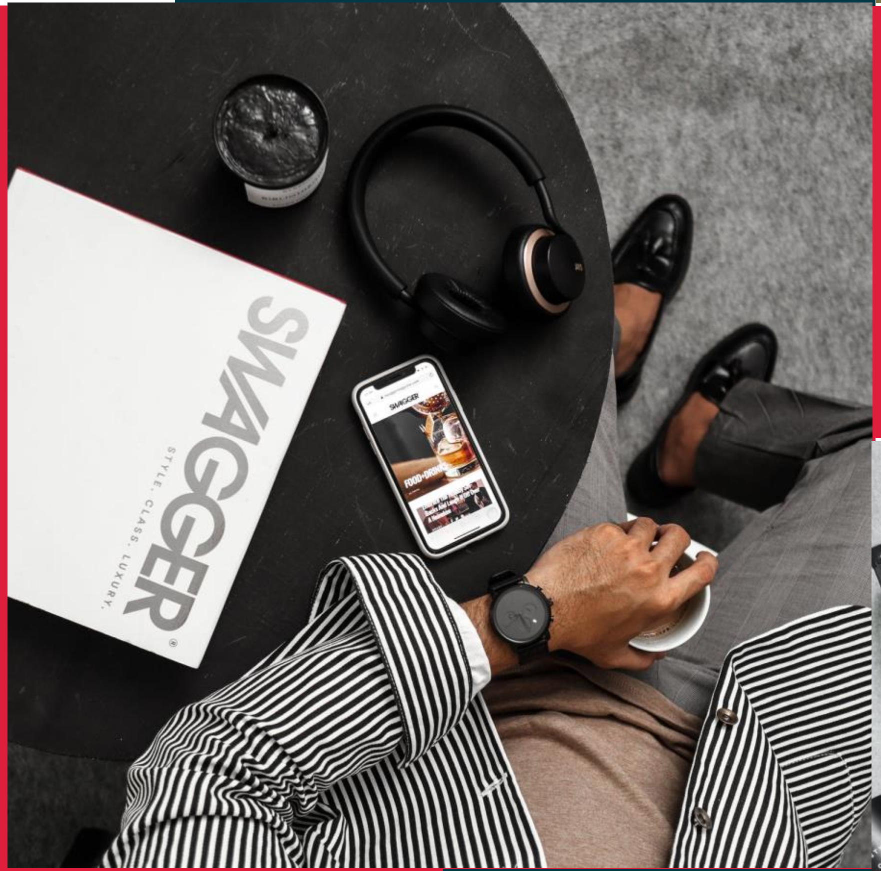
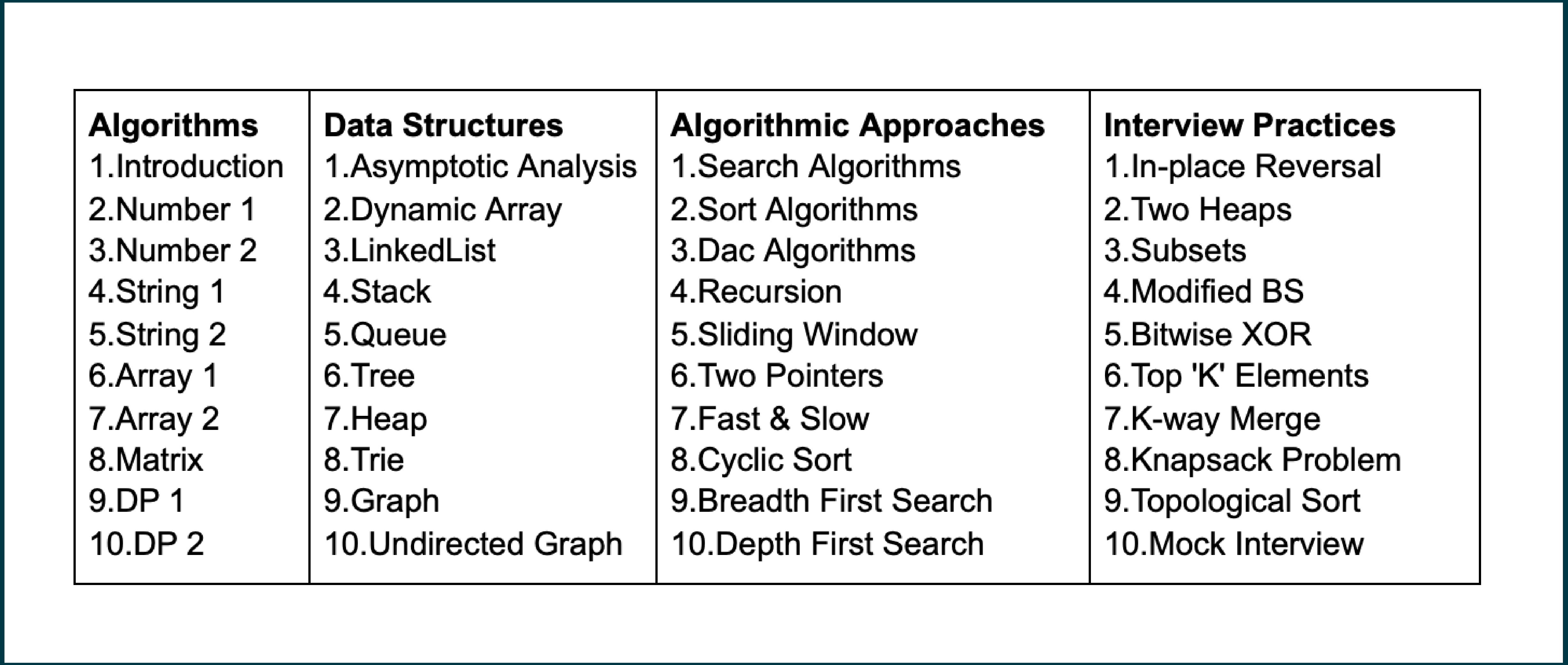
DSA - Algorithms

