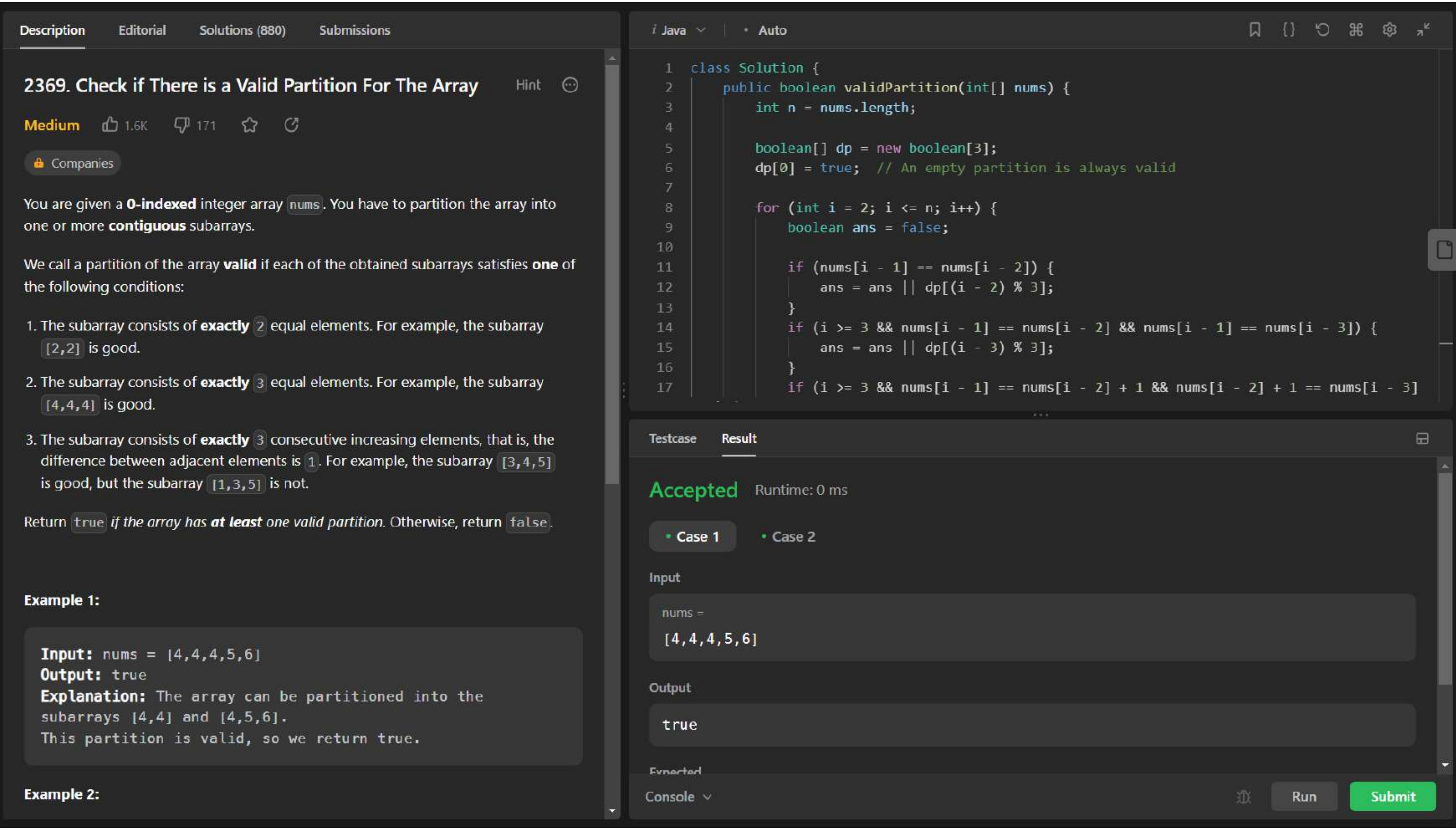


Description Editorial Solutions (880) Submissions

2369. Check if There is a Valid Partition For The Array

Hint

Medium 1.6K 171

Companies

You are given a **0-indexed** integer array `nums`. You have to partition the array into one or more **contiguous** subarrays.

We call a partition of the array **valid** if each of the obtained subarrays satisfies **one** of the following conditions:

1. The subarray consists of **exactly** `2` equal elements. For example, the subarray `[2,2]` is good.
2. The subarray consists of **exactly** `3` equal elements. For example, the subarray `[4,4,4]` is good.
3. The subarray consists of **exactly** `3` consecutive increasing elements, that is, the difference between adjacent elements is `1`. For example, the subarray `[3,4,5]` is good, but the subarray `[1,3,5]` is not.

Return `true` *if the array has **at least** one valid partition*. Otherwise, return `false`.

