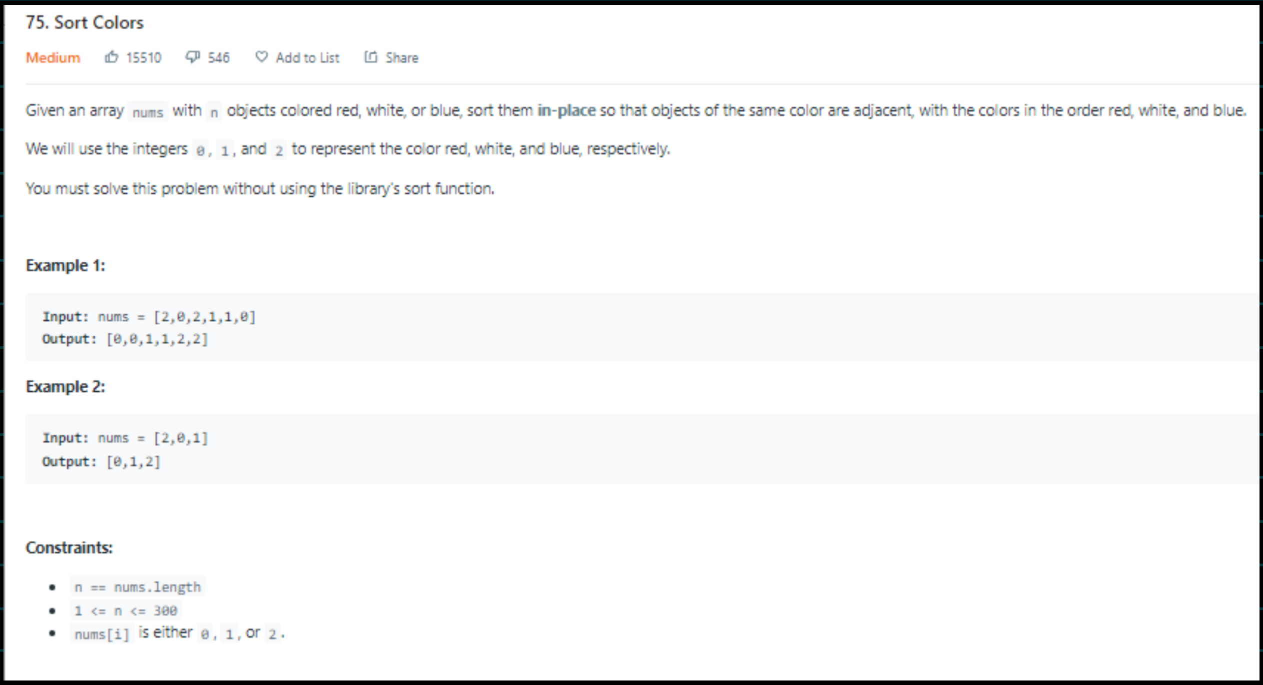
Sort colors



75. Sort Colors

Medium 👍 15510 👎 546 ♡ Add to List ⧉ Share

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]
```

Example 2:

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

Constraints:

- `n == nums.length`
- `1 <= n <= 300`
- `nums[i]` is either `0`, `1`, or `2`.

Brute-force :-

Brute is very Simple, However we are not allowed use sort function

we use merge Sort Algorithm

T.C - O(nlogn)

S.C - O(n)

Better approch :-

Since we are given with 0, 1 and 2 we can create 3 variable which will store count of 0, 1 and 2 and we can run 3 loop to manually override array

$\overrightarrow{[2, 0, 2, 1, 1, 0]}$

T.C - O(n)

S.C - O(1)

count1 = 2
count2 = 2
count0 = 2

```
int index = 0
→ while( count0 != 0)
        arr[index] = 0
        index++
        count --;
```

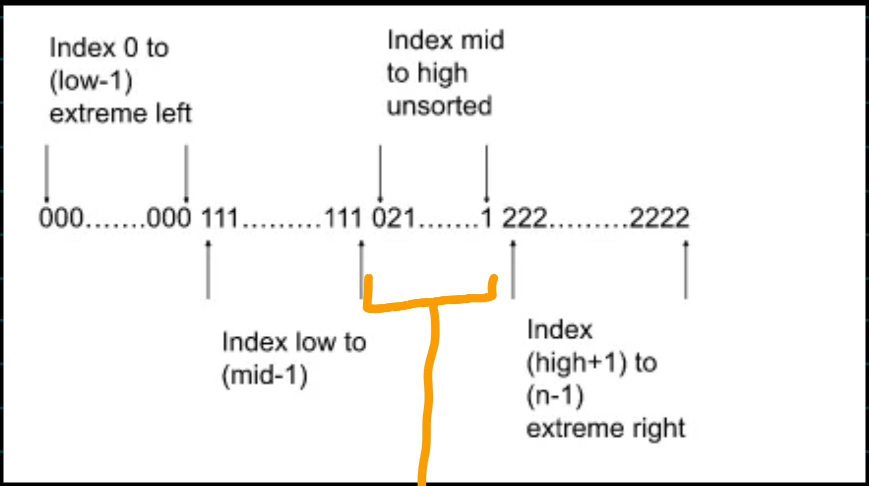
Optimize Approch :- (DNF)

we can observe one that is 0s, 1s, 2s will at some specific position of sorted array

arr [0 .... low-1] contains 0. [left part]

arr [low... mid-1] have 1.

arr [High+1 .... n-1] have 2. [Right part]



Index 0 to (low-1) extreme left
Index mid to high unsorted
000.......000 111.........111 021.......1 222.........2222
Index low to (mid-1)
Index (high+1) to (n-1) extreme right

Taken from Take you forward

unsorted Array

low

[2, 0, 2, 1, 1, 0]

mid

High

mid at first index

High at last index

Low is also at first index because there is No element before that

→ Run while loop (mid ≤ high)

↳ 3 values

mid = ⟶ 0s swap mid with low and increment low++, mid++

0 is at Sorted position No need to touch

mid = → 1s Just increase mid++ since after 0 1 will be correct position according mid

→ 2s swap mid with high and decrease High, Don't increase mid since we don't what we got from high after swap

low ↓

