11. Sort an array of 0's 1's and 2's

Monday, March 13, 2023 12:49 PM

75. Sort Colors

Hint

Medium 14.2K 512

Companies

Given an array `nums` with `n` objects colored red, white, or blue, sort them **in-place** so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

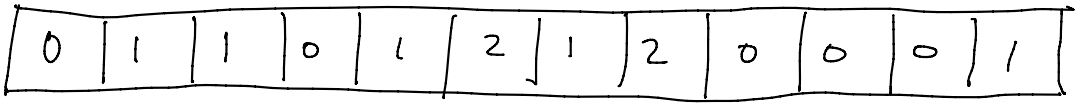
We will use the integers `0`, `1`, and `2` to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Example 1:

```
Input: nums = [2,0,2,1,1,0]
Output: [0,0,1,1,2,2]
```

arr[] ⇒

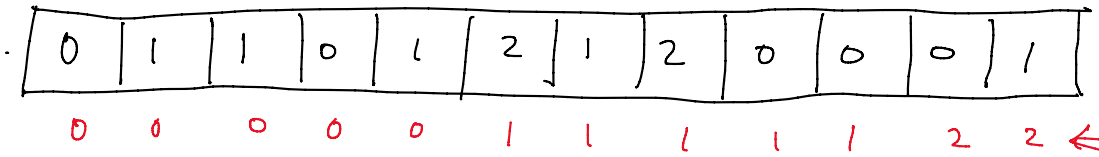


Brute force

* Sorting.

T.C ⇒ O(NlogN) S.C ⇒ O(1)

Optimal Approach (Counting Sort)



0 0 0 0 0 1 1 1 1 1 2 2

→ Linear traverse the array

→ count the no of '0', '1', '2'

→ Run the loop for the 5 times insert '0',

0 → 5

1 → 5

2 → 2

→ Run the loop for the 5 times insert '0', again 5 times insert '1', again 2 times insert '2'.

2 → 2

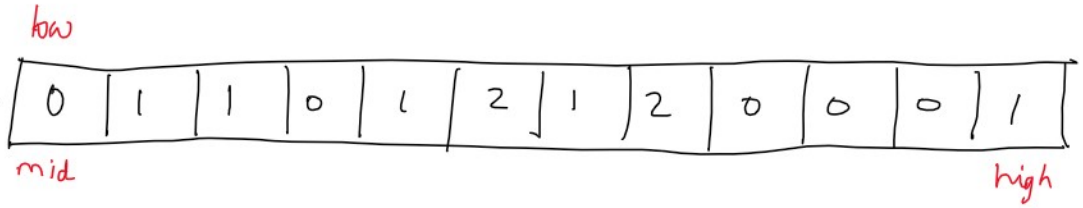
T.C ⇒ O(2N)

3rd Approach

Dutch National Flag Algorithm

→ Consider three pointer → low
→ mid
→ high

→ Place the low, mid at the start and high at the last



→ This algorithm is based on the fact.

[0... low-1] → 0 (left side)

[high+1 ... n] → 2 (right side)

