List of Questions

A. Array / String

1. Merge Sorted Array

Leetcode No: 88 | **Difficulty:** Easy | **Tags:** Array, Two Pointers, Sorting

You are given two integer arrays `nums1` and `nums2`, sorted in **non-decreasing order**, and two integers `m` and `n`, representing the number of elements in `nums1` and `nums2` respectively.

Merge `nums1` and `nums2` into a single array sorted in **non-decreasing order**.

The final sorted array should not be returned by the function, but instead be *stored inside the array* `nums1`. To accommodate this, `nums1` has a length of `m + n`, where the first `m` elements denote the elements that should be merged, and the last `n` elements are set to `0` and should be ignored. `nums2` has a length of `n`.

Example 1:

```
Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3
Output: [1,2,2,3,5,6]
Explanation: The arrays we are merging are [1,2,3] and [2,5,6].
The result of the merge is [1,2,2,3,5,6] with the underlined elements
coming from nums1.
```

Example 2:

```
Input: nums1 = [1], m = 1, nums2 = [], n = 0
Output: [1]
Explanation: The arrays we are merging are [1] and [].
The result of the merge is [1].
```

Example 3:

```
Input: nums1 = [0], m = 0, nums2 = [1], n = 1
Output: [1]
Explanation: The arrays we are merging are [] and [1].
The result of the merge is [1].
Note that because m = 0, there are no elements in nums1. The 0 is only
there to ensure the merge result can fit in nums1.
```

Constraints:

- `nums1.length == m + n`
- `nums2.length == n`
- `0 <= m, n <= 200`

- `1 <= m + n <= 200`
- `-`10^9` <= nums1[i], nums2[j] <= `10^9

Follow up: Can you come up with an algorithm that runs in `O(m + n)` time?

Hints:

- You can easily solve this problem if you simply think about two elements at a time rather than two arrays. We know that each of the individual arrays is sorted. What we don't know is how they will intertwine. Can we take a local decision and arrive at an optimal solution?
- If you simply consider one element each at a time from the two arrays and make a decision and proceed accordingly, you will arrive at the optimal solution.

1. Merge Sorted Array -- Community Solution

☑ Beats 100 % || ✅ Best C++/Java/Python and JavaScript Solution || Two Pointer || STL [Votes: 4285]

Intuition

We are given two sorted arrays nums1 and nums2 of sizes m and n, respectively. We need to merge these two arrays into a single sorted array, and the result should be stored inside nums1. Since nums1 is of size m+n, we can use this extra space to store the merged array. We can iterate through the arrays from the end and place the larger element in the end of nums1.

Approach : *Using STL*

1. Traverse through nums2 and append its elements to the end of nums1 starting from index m.
2. Sort the entire nums1 array using sort() function.

Complexity

- Time complexity: O((m+n)log(m+n))

  due to the sort() function

- Space complexity: O(1)

  We are not using any extra space, so the space complexity is O(1).

Code

C++

```cpp
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int
n) {
        for (int j = 0, i = m; j<n; j++){
            nums1[i] = nums2[j];
            i++;
        }
        sort(nums1.begin(),nums1.end());
    }
};
```

Java

```java
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        for (int j = 0, i = m; j < n; j++) {
            nums1[i] = nums2[j];
            i++;
        }
        Arrays.sort(nums1);
    }
}
```

Python

```python
class Solution(object):
    def merge(self, nums1, m, nums2, n):
      for j in range(n):
          nums1[m+j] = nums2[j]
      nums1.sort()
```

JavaScript

```javascript
var merge = function(nums1, m, nums2, n) {
    for (let i = m, j = 0; j < n; i++, j++) {
        nums1[i] = nums2[j];
    }
    nums1.sort((a, b) => a - b);
};
```

Approach : *Two Pointer*

We can start with two pointers i and j, initialized to m-1 and n-1, respectively. We will also have another pointer k initialized to m+n-1, which will be used to keep track of the position in nums1 where we will be placing the larger element. Then we can start iterating from the end of the arrays i and j, and compare the elements at these positions. We will place the larger element in nums1 at position k, and decrement the corresponding pointer i or j accordingly. We will continue doing this until we have iterated through all the elements in nums2. If there are still elements left in nums1, we don't need to do anything because they are already in their correct place.