Array 2



Course Planning

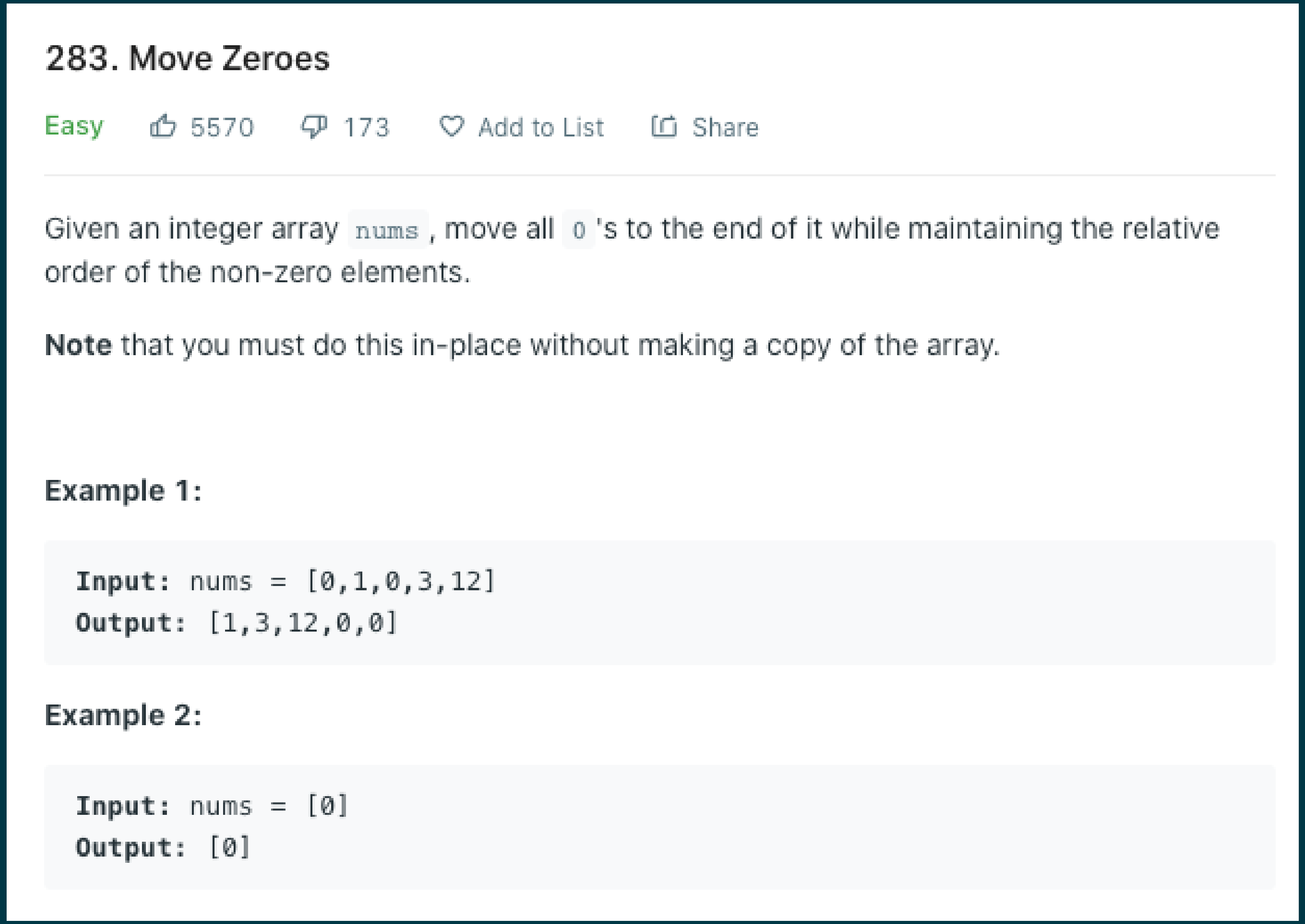


Algorithms	Data Structures	Algorithmic Approaches	Interview Practices
1.Introduction	1.Asymptotic Analysis	1.Search Algorithms	1.In-place Reversal
2.Number 1	2.Dynamic Array	2.Sort Algorithms	2.Two Heaps
3.Number 2	3.LinkedList	3.Dac Algorithms	3.Subsets
4.String 1	4.Stack	4.Recursion	4.Modified BS
5.String 2	5.Queue	5.Sliding Window	5.Bitwise XOR
6.Array 1	6.Tree	6.Two Pointers	6.Top 'K' Elements
7.Array 2	7.Heap	7.Fast & Slow	7.K-way Merge
8.Matrix	8.Trie	8.Cyclic Sort	8.Knapsack Problem
