Experiment 6.1

Student Name: Amit Kumar **UID: 22BCS16121**
Branch: CSE **Section: 641/A**
Semester: 6th **DOP: 25/02/2025**
Subject: AP **Subject Code:22CSP-351**

Aim:

Problem : : Sort Colors

Problem statement: 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]

Algorithm:

1. **Initialization**:
- **low**: Points to the beginning of the array and will be used for placing 0s.
- **mid**: Points to the current element being processed.
- **high**: Points to the end of the array and will be used for placing 2s.

2. **Process the array with the three pointers** (low, mid, and high):
- **While** mid <= high:
    - **If nums[mid] == 0**:
        - Swap nums[low] and nums[mid] (move the 0 to the left side).
        - Increment both low and mid (since you've processed the current mid).
    - **Else if nums[mid] == 1**:
        - Simply increment mid (since 1 is already in the correct position).
    - **Else (nums[mid] == 2)**:
        - Swap nums[mid] and nums[high] (move the 2 to the right side).
        - Decrement high (since the high pointer is now in the correct position).
        -

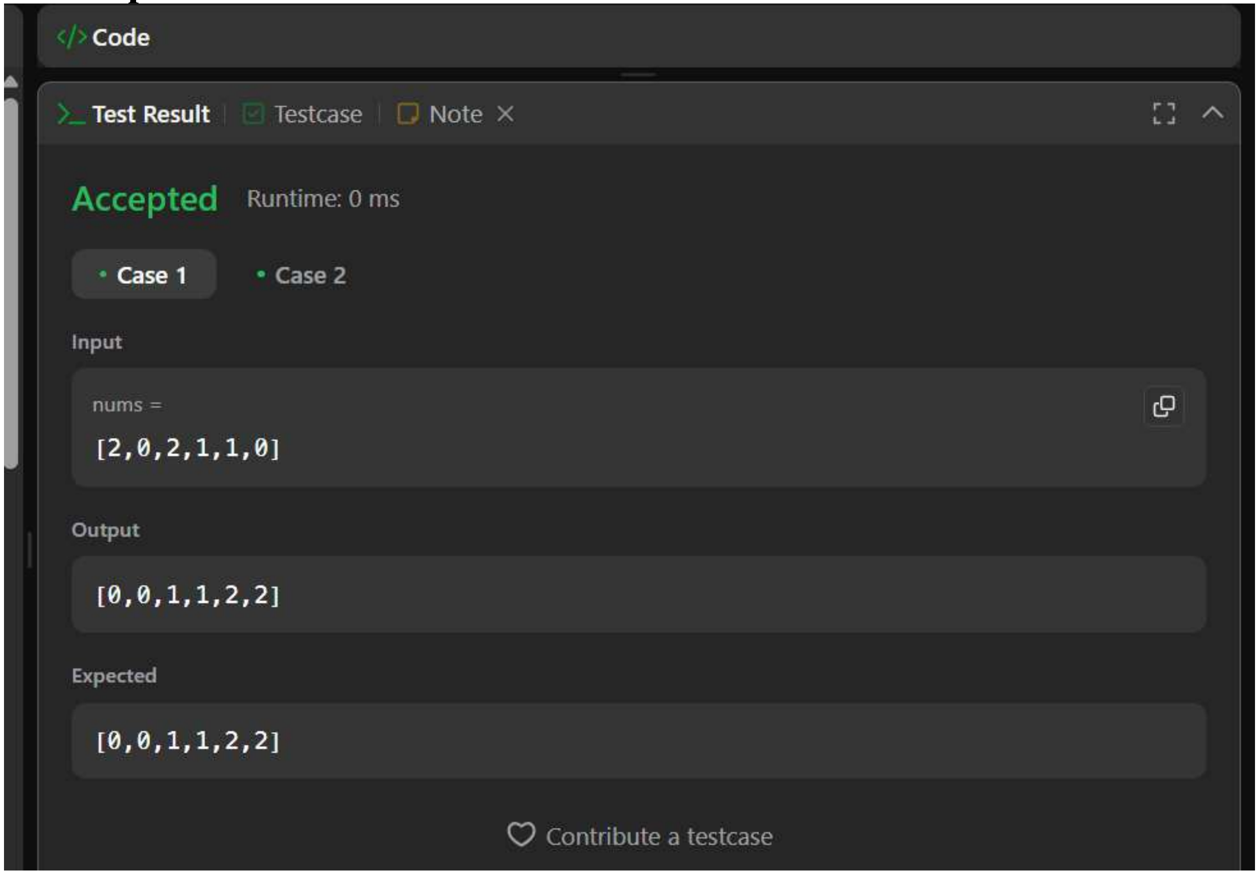
3. **Termination**:
- The algorithm stops when mid surpasses high, meaning all elements have been correctly sorted.

