

**WORKSHEET-5**

**Student Name:** Ajay

**UID:** 22BCS12696

**Branch:** CSE

**Section/Group:** NTPP-603-B

**Semester:** 6th

**Date of Performance:** 20/2/25

**Subject Name:** AP-2

**Subject Code:** 22CSP-351

**Aim(i):** 88. You are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n, representing the number of elements in nums1 and nums2 respectively. Merge nums1 and nums2 into a single array sorted in non-decreasing order.

**Source Code:**

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int midx = m - 1;
        int nidx = n - 1;
        int right = m + n - 1;

        while (nidx >= 0) {
            if (midx >= 0 && nums1[midx] > nums2[nidx]) {
                nums1[right] = nums1[midx];
                midx--;
            } else {
                nums1[right] = nums2[nidx];
                nidx--;
            }
            right--; } };
```

## OUTPUT:

✓ Testcase | >\_ Test Result

Case 1

Case 2

Case 3

+

nums1 =

[1, 2, 3, 0, 0, 0]

m =

3

nums2 =

[2, 5, 6]

Accepted 59 / 59 testcases passed

ILB2SjEs1 submitted at Feb 20, 2025 10:01

Editorial

Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

12.32 MB | Beats 39.42%

100%

95.08% of solutions used 0 ms of runtime

50%

0%

1ms

2ms

3ms

4ms

1ms

2ms

3ms

4ms

Code | C++

## LEARNING OUTCOME:

1. We learnt Merge Sort.
2. We learnt how to sort Arrays.

**Aim(ii): 278.** Suppose you have n versions [1, 2, ..., n] and you want to find out the first bad one, which causes all the following ones to be bad.

You are given an API `bool isBadVersion(version)` which returns whether version is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

**Source Code:**

```
class Solution {
public:
    int firstBadVersion(int n){
        long long l = 1, r = n;
        long long m, res = n;
        while(l <= r){
            m = l + (r - l) / 2;
            if(isBadVersion(m) == 1){
                r = m - 1;
                res = min(res, m);
            } else {
                l = m + 1;
            }
        }
        return res;
    }
};
```

**OUTPUT:**

Accepted Runtime: 0 ms

• Case 1

• Case 2

Input

n =

5

bad =

4

Output

.

ILB2SrjEs1 submitted at Feb 20, 2025 10:13

Editorial

Solution

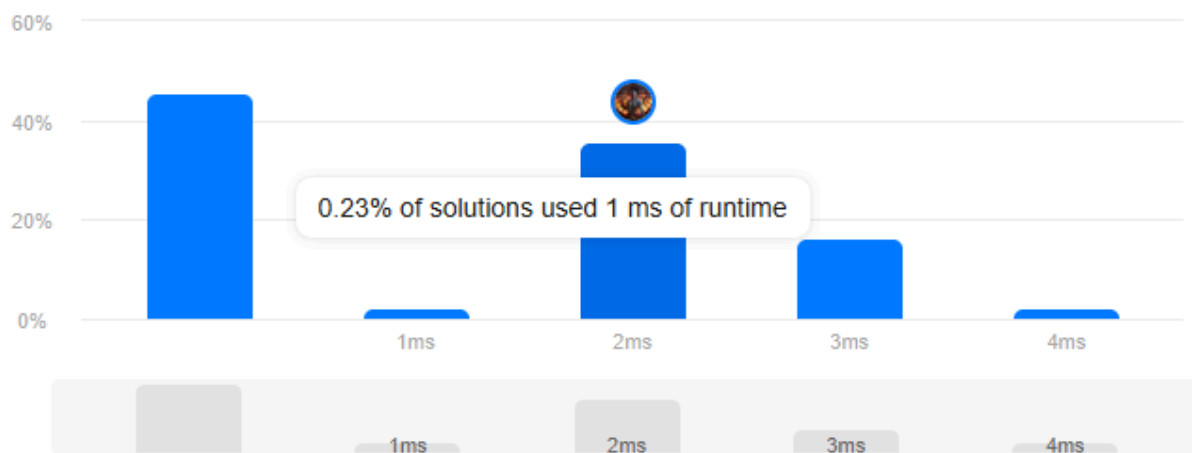
Runtime

2 ms | Beats 54.39%

Analyze Complexity

Memory

7.91 MB | Beats 38.04%



Code | C++

```
class Solution {
public:
    int firstBadVersion(int n){
        long long l = 1, r = n;
        long long m, res = n;
        while(l <= r){
            m = l + (r - l) / 2;
```

## Learning Outcomes

1. We learnt how to use Binary Search.
2. We learnt Edge case Handling.

**Aim(iii):** 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.

**Source Code:**

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        unordered_map<int, int> count = {{0, 0}, {1, 0}, {2, 0}};

        for (int num : nums) {
            count[num]++;
        }

        int idx = 0;
        for (int color = 0; color < 3; color++) {
            int freq = count[color];
            for (int j = 0; j < freq; j++) {
                nums[idx] = color;
                idx++;
            }
        }
    }
};
```

## OUTPUT:

✓ Testcase | > Test Result

**Accepted** Runtime: 0 ms

• Case 1 • Case 2

Input

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


Output

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

Expected

**Accepted** 89 / 89 testcases passed

 ILB2SrijEs1 submitted at Feb 20, 2025 10:20

 Editorial

 Solution

⌚ Runtime

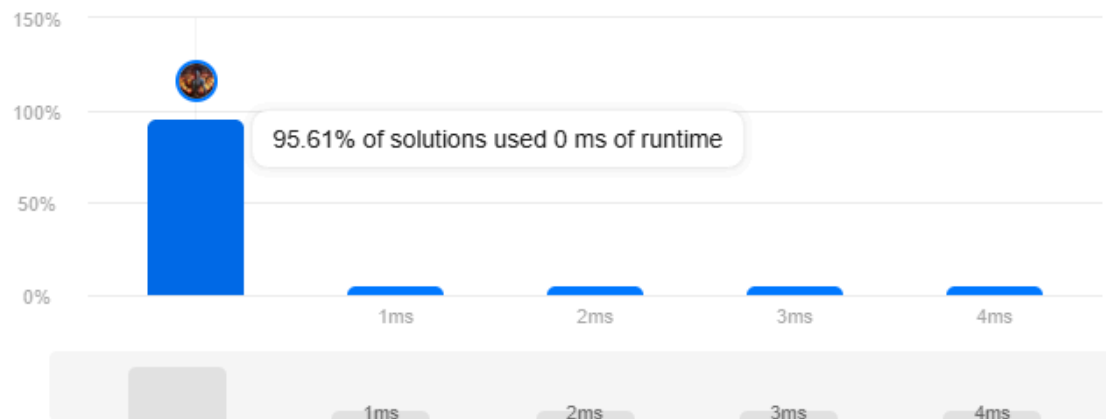


💾 Memory

0 ms | Beats 100.00% 🌿

11.80 MB | Beats 31.23%

🔗 Analyze Complexity



## **Learning Outcomes**

1. We learnt Counting Sort.
2. Usage of a Hash Map.