9.DP 1	9.Graph	9.Breadth First Search	9.Topological Sort
10.DP 2	10.Undirected Graph	10.Depth First Search	10.Mock Interview

https://kurbanovxurshid.medium.com/data-structures-and-algorithms-course-plan-1ab912ec1e17

Asked by Facebook

Explanation



283. Move Zeroes

Easy 👍 5570 👎 173 ♡ Add to List ⎘ Share

Given an integer array `nums`, move all `0`'s to the end of it while maintaining the relative order of the non-zero elements.

Note that you must do this in-place without making a copy of the array.

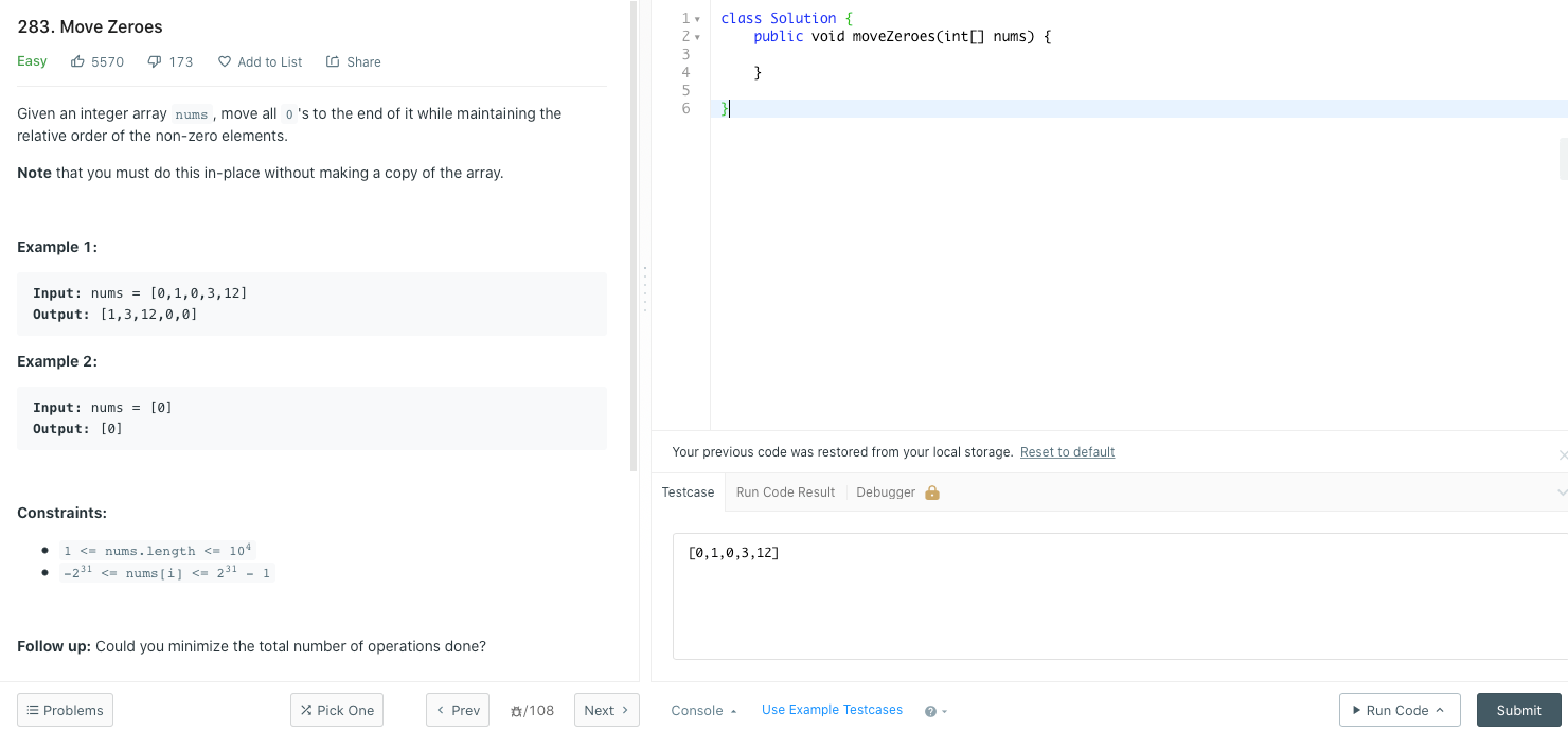
Example 1:

```
Input: nums = [0,1,0,3,12]
Output: [1,3,12,0,0]
```

