Sort Array of 0 and 1s

Given an array consisting of only 0s and 1s, you need to sort the array in-place such that all the 0s come before the 1s.

**Approach 1: Using Two Pointer approach to sort array of 0s and 1s.**

It takes a vector arr as input, representing the array of 0s and 1s.

It initializes two pointers, left and right, at the beginning and end of the array, respectively.

Using a while loop, it moves the left pointer towards the right until it encounters a 1, and the right pointer towards the left until it encounters a 0.

If the left pointer is still to the left of the right pointer, it swaps the values at left and right indices.

Finally, it returns the sorted array.

**Time Complexity:**

**The approach has a time complexity of O(n), where n is the size of the input array.**

**Space Complexity:**

**The approach use constant space, O(1), as they perform the sorting in-place without using any additional data structures that scale with the input size.**

**Approach 2: Using an Optimized Two Pointer approach to sort of 0s and 1s.**

It follows a similar approach as sort01s, but instead of using nested while loops, it uses if-else conditions to handle the pointer movements and swaps.

The rest of the logic and steps are the same as sort01s.

**Time Complexity:**

**The approach has a time complexity of O(n), where n is the size of the input array.**

**Space Complexity:**

**The approach use constant space, O(1), as they perform the sorting in-place without using any additional data structures that scale with the input size.**

**Both approaches are valid and provide the correct sorted array.** You can use either of them based on your preference or coding style.