```
  0   1   2   3   4   5
[ 2,  0,  2,  1,  1,  0 ]
                      ↑
mid ↗                 High = n-1
```

low = 0
mid = 0
High = 5

1st

(a[mid] == 2) swap [0, 2]

```
low → [ 0, 0, 2, 1, 1, 2 ]
     mid ↗           ↑
                     High
```

2nd
if (0) swap (0, 0)
↙ mid, mid++   ↓ low, low++

```
[ 0, 0, 2, 1, 1, 2 ]
  ↗             ↑
low, mid        High
```

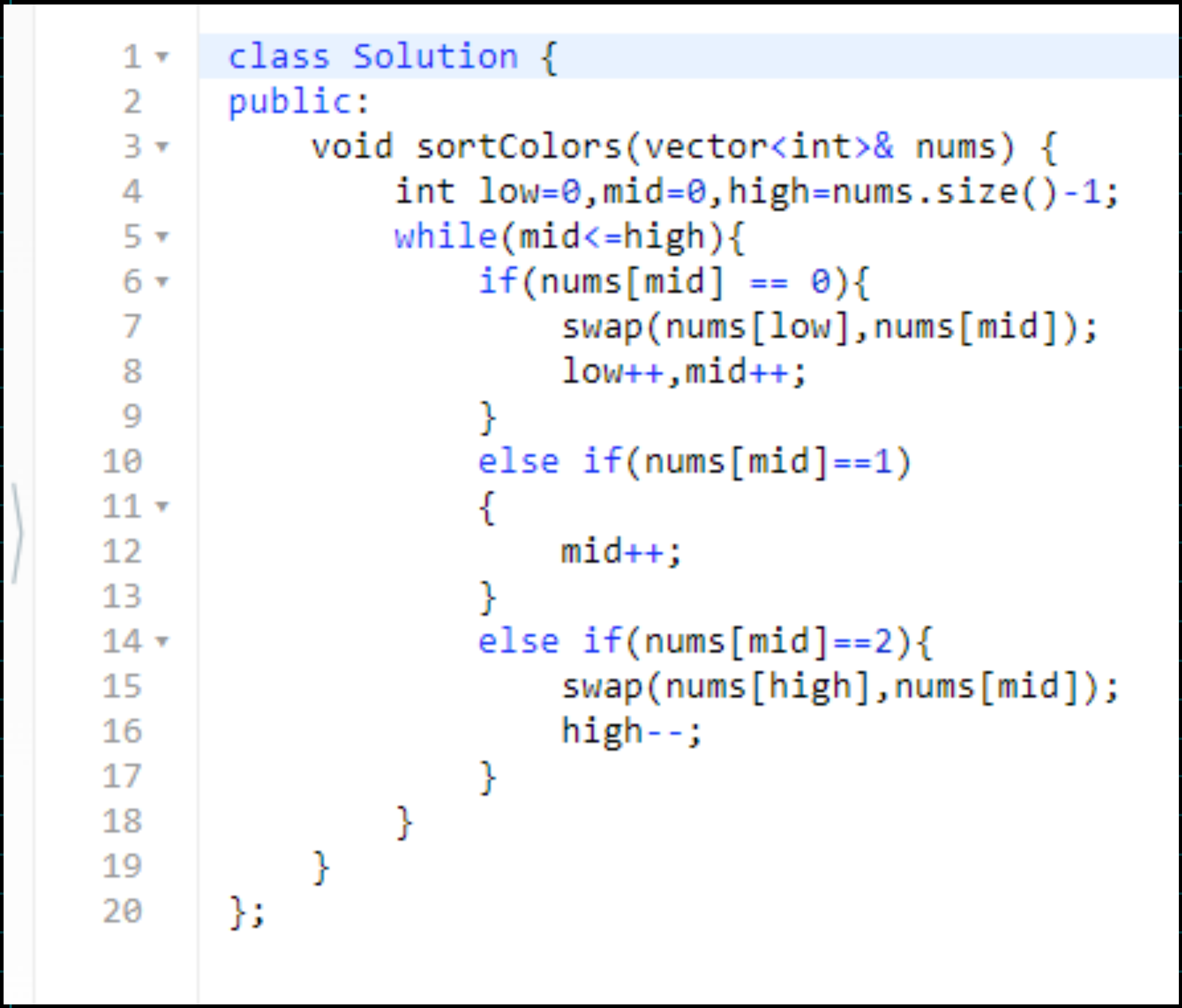
```
[ 0, 0, 2, 1, 1, 2 ]
        ↗     ↑
   low, mid   High
```

```
[ 0, 0, 1, 1, 2, 2 ]
        ↗  ↑
low, mid   High
```

if (a[mid] == 1)
mid++

```
[ 0, 0, 1, 1, 2, 2 ]   (Result
      ↗ ↗ ↖
   low mid  High
```

Optimal code



```cpp
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int low=0,mid=0,high=nums.size()-1;
        while(mid<=high){
            if(nums[mid] == 0){
                swap(nums[low],nums[mid]);
                low++,mid++;
            }
            else if(nums[mid]==1)
            {
                mid++;
            }
            else if(nums[mid]==2){
                swap(nums[high],nums[mid]);
                high--;
            }
        }
    }
};
```

TC - $O(n)$

S.C - $O(1)$