and `4` and convert `nums` to `[5,2,4,7]`.

Return *the minimum number of operations to make an array that is sorted in* ***non-decreasing*** *order*.

**Example 1:**

```
Input: nums = [3,9,3]
Output: 2
Explanation: Here are the steps to sort the array in non-
decreasing order:
- From [3,9,3], replace the 9 with 3 and 6 so the array becomes
[3,3,6,3]
- From [3,3,6,3], replace the 6 with 3 and 3 so the array becomes
[3,3,3,3,3]
There are 2 steps to sort the array in non-decreasing order.
Therefore, we return 2.
```

**Example 2:**

```
Input: nums = [1,2,3,4,5]
Output: 0
Explanation: The array is already in non-decreasing order.
Therefore, we return 0.
```

Java Auto

```java
class Solution {
    public long minimumReplacement(int[] nums) {
        int n = nums.length;
        int last = nums[n - 1];  // Initialize 'last' with the last element
        long ans = 0;  // Initialize the total operations count

        // Traverse the array in reverse order
        for (int i = n - 2; i >= 0; --i) {
            if (nums[i] > last) {  // If the current element needs replacement
                int t = nums[i] / last;  // Calculate how many times the element needs
to be divided
                if (nums[i] % last != 0) {
                    t++;  // If there's a remainder, increment 't'
                }
                last = nums[i] / t;  // Update 'last' for the next comparison
                ans += t - 1;  // Add (t - 1) to 'ans' for the number of operations
            } else {
```

Ln 22, Col 2

Testcase Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

nums =

```
[3,9,3]
```

Output

```
2
```

Expected

Console Run Submit