26. Remove Duplicates from Sorted Array

March 5, 2016 | 582.4K views

Average Rating: 3.96 (373 votes)

Given a sorted array *nums*, remove the duplicates **in-place** such that each element appear only *once* and return the new length.

Do not allocate extra space for another array, you must do this by **modifying the input array in-place** with O(1) extra memory.

Example 1:

```
Given nums = [1,1,2],

Your function should return length = 2, with the first two elements of nums being 1
 and 2 respectively.

It doesn't matter what you leave beyond the returned length.
```

Example 2:

```
Given nums = [0,0,1,1,1,2,2,3,3,4],

Your function should return length = 5, with the first five elements of nums
 being modified to 0, 1, 2, 3, and 4 respectively.

It doesn't matter what values are set beyond the returned length.
```

Clarification:

Confused why the returned value is an integer but your answer is an array?

Note that the input array is passed in by **reference**, which means modification to the input array will be known to the caller as well.

Internally you can think of this:

```
// nums is passed in by reference. (i.e., without making a copy)
int len = removeDuplicates(nums);

// any modification to nums in your function would be known by the caller.
// using the length returned by your function, it prints the first len elements.
for (int i = 0; i < len; i++) {
    print(nums[i]);
}
```

Solution

Approach 1: Two Pointers

Algorithm

Since the array is already sorted, we can keep two pointers i and j, where i is the slow-runner while j is the fast-runner. As long as $nums[i] = nums[j]$, we increment j to skip the duplicate.

When we encounter $nums[j] \neq nums[i]$, the duplicate run has ended so we must copy its value to $nums[i + 1]$. i is then incremented and we repeat the same process again until j reaches the end of array.

Java

Copy

```
public int removeDuplicates(int[] nums) {
    if (nums.length == 0) return 0;
    int i = 0;
    for (int j = 1; j < nums.length; j++) {
        if (nums[j] != nums[i]) {
            i++;
            nums[i] = nums[j];
        }
    }
    return i + 1;
}
```

Complexity analysis

- Time complexity : $O(n)$. Assume that n is the length of array. Each of i and j traverses at most n steps.
- Space complexity : $O(1)$.

Rate this article:

Previous Next

Comments: 314

Sort By

Type comment here... (Markdown is supported)

Preview Post

vsetteacher ★ 110 January 24, 2019 3:23 PM

how did this solution remove duplicates? Looks like it just change first length of n to unique value.

102 Share Reply

SHOW 10 REPLIES

happypotter ★ 56 February 21, 2019 7:00 AM Report

C++ STL 2 lines solution:
int removeDuplicates(vector& nums) {
nums.erase(unique(nums.begin(), nums.end()), nums.end());
return nums.size();
}

Read More

56 Share Reply

SHOW 5 REPLIES

nyasha ★ 51 October 9, 2018 7:10 AM

My Java solution beats 100% (7ms) !

```
class Solution {
    public int removeDuplicates(int[] nums) {
```

Read More

48 Share Reply

SHOW 1 REPLY

stantoncbradley ★ 120 November 5, 2018 1:55 AM Report

with this solution, don't you need to trim the array at the end? otherwise wouldn't

```
let nums = [1,1,2,2,3,3];
removeDuplicates(nums);
// nums: [1,2,3,2,3,3]
```

Read More

117 Share Reply

SHOW 10 REPLIES

zhengzhicong ★ 294 November 11, 2018 12:27 PM Report

python3:

```
class Solution:
    def removeDuplicates(self, nums):
        """
```

Read More

64 Share Reply

SHOW 3 REPLIES

toddniven ★ 46 October 24, 2018 11:18 AM

Is this the obvious solution?

```
    def removeDuplicates(self, nums):
        """
        :type nums: List[int]
```

Read More

46 Share Reply

SHOW 5 REPLIES

terrible_whiteboard ★ 626 May 19, 2020 6:07 PM Report

I made a video if anyone is having trouble understanding the solution (clickable link)
https://youtu.be/rlfsnRY059k

Read More

21 Share Reply

zeeshanakhter1729 ★ 29 April 15, 2019 9:26 PM

C++ solution.

```
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
```

Read More

27 Share Reply

SHOW 2 REPLIES

Qfab ★ 45 July 21, 2018 12:34 AM Report

My JavaScript solution:

```
var removeDuplicates = function(nums) {
    nums.splice(0, nums.length, ...(new Set(nums)));
};
```

Read More

38 Share Reply

SHOW 11 REPLIES

lc2313445 ★ 39 October 1, 2018 2:04 PM

An easy python solution beat 98.3%

def removeDuplicates(nums):

```
    dif nums:
```

Read More

22 Share Reply

SHOW 3 REPLIES