2.Medium\02.Sort_0_1_2.cpp

```cpp
/*
QUESTION:-
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]
*/

/*
APPROACH:-
-> Initialize three pointers: low at the beginning of the array, mid at the beginning of the array, and high at the end of the array.
-> Iterate through the array while the mid pointer is less than or equal to the high pointer:
1. If the current element at the mid pointer is 0 (red), we swap it with the element at the low pointer and increment both low and mid pointers. This ensures that red elements are moved to the left side of the array.
2. If the current element at the mid pointer is 1 (white), we simply increment the mid pointer. This keeps white elements in the middle of the array.
3. If the current element at the mid pointer is 2 (blue), we swap it with the element at the high pointer and decrement the high pointer. This ensures that blue elements are moved to the right side of the array.

Repeat step 2 until the mid pointer crosses the high pointer.
At the end of the algorithm, the array will be sorted in the desired order.
*/

// CODE:-
void sortColors(vector<int> &nums)
{
    int low = 0, mid = 0, high = nums.size() - 1;
    while (mid <= high)
    {
        if (nums[mid] == 0)
            swap(nums[mid++], nums[low++]);
        else if (nums[mid] == 1)
            mid++;
        else
            swap(nums[mid], nums[high--]);
    }
}

```

```
45 // TIME COMPLEXITY = O(N)
46 // SPACE COMPLEXITY = O(0)
```