Example 2:

```
Input: nums = [0]
Output: [0]
```

Move Zeroes



283. Move Zeroes

Easy 5570 173 Add to List Share

Given an integer array `nums`, move all `0`'s to the end of it while maintaining the relative order of the non-zero elements.

Note that you must do this in-place without making a copy of the array.

Example 1:

```
Input: nums = [0,1,0,3,12]
Output: [1,3,12,0,0]
```

Example 2:

```
Input: nums = [0]
Output: [0]
```

Constraints:

- `1 <= nums.length <=` 10^4
- -2^{31} `<= nums[i] <=` $2^{31} - 1$

Follow up: Could you minimize the total number of operations done?

```
class Solution {
    public void moveZeroes(int[] nums) {
        
    }
    
}
```

Your previous code was restored from your local storage. Reset to default

```
[0,1,0,3,12]
```

https://leetcode.com/problems/move-zeroes/

First Theory

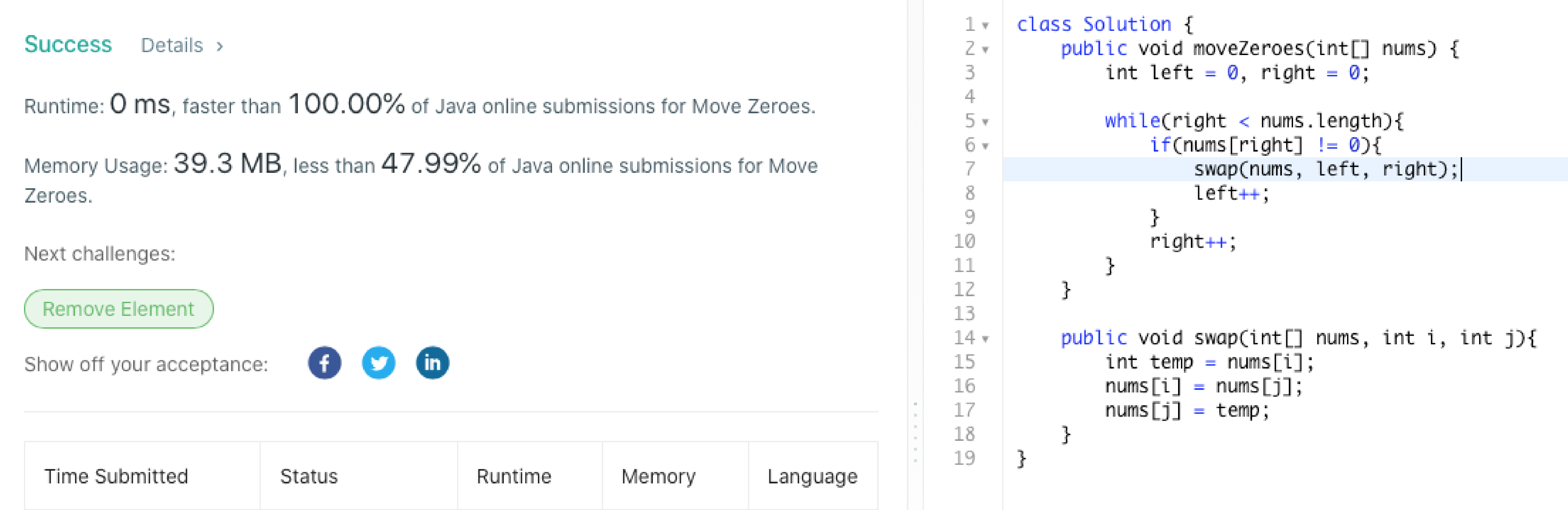
[0,1,0,3,12]

[1,0,0,3,12]

[1,3,0,0,12]

[1,3,12,0,0]

First Solution



Success Details ›

Runtime: 0 ms, faster than 100.00% of Java online submissions for Move Zeroes.