Code:

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

        // Using the Dutch National Flag algorithm
        while (mid <= high) {
            if (nums[mid] == 0) {
                // Swap nums[low] and nums[mid], then move both pointers
                swap(nums[low], nums[mid]);
                low++;
                mid++;
            } else if (nums[mid] == 1) {
                // Move mid pointer if it's 1
                mid++;
            } else {
                // Swap nums[mid] and nums[high], then move high pointer
                swap(nums[mid], nums[high]);
                high--;
            }
        }
    }
};
```

Output:



Accepted Runtime: 0 ms

Input

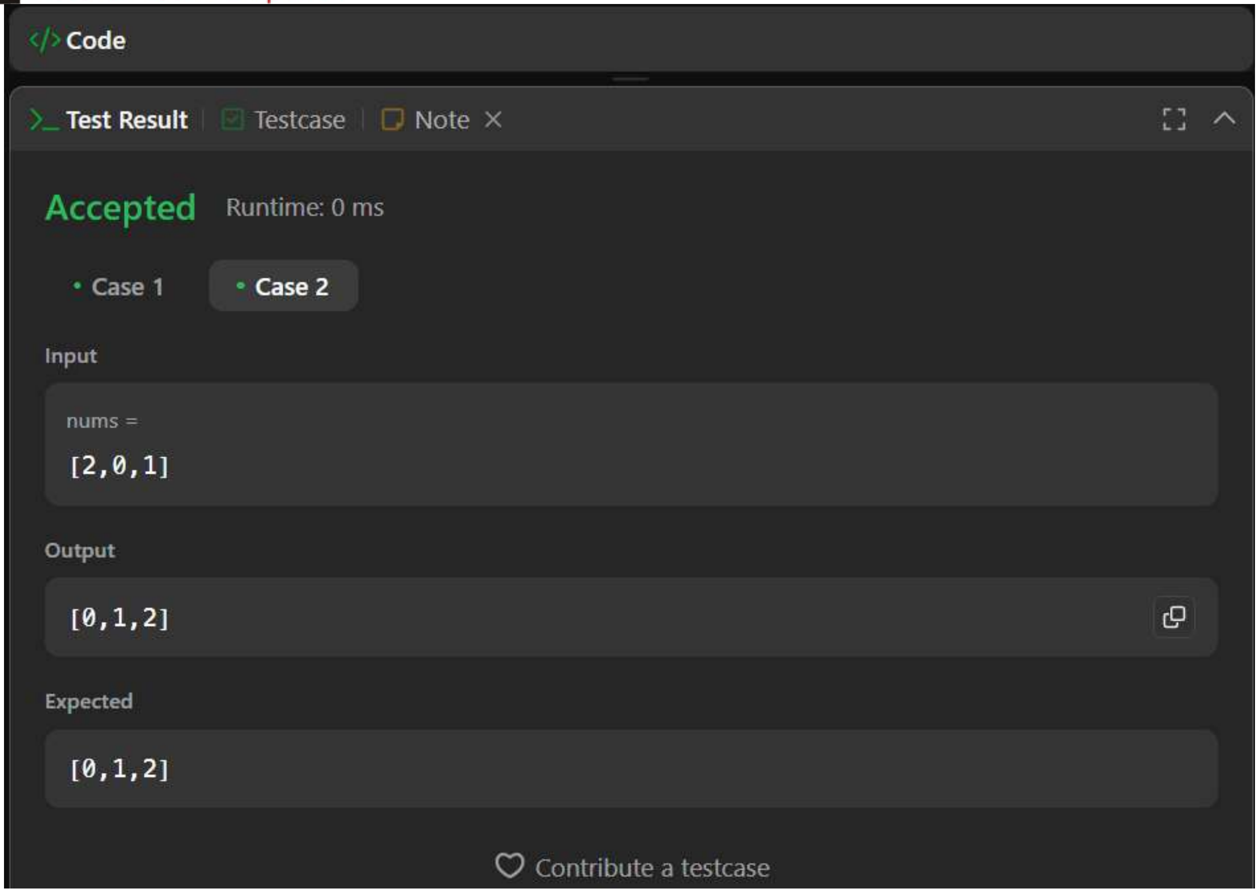
```
nums =
[2,0,2,1,1,0]
```

Output

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

Expected

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



Accepted Runtime: 0 ms

Input

```
nums =
[2,0,1]
```

Output

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

Expected

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