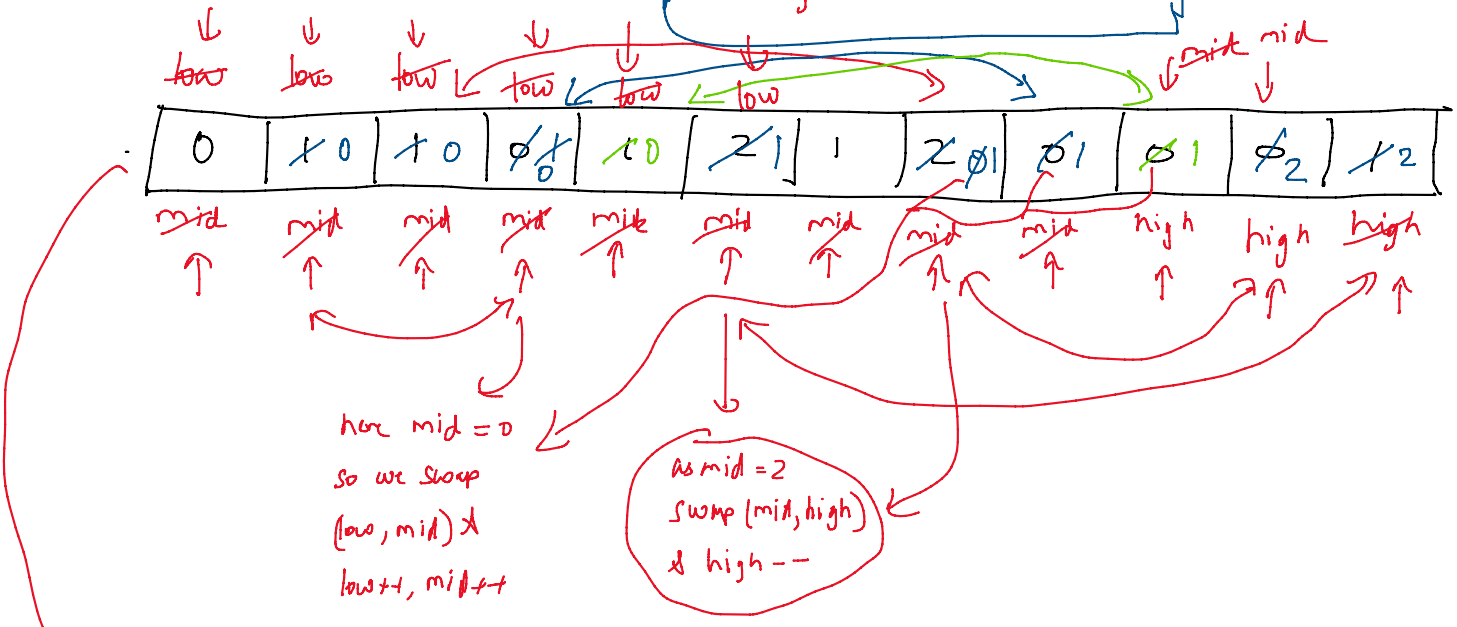
→ will move mid pointer unless and until the mid pointer cross the high pointer

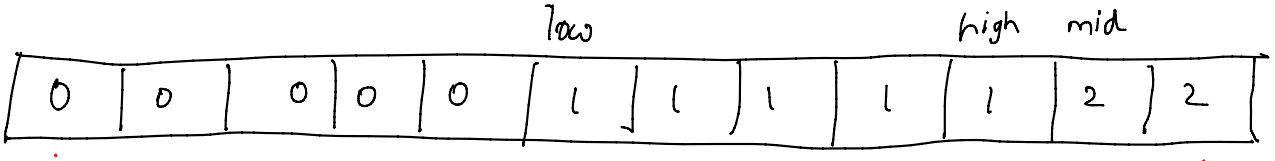
→ While moving the mid pointer we will have 3 checks

mid pointing to 0 → swap (a[low], a[mid]) → low++, mid++

mid pointing to 1 → mid++

mid pointing to 2 → swap (a[mid], a[high]) → high--





a[0 ... low-1] ⇒ 0

a[low .. mid-1] ⇒ 1

a[high+1 ....] ⇒ 2

T.C ⇒ O(N) S.C ⇒ O(1)

It work in one pass

It work in one pass

```cpp
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int low = 0;
        int mid = 0;
        int high = nums.size() - 1;

        while(mid <= high){
            switch(nums[mid]){

                //if the element is 0
                case 0:
                    swap(nums[low++], nums[mid++]);
                    break;

                // if the element is 1
                case 1:
                    mid++;
                    break;

                // if the element is 2
                case 2:
                    swap(nums[mid], nums[high--]);
                    break;
            }
        }
    }
};
```