Memory Usage: 39.3 MB, less than 47.99% of Java online submissions for Move Zeroes.

Next challenges:

Remove Element

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language

```java
class Solution {
    public void moveZeroes(int[] nums) {
        int left = 0, right = 0;

        while(right < nums.length){
            if(nums[right] != 0){
                swap(nums, left, right);
                left++;
            }
            right++;
        }
    }

    public void swap(int[] nums, int i, int j){
        int temp = nums[i];
        nums[i] = nums[j];
        nums[j] = temp;
    }
}
```

[0,1,0,3,12]

[1,0,0,3,12]

[1,3,0,0,12]

[1,3,12,0,0]

Second Theory

[0,1,0,3,12]

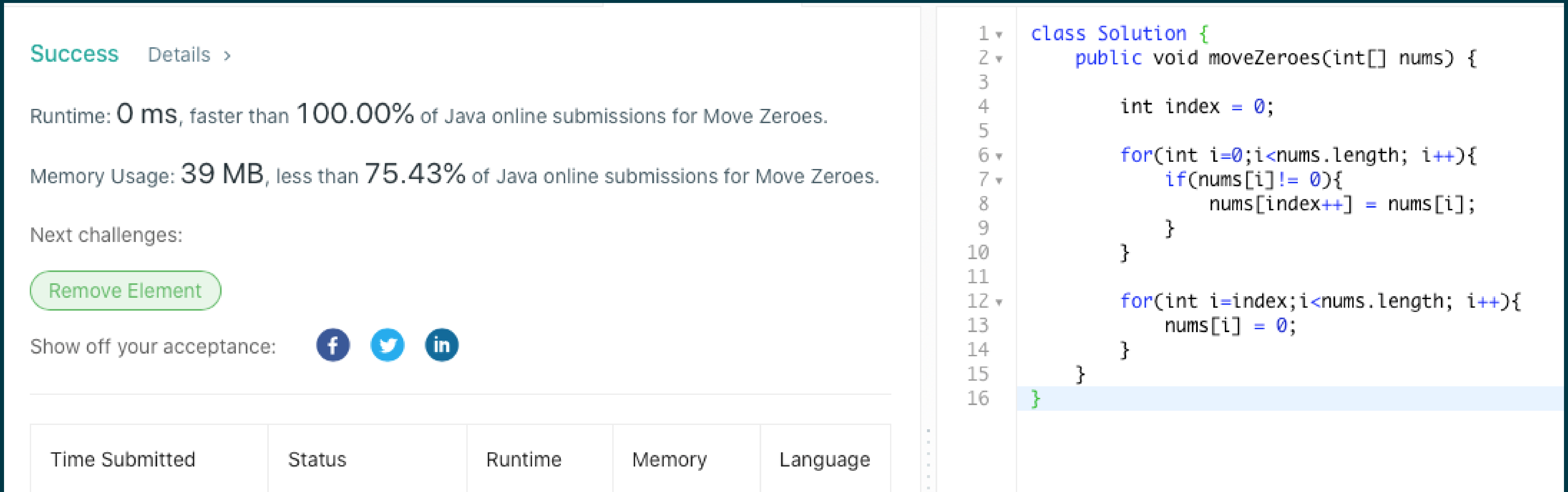
[1,1,0,3,12]

[1,3,0,3,12]

[1,3,12,3,12]

[1,3,12,0,0]

Second Solution



Success Details ›

Runtime: 0 ms, faster than 100.00% of Java online submissions for Move Zeroes.

Memory Usage: 39 MB, less than 75.43% of Java online submissions for Move Zeroes.

