27. Remove Element

March 13, 2016 array (/articles/?tag=array) two-pointers (/articles/?tag=two-pointers)

Question | Editorial Solution

Question

> Given an array and a value, remove all instances of that value in place and return the new length.
>
> Do not allocate extra space for another array, you must do this in place with constant memory.
>
> The order of elements can be changed. It doesn't matter what you leave beyond the new length.
>
> **Example:**
> Given input array *nums* = `[3,2,2,3]`, *val* = `3`
>
> Your function should return length = 2, with the first two elements of *nums* being 2.

Summary

This is a pretty easy problem, but one may get confused by the term "in-place" and thought it is impossible to remove an element from the array without making a copy of the array.

Hints

1. Try two pointers.
2. Did you use the property of "the order of elements can be changed"?
3. What happens when the elements to remove are rare?

Solution

Approach #1 (Two Pointers) [Accepted]

Intuition

Since question asked us to remove all elements of the given value in-place, we have to handle it with $O(1)$ extra space. How to solve it? We can keep two pointers i and j, where i is the slow-runner while j is the fast-runner.

Algorithm

When $nums[j]$ equals to the given value, skip this element by incrementing j. As long as $nums[j] \neq val$, we copy $nums[j]$ to $nums[i]$ and increment both indexes at the same time. Repeat the process until j reaches the end of the array and the new length is i.

This solution is very similar to the solution to Remove Duplicates from Sorted Array (https://leetcode.com/articles/remove-duplicates-sorted-array/).

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

```java
public int removeElement(int[] nums, int val) {
    int i = 0;
    for (int j = 0; j < nums.length; j++) {
        if (nums[j] != val) {
            nums[i] = nums[j];
            i++;
        }
    }
    return i;
}
```

Complexity analysis

- Time complexity : $O(n)$. Assume the array has a total of n elements, both i and j traverse at most $2n$ steps.
- Space complexity : $O(1)$.

Approach #2 (Two Pointers - when elements to remove are rare) [Accepted]

Intuition

Now consider cases where the array contains few elements to remove. For example, $nums = [1, 2, 3, 5, 4], val = 4$. The previous algorithm will do unnecessary copy operation of the first four elements. Another example is $nums = [4, 1, 2, 3, 5], val = 4$. It seems unnecessary to move elements $[1, 2, 3, 5]$ one step left as the problem description mentions that the order of elements could be changed.

Algorithm

When we encounter $nums[i] = val$, we can swap the current element out with the last element and dispose the last one. This essentially reduces the array's size by 1.

Note that the last element that was swapped in could be the value you want to remove itself. But don't worry, in the next iteration we will still check this element.

```java
public int removeElement(int[] nums, int val) {
    int i = 0;
    int n = nums.length;
    while (i < n) {
        if (nums[i] == val) {
            nums[i] = nums[n - 1];
            // reduce array size by one
            n--;
        } else {
            i++;
        }
    }
    return n;
}
```

Complexity analysis

- Time complexity : $O(n)$. Both i and n traverse at most n steps. In this approach, the number of assignment operation is equal to the number of elements to remove. So it is more efficient if elements to remove are rare.
- Space complexity : $O(1)$.

Analysis written by @feelxia, revised by @1337c0d3r.

(/ratings/107/22/?return=/articles/remove-element/) (/ratings/107/22/?return=/articles/remove-element/) (/ratings/107/22/?return=/articles/remove-element/) (/ratings/107/22/

Average Rating: 4.67 (57 votes)

Previous (/articles/linked-list-cycle/)

Next (/articles/contains-duplicate/)

Join the conversation

Signed in as **lixiaozheng.good**.

Post a Reply

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

mkhan31995 commented last month

I was trying to just return the count, and then finally I understood what the question wants.

liscuss.leetcode.com/user/mkhan31995)

leicasper commented last month

Here is my JavaScript solution:

liscuss.leetcode.com/user/leicasper)

let removeElement = function(nums, val) {
let len = nums.length;

if (len === 0) {
return len;
}

let i = len;
while (i--) {
if (nums[i] === val) {
nums.splice(i, 1);
}
}

return nums.length;
}

mh-c commented 2 months ago

@kvelury (https://discuss.leetcode.com/uid/186348)

liscuss.leetcode.com/user/mh-

Read the question carefully.
You need to modify the array itself. Instead of just return the count.

kvelury commented 3 months ago

Can someone tell me why it's not accepting my answer?

liscuss.leetcode.com/user/kvelury)

public int removeElement(int[] nums, int val) {
int countOfVal = 0;
for (int i = 0; i < nums.length; i++) {
if (nums[i] == val) countOfVal++;
}
return nums.length - countOfVal;
}

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

fateflame commented 4 months ago

lis cuss.leetcode.com/user/fateflame)

My solution is similar to solution 2 but adding an step to ensure that nums[n-1], the number swapped with the disposed one, must be a valid number.

```
class Solution {
public:
int removeElement(vector<int>& nums, int val) {
if (nums.empty()) return 0;
int i = 0, j= nums.size();
while (i<nums.size()) {
if (nums[i] == val) {
while (j>i) {
--j;
if (nums[j] != val) {
//switch nums[i] and nums[j]
int temp = nums[i];
nums[i] = nums[j];
nums[j] = temp;
break;
}
}
if (i >= j) break;
}
++i;
}
return i;
}
};
```

rabbit_xin commented 4 months ago

lis cuss.leetcode.com/user/rabbit_xin)

```
class Solution {
public:
int removeElement(vector<int>& nums, int val) {
int insert_index=0;
int cnt=0;
for(int i=0;i<nums.size();i++)
{
if(nums[i]!=val)nums[insert_index++] = nums[i];
else cnt++;
}
return nums.size()-cnt;
}
};
```

YamiOdymel commented 5 months ago

lis cuss.leetcode.com/user/yamiodymel)

/**

- @param {number[]} nums
- @param {number} val
- @return {number}
  */
  var removeElement = function(nums, val) {
  return nums.filter(function(v, i){
  return v !== val
  })
  };

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

yli0-umass-edu commented 6 months ago

avoid the useless write.
liscuss.leetcode.com/user/yli0-

```
public int removeElement(int[] nums, int val) {
    int l = 0, r = nums.length-1;
    while(l < r) {
        while(l < nums.length && nums[l] != val) l++;
        while(r >= 0 && nums[r] == val) r--;
        if(l < r) {
            nums[l] = nums[r];
            nums[r] = val;
            l++;
            r--;
        } else break;
    }
    if(l == r && nums[l] != val) l++;
    return l;
}
```

issue commented 7 months ago

liscuss.leetcode.com/user/issue)

```
int removeElement(vector<int>& nums, int val) {
    auto pend = std::remove_if(nums.begin(), nums.end(), [val](const int elem) {
        return (elem == val);
    });

    return pend - nums.begin();
}
```

whoppers commented 8 months ago

what if the rare number is in the middle, still need to travel to the whole n
liscuss.leetcode.com/user/whoppers)

View original thread (https://discuss.leetcode.com/topic/45)

Load more comments...

Frequently Asked Questions (/faq/) | Terms of Service (/tos/)

Privacy

Copyright © 2017 LeetCode

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)