Experiment-2.1

Student Name: Muskan Sai UID: 22BCS13525

Branch: BE-CSE Section/Group: KPIT-901/B Semester: 6th DateofPerformance: 20/02/25

Subject Name: AP Lab - 2 **Subject Code:** 22CSP-351

1. Aim: Sort colors

2. Objective:

The objective of the **Sort Colors** problem is to **sort an array of integers** representing colors (0 for Red, 1 for White, 2 for Blue) **in-place** such that all occurrences of:

- 0s (Red) come first
- Followed by 1s (White)
- Followed by 2s (Blue)

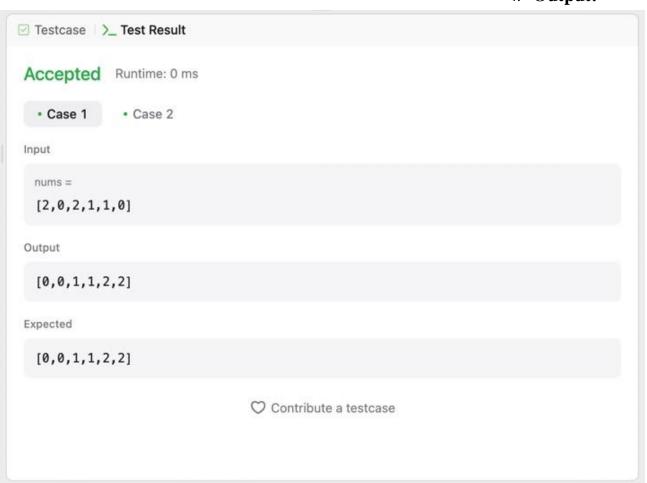
3. Implementation/Code:

```
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 { // nums[mid] == 2
    swap(nums[mid], nums[high]);
}</pre>
```

```
high--;
}
}
```

4. Output:





5. Learning Outcomes:

- Understanding the Dutch National Flag Algorithm
- Understand **swap-based sorting** techniques.
- Understand **swap-based sorting** techniques.

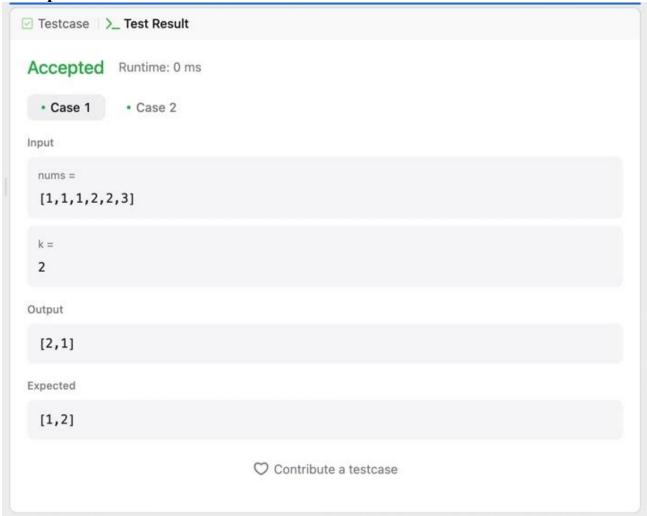
QUESTION 2

- **1. Aim:** Top K Frequent Elements.
- 2. **Objective:** Given an integer array nums and an integer k, the objective of this problem is to **return the k most frequent elements** in the array. This problem tests your ability to manipulate and process data efficiently, specifically focusing on how to manage frequency counting and extract top elements in an optimized manner.

3. Implementation/Code:

```
vector<int> topKFrequent(vector<int>& nums, int k) {
unordered_map<int, int> freq;
for (int num : nums) {
freq[num]++;
priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>>
minHeap;
for (auto& pair : freq) {
minHeap.push({pair.second, pair.first});
if (minHeap.size() > k) {
minHeap.pop();
}
}
vector<int> result;
while (!minHeap.empty()) {
result.push_back(minHeap.top().second);
minHeap.pop();
}
return result;
}
```

4. Output:



5. Learning Outcome:

- I. Learn the Basic Concept of Frequency-Based Searching
- II. Learn to Use Built-in Methods for Efficient Processing
- III. Understanding Data Structures for Efficient Retrival