Next challenges:

Time Submitted	Status	Runtime	Memory	Language

```java
class Solution {
    public void moveZeroes(int[] nums) {

        int index = 0;

        for(int i=0;i<nums.length; i++){
            if(nums[i]!= 0){
                nums[index++] = nums[i];
            }
        }

        for(int i=index;i<nums.length; i++){
            nums[i] = 0;
        }
    }
}
```

[0,1,0,3,12]

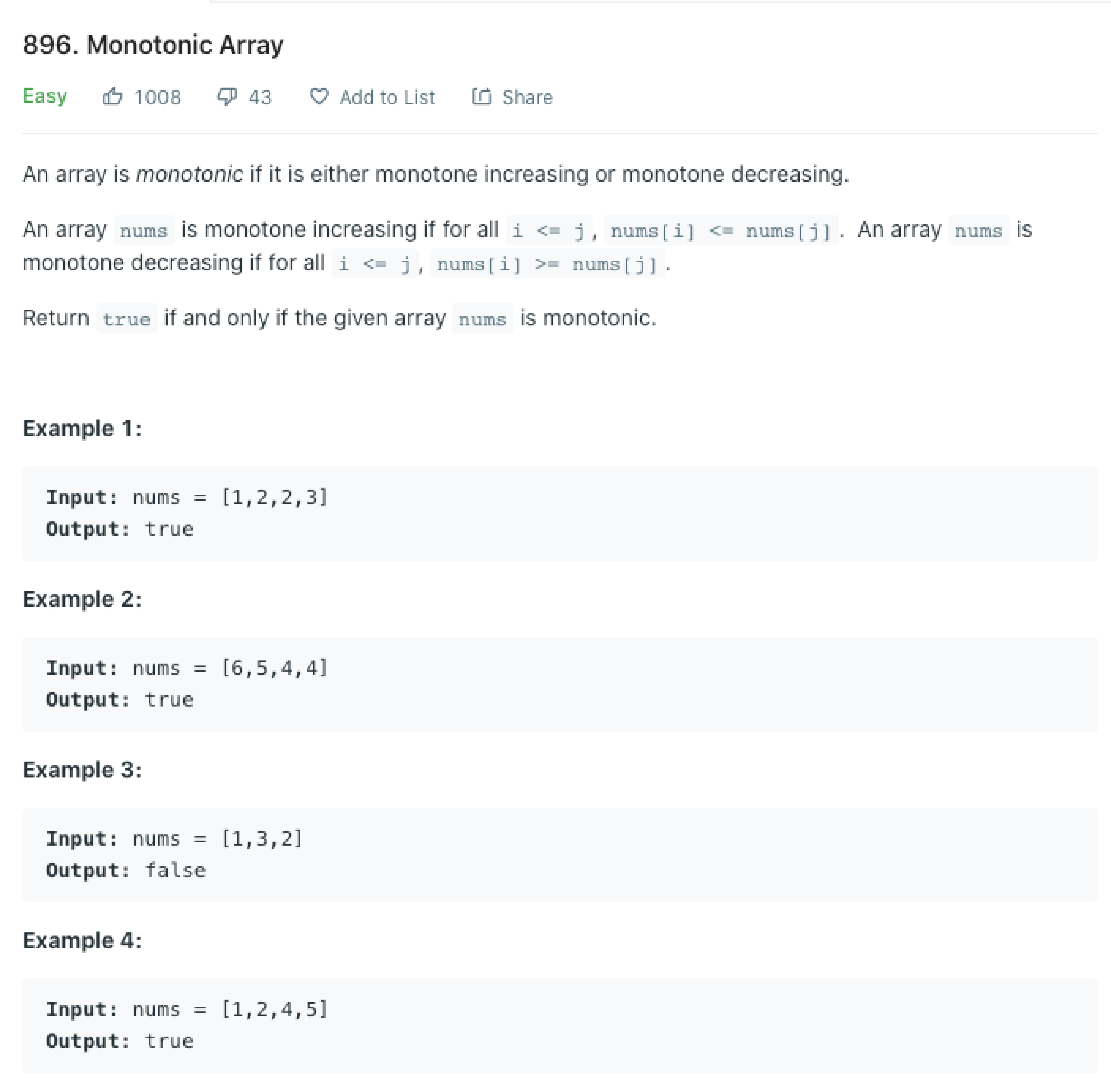
[1,1,0,3,12]

[1,3,0,3,12]

[1,3,12,3,12]

[1,3,12,0,0]

Task 1 – Monotonic Array



896. Monotonic Array

An array is *monotonic* if it is either monotone increasing or monotone decreasing.

An array `nums` is monotone increasing if for all `i <= j`, `nums[i] <= nums[j]`. An array `nums` is monotone decreasing if for all `i <= j`, `nums[i] >= nums[j]`.

Return `true` if and only if the given array `nums` is monotonic.

