# 283. Move Zeroes

Hint

Easy 13.9K 353

Companies

Given an integer array `nums`, move all `0`'s to the end of it while maintaining the relative order of the non-zero elements.

**Note** that you must do this in-place without making a copy of the array.

**Example 1:**

```
Input: nums = [0,1,0,3,12]
Output: [1,3,12,0,0]
```

**Example 2:**

```
Input: nums = [0]
Output: [0]
```

**Constraints:**

- $1 <= nums.length <= 10^4$
- $-2^{31} <= nums[i] <= 2^{31} - 1$

**Follow up:** Could you minimize the total number of operations done?

Accepted **2.3M** | Submissions **3.7M** | Acceptance Rate **61.4%**

Brute force:

One approach is we can scan the whole array and count no. of zeroes and at the same time fill the nonzero elements in xtra array. After filling all non zero elements, now add 'count' no. of zeroes at the end.

Optimum approach:

We can use two pointer approach:

We can use two pointer approach

i : This will point to the position where zero is there

j : This will point to the position where non zero element is there.

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| ~~0~~ 1 | ~~1~~ 3 | ~~0~~ 12 | ~~3~~ 0 | ~~12~~ 0 |

i = ~~0~~ ~~1~~ ~~2~~ 3

j = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

```
class Solution {
public:
    void moveZeroes(vector<int>& nums) {
        int i=0,j=0;
        int n=nums.size();
        while(j<n){
            if(nums[i]!=0) {
                i++;
                j++;
            }
            else if(nums[j]==0) j++;
            else{
                swap(nums[i],nums[j]);
                i++;
                j++;
            }
        }
    }
};
```

we can further optimize this code to make it shorter.

```
Zero pointer = 0
for (non zero pointer : 0 to n-1)
{
    if ( a[nonzero pointer] != 0)
    {
        Swap (a[zero pointer ++] , a[nonzero pointer])
    }
}
```

The difference b/w above two codes is that in first code we are doing less no. of swap because we are only incrementing i & j. But in 2nd code we are swapping more no. of times.