Problem Link:

https://leetcode.com/problems/move-zeroes/

Problem Description:

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.

Problem Approach:

Two Pointers

Sample Test Case:

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

Output: [1,3,12,0,0]

Solution:

We start by initializing a zeroPointer, that'll keep track of the indices where we'll put the numbers by replacing them with 0. So, the solution is quite simple. We start iterating through the array, checking if the current element is 0 or not. If it is, then we do nothing and move to the next iteration. Then, when we find a non-zero element, we swap it with the zero at the zeroPointer index. Also, we increment the zeroPointer with one, so the next element to be swapped gets swapped with the next 0. When we'll be done iterating through the array, we'd have the array with all the non-zero elements in the beginning in order of how they used to be, along with the zeros in the end of the array.

The time and space complexity of this solution is O(n) and O(1) respectively, and that's what makes it an efficient solution.

Code (Python):

```
def moveZeroes(self, nums: List[int]) -> None:
        zeroPointer = 0
        for i in range(0,len(nums)):
            if nums[i] != 0:
                nums[i],nums[zeroPointer] = nums[zeroPointer], nums[i]
                zeroPointer+=1
```