Complexity

- Time complexity: O(m+n)

  We are iterating through both arrays once, so the time complexity is O(m+n).

- Space complexity: O(1)

  We are not using any extra space, so the space complexity is O(1).

Code

C++

```cpp
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int
n) {
        int i = m - 1;
        int j = n - 1;
        int k = m + n - 1;

        while (j >= 0) {
            if (i >= 0 && nums1[i] > nums2[j]) {
                nums1[k--] = nums1[i--];
            } else {
                nums1[k--] = nums2[j--];
            }
        }
    }
};
```

Java

```java
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        int i = m - 1;
        int j = n - 1;
        int k = m + n - 1;

        while (j >= 0) {
            if (i >= 0 && nums1[i] > nums2[j]) {
                nums1[k--] = nums1[i--];
            } else {
                nums1[k--] = nums2[j--];
            }
        }
    }
}
```

Python

```python
class Solution(object):
    def merge(self, nums1, m, nums2, n):
        i = m - 1
        j = n - 1
        k = m + n - 1

        while j >= 0:
            if i >= 0 and nums1[i] > nums2[j]:
                nums1[k] = nums1[i]
                i -= 1
            else:
                nums1[k] = nums2[j]
                j -= 1
            k -= 1
```

JavaScript

```javascript
var merge = function(nums1, m, nums2, n) {
    let i = m - 1;
    let j = n - 1;
    let k = m + n - 1;

    while (j >= 0) {
        if (i >= 0 && nums1[i] > nums2[j]) {
            nums1[k--] = nums1[i--];
        } else {
            nums1[k--] = nums2[j--];
        }
    }
};
```

2. Remove Element

Leetcode No: 27 | **Difficulty:** Easy | **Tags:** Array, Two Pointers

Given an integer array `nums` and an integer `val`, remove all occurrences of `val` in `nums` **in-place**. The order of the elements may be changed. Then return *the number of elements in* `nums` *which are not equal to* `val`.

Consider the number of elements in `nums` which are not equal to `val` 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 elements which are not equal to `val`. 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 val = ...; // Value to remove
int[] expectedNums = [...]; // The expected answer with correct length.
                            // It is sorted with no values equaling val.

int k = removeElement(nums, val); // Calls your implementation

assert k == expectedNums.length;
sort(nums, 0, k); // Sort the first k elements of nums
for (int i = 0; i < actualLength; i++) {
    assert nums[i] == expectedNums[i];
}
```

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

Example 1:

```
Input: nums = [3,2,2,3], val = 3
Output: 2, nums = [2,2,_,_]
Explanation: Your function should return k = 2, with the first two
elements of nums being 2.
It does not matter what you leave beyond the returned k (hence they are
underscores).
```

Example 2:

```
Input: nums = [0,1,2,2,3,0,4,2], val = 2
Output: 5, nums = [0,1,4,0,3,_,_,_]
Explanation: Your function should return k = 5, with the first five
elements of nums containing 0, 0, 1, 3, and 4.
Note that the five elements can be returned in any order.
It does not matter what you leave beyond the returned k (hence they are
underscores).
```

Constraints:

- `0 <= nums.length <= 100`
- `0 <= nums[i] <= 50`
- `0 <= val <= 100`

Hints:

- The problem statement clearly asks us to modify the array in-place and it also says that the element beyond the new length of the array can be anything. Given an element, we need to remove all the occurrences of it from the array. We don't technically need to **remove** that element per se, right?
- We can move all the occurrences of this element to the end of the array. Use two pointers!
- Yet another direction of thought is to consider the elements to be removed as non-existent. In a single pass, if we keep copying the visible elements in-place, that should also solve this problem for us.

2. Remove Element -- Community Solution

☑Best 100% || C++ || JAVA || PYTHON || Beginner Friendly🔥🔥🔥 [Votes: 2472]

Intuition

The intuition behind this solution is to iterate through the array and keep track of two pointers: `index` and `i`. The `index` pointer represents the position where the next non-target element should be placed, while the `i` pointer iterates through the array elements. By overwriting the target elements with non-target elements, the solution effectively removes all occurrences of the target value from the array.

Approach

1. Initialize `index` to 0, which represents the current position for the next non-target element.
2. Iterate through each element of the input array using the `i` pointer.
3. For each element `nums[i]`, check if it is equal to the target value.
   - If `nums[i]` is not equal to `val`, it means it is a non-target element.
   - Set `nums[index]` to `nums[i]` to store the non-target element at the current `index` position.
   - Increment `index` by 1 to move to the next position for the next non-target element.
4. Continue this process until all elements in the array have been processed.
5. Finally, return the value of `index`, which represents the length of the modified array.

Complexity

- Time complexity:

$$ O(n) $$ - Space complexity:

$$ O(1) $$

Code

C++

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

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

