Move Zeros In-Place

Give DSA notes for this problem : 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.

Here are concise **DSA notes** for the problem of moving all zeros to the end of an array while maintaining the relative order of the non-zero elements:

Problem Analysis

- Input: An integer array `nums`.
- Output: Modify the array `nums` in-place to move all `0`s to the end, maintaining the relative order of non-zero elements.
- Constraints:
  - Do not use extra space (solve in-place).
  - Maintain the order of non-zero elements.

Approach

1. Two-Pointer Technique

- Use two pointers:
  - `i`: Keeps track of the current index where a non-zero element should be placed.
  - `j`: Traverses the array.

Steps:

1. Traverse the array with pointer `j`.
2. When `nums[j] != 0`:

    - Swap `nums[i]` and `nums[j]` if `i != j`.
    - Increment `i`.
3. At the end of traversal, all non-zero elements will be at the front in their original order, and zeros will be at the end.

Code Snippet (Two Pointers)

java

```java
public void moveZeroes(int[] nums) {
    int i = 0; // Points to the index where the next non-zero should go
    for (int j = 0; j < nums.length; j++) {
        if (nums[j] != 0) {
            // Swap only if i and j are different
            if (i != j) {
                int temp = nums[i];
                nums[i] = nums[j];
                nums[j] = temp;
            }
            i++; // Move pointer for non-zero placement
        }
    }
}
```

Key Points

1. **Time Complexity:**
    - Traversing the array once: $O(n)$.
    - Swapping elements: $O(1)$ per swap.
      Overall Complexity: $O(n)$.
2. **Space Complexity:**

    - No extra space used apart from variables.
      Overall Complexity: $O(1)$.

3. **In-Place Operation:**
    - Modifies the input array directly.

Optimization Thoughts

- If the input array is already mostly sorted with zeros at the end, the algorithm will still traverse all elements, but swaps will be minimal.

Variants of the Problem

1. Move zeros to the **beginning** of the array.
2. Move a different target element (e.g., `-1`) to the end while maintaining the order of other elements.
3. Maintain the order but use a stable sorting technique (less efficient for in-place).

Would you like a more detailed walkthrough or practice variations for this problem?