283. Move zeroes:

I/p: nums = [0, 1, 0, 3, 12]

O/p: [1, 3, 12, 0, 0]

we need to:

* Move all zeros to end
* keep relative order of non-zero elements
* Do this in-place (modify the original array)

Brute force:

1. create a new array
2. copy all non-zero elements in order
3. Fill remaining positions with zeros.

But according to the problem statement we shouldn't use extra array.

Efficient two-pointer approach:

use two pointers:

* lastNonZeroFoundAt → points to the index where the next non-zero should go.
* current → loops through the array.

Steps:

1. Loop through array with current
2. If nums[current] is non-zero:
   * swap nums[lastNonZeroFoundAt] and nums[current]
   * Increment lastNonZeroFoundAt
3. Continue until the end.

```
class Solution {
    public void moveZeroes(int[] nums) {
        int lastNonZeroFoundAt = 0;
        for(int current = 0; current < nums.length; current++){
            if (nums[current] != 0) {
                int temp = nums[lastNonZeroFoundAt];
                nums[lastNonZeroFoundAt] = nums[current];
                nums[current] = temp;
                lastNonZeroFoundAt++;
            }
        }
    }
}
```

Dry Run:

nums = [0, 1, 0, 3, 12]

lastNonZeroFoundAt = 0

current = 0 → nums[0] = 0 → skip

current = 1 → nums[1] = 0 → swap nums[0] and nums[1] →
nums = [1, 0, 0, 3, 12], lastNonZeroFoundAt = 1

current = 2 → nums[2] = 0 → skip

current = 3 → nums[3] = 3 → swap nums[1] and nums[3] →
nums = [1, 3, 0, 0, 12], lastNonZeroFoundAt = 2

current = 4 → nums[4] = 12 → swap nums[2] and nums[4]
nums = [1, 3, 12, 0, 0], lastNonZeroFoundAt = 3

Result = [1, 3, 12, 0, 0]

Time → O(n) space → O(1).