Example 1:

```
Input: nums = [4,4,4,5,6]
Output: true
Explanation: The array can be partitioned into the
subarrays [4,4] and [4,5,6].
This partition is valid, so we return true.
```

Example 2:

Java Auto

```java
class Solution {
    public boolean validPartition(int[] nums) {
        int n = nums.length;

        boolean[] dp = new boolean[3];
        dp[0] = true;  // An empty partition is always valid

        for (int i = 2; i <= n; i++) {
            boolean ans = false;

            if (nums[i - 1] == nums[i - 2]) {
                ans = ans || dp[(i - 2) % 3];
            }
            if (i >= 3 && nums[i - 1] == nums[i - 2] && nums[i - 1] == nums[i - 3]) {
                ans = ans || dp[(i - 3) % 3];
            }
            if (i >= 3 && nums[i - 1] == nums[i - 2] + 1 && nums[i - 2] + 1 == nums[i - 3]
```

Testcase Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

nums =

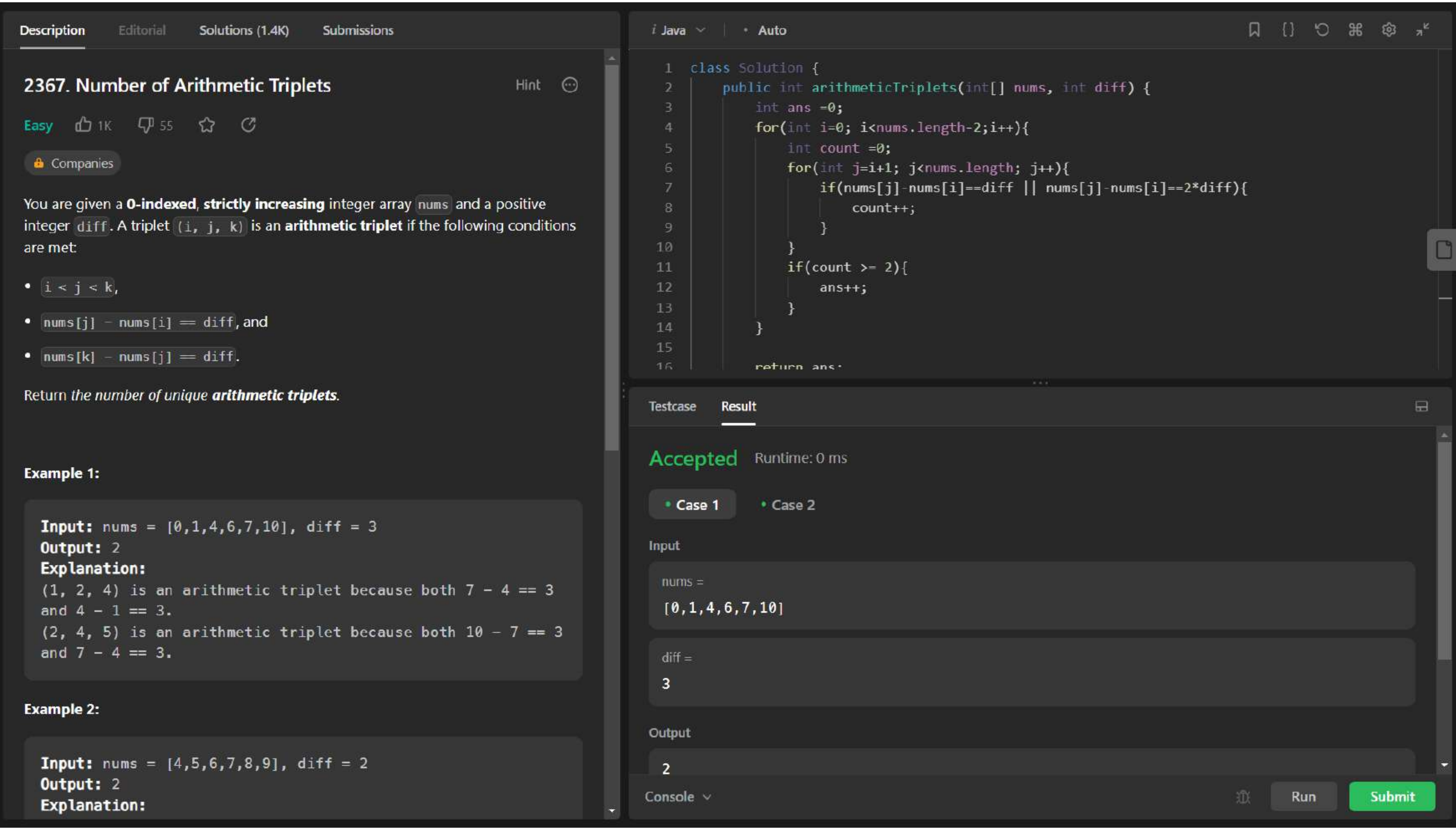
```
[4,4,4,5,6]
```

Output

```
true
```

Expected

Console Run Submit



2367. Number of Arithmetic Triplets

You are given a **0-indexed**, **strictly increasing** integer array `nums` and a positive integer `diff`. A triplet `(i, j, k)` is an **arithmetic triplet** if the following conditions are met:

- `i < j < k`,
- `nums[j] - nums[i] == diff`, and
- `nums[k] - nums[j] == diff`.

Return *the number of unique* ***arithmetic triplets***.

Example 1:

```
Input: nums = [0,1,4,6,7,10], diff = 3
Output: 2
Explanation:
(1, 2, 4) is an arithmetic triplet because both 7 - 4 == 3
and 4 - 1 == 3.
(2, 4, 5) is an arithmetic triplet because both 10 - 7 == 3
and 7 - 4 == 3.
```

Example 2:

```
Input: nums = [4,5,6,7,8,9], diff = 2
Output: 2
Explanation:
```

```java
class Solution {
    public int arithmeticTriplets(int[] nums, int diff) {
        int ans =0;
        for(int i=0; i<nums.length-2;i++){
            int count =0;
            for(int j=i+1; j<nums.length; j++){
                if(nums[j]-nums[i]==diff || nums[j]-nums[i]==2*diff){
                    count++;
                }
            }
            if(count >= 2){
                ans++;
            }
        }

        return ans;
```

Accepted Runtime: 0 ms

Input

nums =
```
[0,1,4,6,7,10]
```

diff =
```
3
```

Output
```
2
```