Description | Discussion (56) | Solutions (5.2K) | Submissions

C++ | Auto

287. Find the Duplicate Number

Medium | 18.1K | 2.6K

Companies

Given an array of integers `nums` containing `n + 1` integers where each integer is in the range `[1, n]` inclusive.

There is only **one repeated number** in `nums`, return *this repeated number*.

You must solve the problem **without** modifying the array `nums` and uses only constant extra space.

Example 1:

```
Input: nums = [1,3,4,2,2]
Output: 2
```

Example 2:

```
Input: nums = [3,1,3,4,2]
Output: 3
```

```cpp
class Solution {
public:
    int findDuplicate(vector<int>& nums) {
        int n = nums.size();
        int res;
        sort(nums.begin(),nums.end());
        for(int i = 0 ; i<n-1;i++){
            if(nums[i]==nums[i+1]){
                res = nums[i];
            }
        }
        return res;
    }
};
```

Console | Run | Submit

Problem | Editorial | Submissions | Comments

Move all negative elements to end

Easy | Accuracy: 56.24% | Submissions: 74K+ | Points: 2

Given an unsorted array **arr[]** of size **N** having both negative and positive integers. The task is place all negative element at the end of array without changing the order of positive element and negative element.

Example 1:

Input :
N = 8
arr[] = {1, -1, 3, 2, -7, -5, 11, 6 }
Output :
1 3 2 11 6 -1 -7 -5

Example 2:

C++ (g++ 5.4) | Start Timer

```
// } Driver Code Ends
class Solution{
    public:
    void segregateElements(int arr[],int n)
    {
        // Your code goes here
        int brr[n];
        int i = 0;

        for(int j = 0 ; j<n;j++){
            if(arr[j]>0){
                brr[i]=arr[j];
                i++;
            }
        }
        for(int j = 0 ; j<n;j++){
            if(arr[j]<=0){
                brr[i] = arr[j];
                i++;
            }
        }
        for(int j = 0 ; j<n ; j++){
            arr[j]=brr[j];
        }
    }
};
// } Driver Code Ends
```

Custom Input | Compile & Run | Submit

88. Merge Sorted Array

Hint

Easy 9.3K 878

Companies

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.
```

C++ Auto

```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;

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

Console Run Submit