```python
class Solution:
    def removeElement(self, nums: List[int], val: int) -> int:
        index = 0
        for i in range(len(nums)):
            if nums[i] != val:
                nums[index] = nums[i]
                index += 1
        return index
```

If you are a beginner solve these problems which makes concepts clear for future coding: 1. Two Sum 2. Roman to Integer 3. Palindrome Number 4. Maximum Subarray 5. Remove Element 6. Contains Duplicate 7. Add Two Numbers 8. Majority Element 9. Remove Duplicates from Sorted Array 10. **Practice them in a row for better understanding and please Upvote for more questions.**

If you found my solution helpful, I would greatly appreciate your upvote, as it would motivate me to continue sharing more solutions.

B. Two Pointers

1. Valid Palindrome

Leetcode No: 125 | **Difficulty:** Easy | **Tags:** Two Pointers, String

A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string `s`, return `true` *if it is a **palindrome**, or* `false` *otherwise*.

Example 1:

```
Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.
```

Example 2:

```
Input: s = "race a car"
Output: false
Explanation: "raceacar" is not a palindrome.
```

Example 3:

```
Input: s = " "
Output: true
Explanation: s is an empty string "" after removing non-alphanumeric
characters.
Since an empty string reads the same forward and backward, it is a
palindrome.
```

Constraints:

- $1 <= s.length <= 2 * 10^5$
- `s` consists only of printable ASCII characters.

1. Valid Palindrome -- Community Solution

🚀Super Clean |C++ ☑️|Easy code with explanation | O(n) [Votes: 726]

Intuition

By reading the question you may think that first we need to convert the string to desired form and then check if its a valid palindrome or not.

But all of that is just distraction you can check valid palindrome by simply using two pointers no need to convert.

Approach

If a character is not alphanumeric we can simply ignore it and update our pointer to next character (for both pointers)

and then we check by converting the alphanumeric character to lowercase (tolower() function on numbers has no change) if those two are not equal then return false(not palindrome) else update both pointers and continue.

Finally after all checking if reach at last then string must be a valid palindrome so return true.

Complexity

- Time complexity:$$O(n)$$
- Space complexity:$$O(1)$$ since only use two pointers

Code

```
class Solution {
public:
    bool isPalindrome(string s) {
       int start=0;
       int end=s.size()-1;
       while(start<=end){
           if(!isalnum(s[start])){start++; continue;}
           if(!isalnum(s[end])){end--;continue;}
           if(tolower(s[start])!=tolower(s[end]))return false;
           else{
               start++;
               end--;
           }
       }
       return true;
}
};
```

PS - Please upvote (bottom left) if this solution was helpful :) It motivates me to write more solutions

Python in-place two-pointer solution. [Votes: 382]

```
def isPalindrome(self, s):
    l, r = 0, len(s)-1
    while l < r:
        while l < r and not s[l].isalnum():
            l += 1
        while l <r and not s[r].isalnum():
            r -= 1
        if s[l].lower() != s[r].lower():
            return False
        l +=1; r -= 1
    return True
```