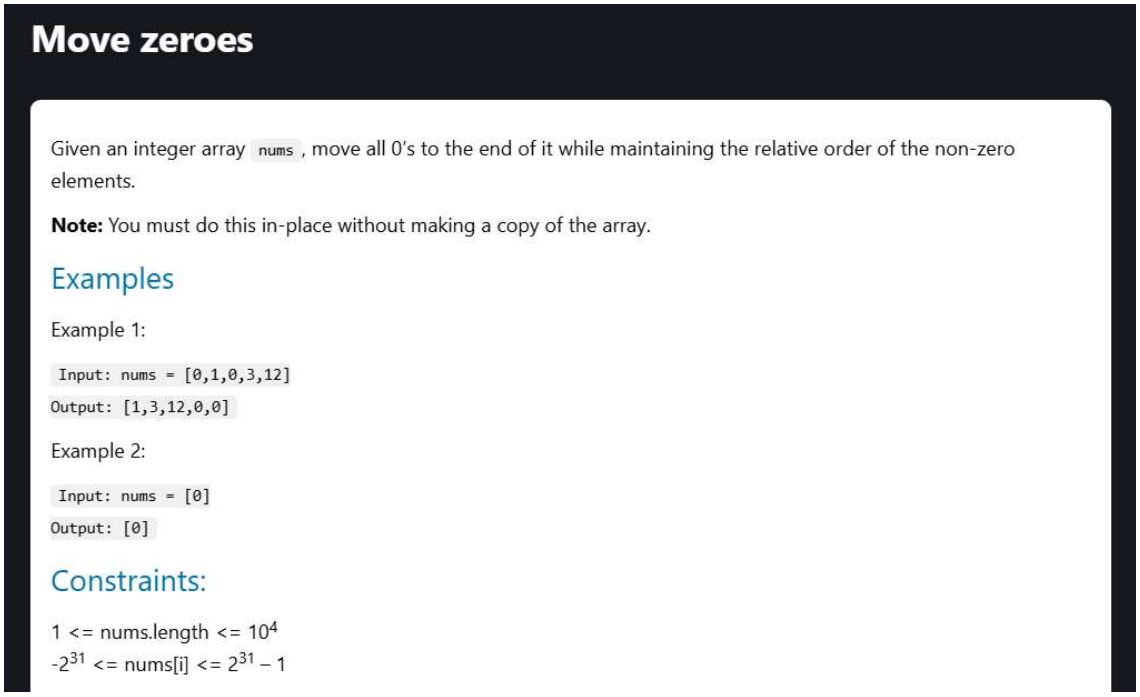

Move zeroes

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: You must do this in-place without making a copy of the array.

Examples

Example 1:

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

Example 2:

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

Constraints:

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

Optimal Approach – Two Pointers

Initialize a pointer `x = 0`.

Loop through the array:

If the current element is not 0, assign it to `nums[x]` and increment `x`.

After the loop, from index `x` to the end of the array, fill all values with 0.

Dry Run

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

`x = 0`

Loop:

`i = 0` → `nums[0] = 0` → skip
`i = 1` → `nums[1] = 1` → `nums[0] = 1`, `x = 1`
`i = 2` → `nums[2] = 0` → skip
`i = 3` → `nums[3] = 3` → `nums[1] = 3`, `x = 2`
`i = 4` → `nums[4] = 12` → `nums[2] = 12`, `x = 3`

Fill remaining with 0s from index 3 onward:

`nums[3] = 0`
`nums[4] = 0`

Final: `nums = [1, 3, 12, 0, 0]`

Time and Space Complexity

Time Complexity: O(n)

One pass to shift non-zero elements.
Another pass to fill in zeros.

Space Complexity: O(1)

In-place modifications with constant extra space.

JavaScript | C++ | C | Java | Python

```javascript
var moveZeroes = function(nums) {
    let x = 0;
    for (let i = 0; i < nums.length; i++) {
        if (nums[i] !== 0) {
            nums[x] = nums[i];
            x++;
        }
    }
    for (let i = x; i < nums.length; i++) {
        nums[i] = 0;
    }
};
```