<https://leetcode.com/problems/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** 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 <= 104
* -231 <= nums[i] <= 231 - 1

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

**Attempt 1: 2023-02-28**

**Solution 1: Two Pointers (10 min)**

class Solution {

public void moveZeroes(int[] nums) {

if(nums.length == 1) {

return;

}

int i = 0;

int j = 0;

while(j < nums.length) {

if(nums[j] != 0) {

nums[i++] = nums[j];

}

j++;

}

while(i < nums.length) {

nums[i++] = 0;

}

}

}

Time Complexity:O(n)

Space Complexity:O(1)

**Refer to**

<https://leetcode.com/problems/move-zeroes/solutions/72011/simple-o-n-java-solution-using-insert-index/>

// Shift non-zero values as far forward as possible

// Fill remaining space with zeros

public void moveZeroes(int[] nums) {

if (nums == null || nums.length == 0) return;

int insertPos = 0;

for (int num: nums) {

if (num != 0) nums[insertPos++] = num;

}

while (insertPos < nums.length) {

nums[insertPos++] = 0;

}

}

**Solution 2: Swap as Snowball (30 min)**

class Solution {

public void moveZeroes(int[] nums) {

if(nums.length == 1) {

return;

}

int snowballsize = 0;

for(int i = 0; i < nums.length; i++) {

if(nums[i] == 0) {

snowballsize++;

} else if(snowballsize > 0) {

nums[i - snowballsize] = nums[i];

nums[i] = 0;

}

}

}

}

Time Complexity:O(n)

Space Complexity:O(1)

**Refer to**

<https://leetcode.com/problems/move-zeroes/solutions/172432/the-easiest-but-unusual-snowball-java-solution-beats-100-o-n-clear-explanation/>

**THE EASIEST but UNUSUAL snowball JAVA solution BEATS 100% (O(n)) + clear explanation**

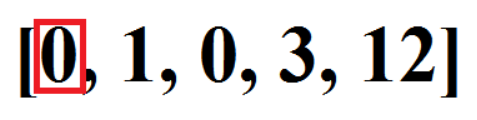
The idea is that we go through the array and gather all zeros on our road.



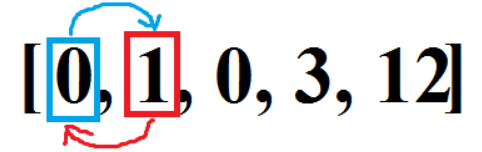
Let's take our example:

the first step - we meet 0.

Okay, just remember that now the size of our snowball is 1. Go further.



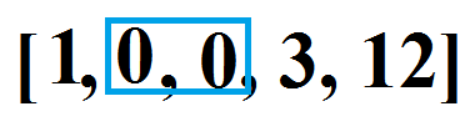
Next step - we encounter 1. Swap the most left 0 of our snowball with element 1.



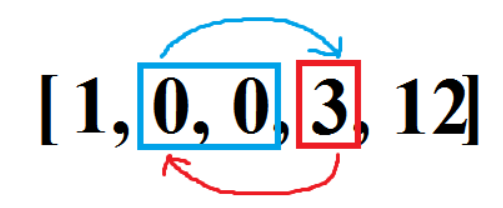
Next step - we encounter 0 again!



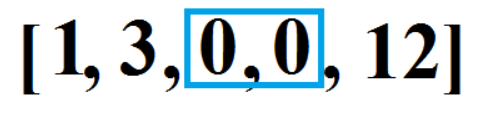
Our ball gets bigger, now its size = 2.



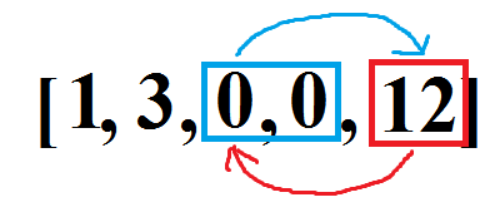
Next step - 3. Swap again with the most left zero.



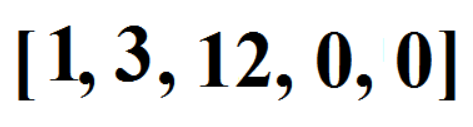
Looks like our zeros roll all the time



Next step - 12. Swap again:



We reached finish! Congratulations!



Here's the code

class Solution {

public void moveZeroes(int[] nums) {

int snowBallSize = 0;

for (int i=0;i<nums.length;i++){

if (nums[i]==0){

snowBallSize++;

}

else if (snowBallSize > 0) {

int t = nums[i];

nums[i]=0;

nums[i-snowBallSize]=t;

}

}

}

}