Unknown Title

Description

Description

Note

Note

Editorial

Editorial

Solutions

Solutions

Submissions

Submissions

Code

Code

☑

Testcase

Testcase

>_

Test Result

Test Result

26. Remove Duplicates from Sorted Array

Easy

Topics
Companies
Hint

Given an integer array `nums` sorted in **non-decreasing order**, remove the duplicates **in-place** such that each unique element appears only **once**. The **relative order** of the elements should be kept the **same**. Then return *the number of unique elements in* `nums`.

Consider the number of unique elements of `nums` to be `k`, to get accepted, you need to do the following things:

- Change the array `nums` such that the first `k` elements of `nums` contain the unique elements in the order they were present in `nums` initially. The remaining elements of `nums` are not important as well as the size of `nums`.
- Return `k`.

Custom Judge:

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int[] expectedNums = [...]; // The expected answer with correct length

int k = removeDuplicates(nums); // Calls your implementation

assert k == expectedNums.length;
```

```
for (int i = 0; i < k; i++) {
    assert nums[i] == expectedNums[i];
}
```

If all assertions pass, then your solution will be **accepted**.

Example 1:

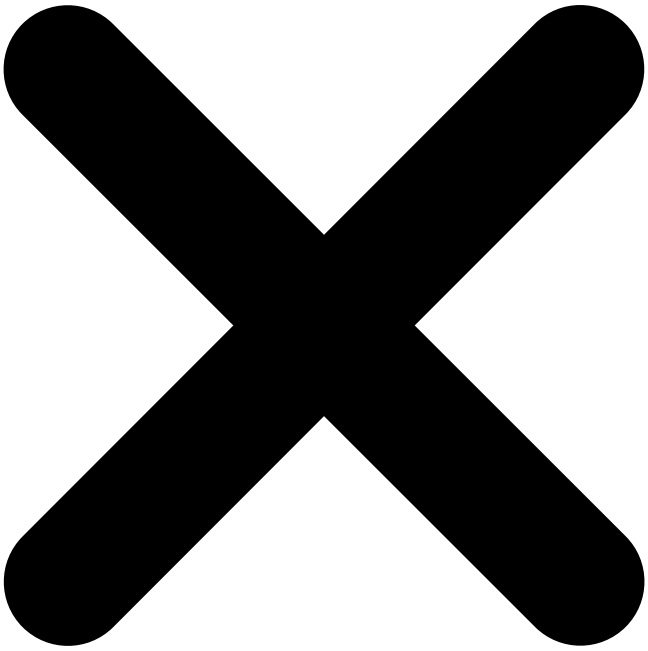
Input: nums = [1,1,2]
Output: 2, nums = [1,2,_]
Explanation: Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).

Example 2:

Input: nums = [0,0,1,1,1,2,2,3,3,4]
Output: 5, nums = [0,1,2,3,4,_,_,_,_,_]
Explanation: Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).

Constraints:

- $1 <= nums.length <= 3 * 10^4$
- $-100 <= nums[i] <= 100$
- nums is sorted in **non-decreasing** order.



Seen this question in a real interview before?

1/5

Yes

No

Accepted

5M

Submissions

8.7M

Acceptance Rate

57.7%

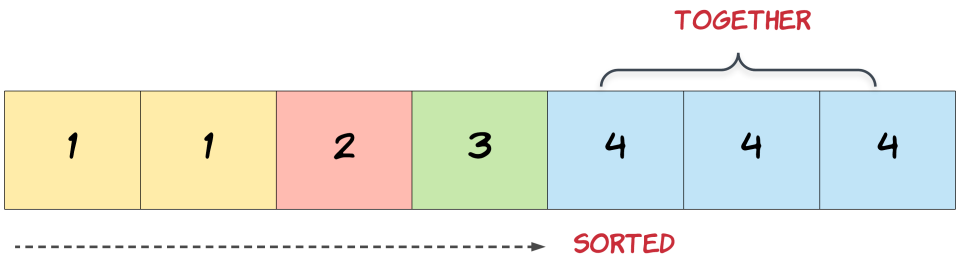
Topics

Array Two Pointers

Companies

Hint 1

In this problem, the key point to focus on is the input array being sorted. As far as duplicate elements are concerned, what is their positioning in the array when the given array is sorted? Look at the image below for the answer. If we know the position of one of the elements, do we also know the positioning of all the duplicate elements?



Hint 2

We need to modify the array in-place and the size of the final array would potentially be smaller than the size of the input array. So, we ought to use a two-pointer approach here. One, that would keep track of the current element in the original array and another one for just the unique elements.

Hint 3

Essentially, once an element is encountered, you simply need to **bypass** its duplicates and move on to the next unique element.

Discussion (654)

</>
@

💡 Discussion Rules

1. Please don't post **any solutions** in this discussion.

2. The problem discussion is for asking questions about the problem or for sharing tips - anything except for solutions.

3. If you'd like to share your solution for feedback and ideas, please head to the solutions tab and post it there.

No comments yet.


```
1
2
3
4
5
class Solution {
    public int removeDuplicates(int[] nums) {


    }
}
```

Saved

Ln 1, Col 1

nums =

[1,1,2]

1

[1,1,2]

</>
Source

FindHeaderBarSize

FindTabBarSize

FindBorderBarSize