Example 1:

```
Input: nums = [1,2,2,3]
Output: true
```

Example 2:

```
Input: nums = [6,5,4,4]
Output: true
```

Example 3:

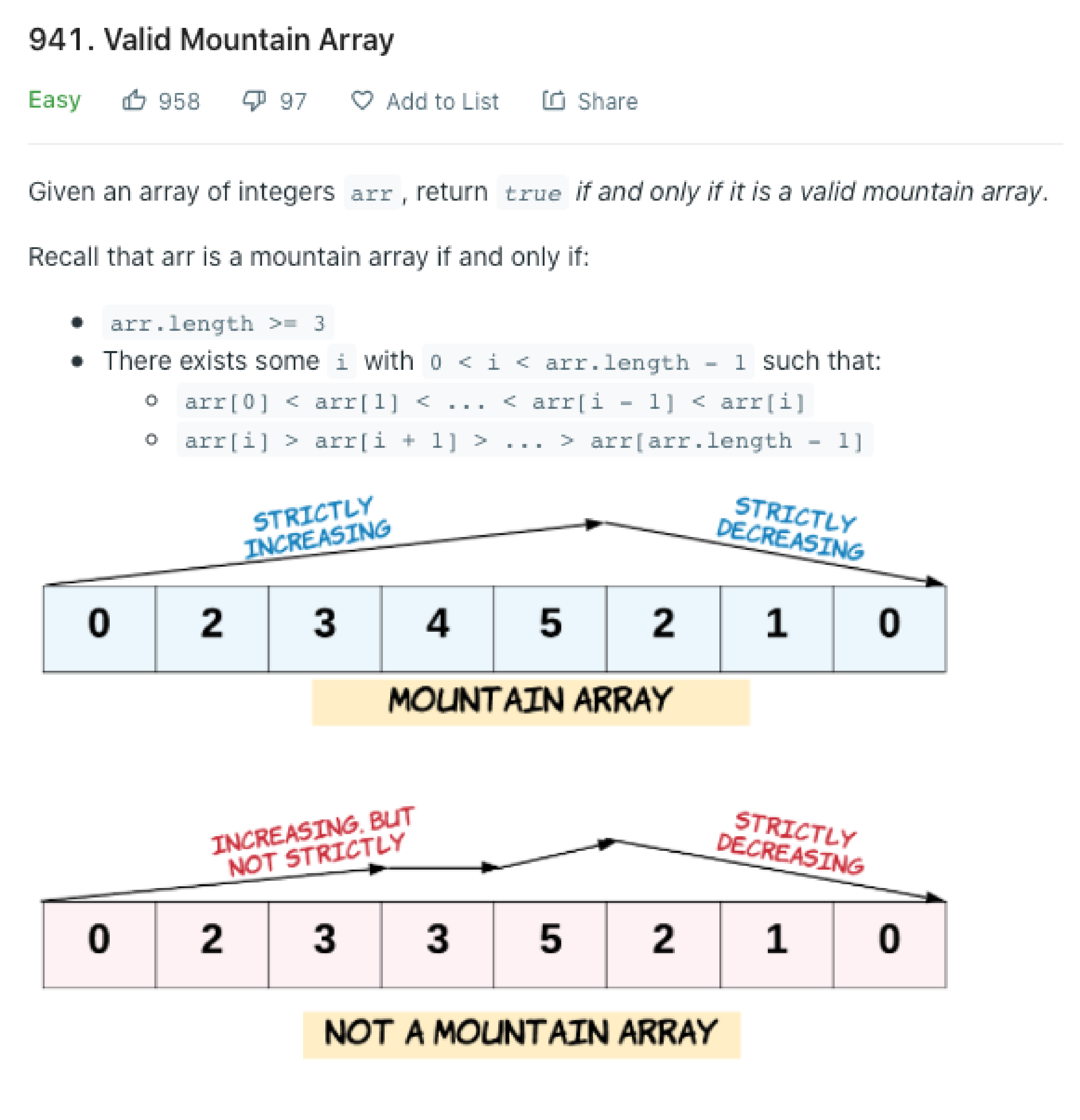
```
Input: nums = [1,3,2]
Output: false
```

Example 4:

```
Input: nums = [1,2,4,5]
Output: true
```

https://leetcode.com/problems/monotonic-array/

Task 2 – Valid Mountain Array



941. Valid Mountain Array

Easy 958 97 Add to List Share

Given an array of integers `arr`, return `true` *if and only if it is a valid mountain array*.

