

Remove Duplicates in Sorted array.

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).

Brute force:

- We can use hash set to remove / store non-duplicate element and assign it to the array from starting index.

$$T.C \sim O(N + N \log N)$$

Optimal approach:

• Array is sorted