Recall that arr is a mountain array if and only if:

- `arr.length >= 3`
- There exists some `i` with `0 < i < arr.length - 1` such that:
  - `arr[0] < arr[1] < ... < arr[i - 1] < arr[i]`
  - `arr[i] > arr[i + 1] > ... > arr[arr.length - 1]`

0	2	3	4	5	2	1	0

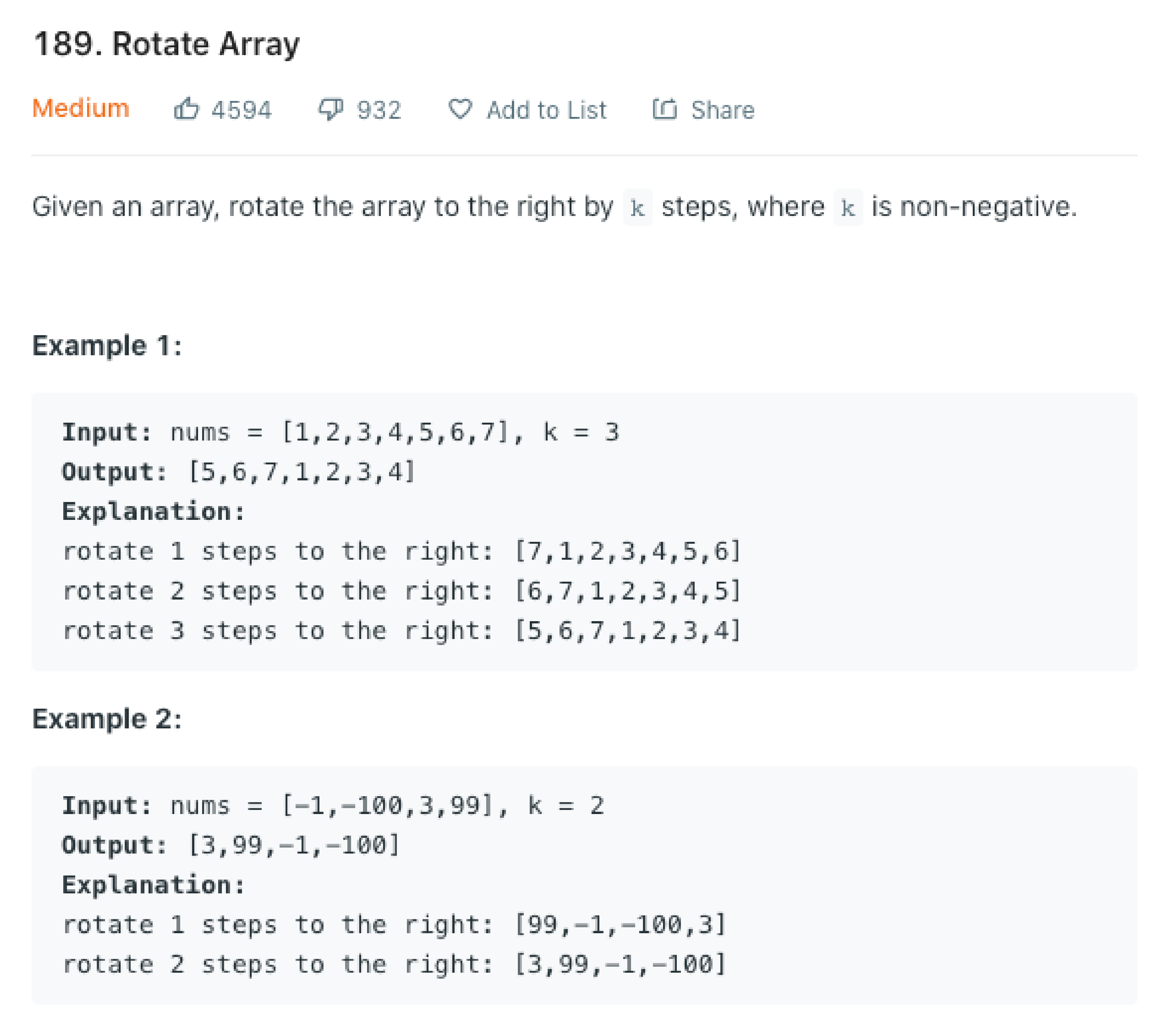
MOUNTAIN ARRAY

0	2	3	3	5	2	1	0

NOT A MOUNTAIN ARRAY

https://leetcode.com/problems/valid-mountain-array/

Task 3 – Rotate Array



189. Rotate Array

Given an array, rotate the array to the right by `k` steps, where `k` is non-negative.

Example 1:

```
Input: nums = [1,2,3,4,5,6,7], k = 3
Output: [5,6,7,1,2,3,4]
Explanation:
rotate 1 steps to the right: [7,1,2,3,4,5,6]
rotate 2 steps to the right: [6,7,1,2,3,4,5]
rotate 3 steps to the right: [5,6,7,1,2,3,4]
```

Example 2:

```
Input: nums = [-1,-100,3,99], k = 2
Output: [3,99,-1,-100]
Explanation:
rotate 1 steps to the right: [99,-1,-100,3]
rotate 2 steps to the right: [3,99,-1,-100]
```

https://leetcode.com/problems/rotate-array/