## **Experiment-4**

**Student Name:** Mohammad Saiful Haque **UID:** 22BCS15656

**Branch:** BE-CSE **Section/Group:** KPIT-901/B **Semester:** 6<sup>th</sup> **Date of Performance:**18/01/25

**Subject Name:** Advanced Programming Lab - 2 **Subject Code:** 22CSP-351

**1. Aim:** Sorting and Searching.

1. Problem: 88. Merge Sorted Array.

2. Problem: 347. Top K Frequent Elements.

#### 2. Objective:

- 1. Merge two sorted arrays nums1 and nums2 in-place without using extra space.
- 2. Find the k most frequent elements in an array using an optimized approach.

#### 3. Implementation/Code:

```
1.)
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int i = m - 1, j = n - 1, k = m + n - 1;
        while (i >= 0 && j >= 0) {
            if (nums1[i] > nums2[j]) {
                 nums1[k--] = nums1[i--];
            } else {
                 nums1[k--] = nums2[j--];
            }
            while (j >= 0) {
                     nums1[k--] = nums2[j--];
            }
        }
```

```
};
2.)
class Solution {
public:
  vector<int> topKFrequent(vector<int>& nums, int k) {
     unordered_map<int, int> freq;
     for (int num: nums) {
       freq[num]++;
     }
     vector<vector<int>>> buckets(nums.size() + 1);
     for (auto& [num, count]: freq) {
       buckets[count].push_back(num);
     }
     vector<int> result;
     for (int i = nums.size(); i \ge 0 \&\& result.size() < k; --i) {
       for (int num : buckets[i]) {
          result.push_back(num);
          if (result.size() == k) return result;
        }
     return result;
};
```

# 4. Output:

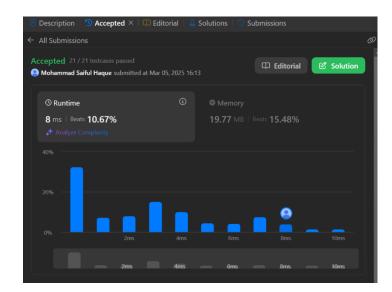
1.





2.





# 5. Time Complexity:

- 1. O(m+n)
- 2. O(n)

### **6. Space Complexity:**

- 1. O(1)
- 2. O(n)

### 7. Learning Outcome:

- 1. Efficient merging using the two-pointer technique.
- 2. Modifying arrays in-place to optimize space.
- 3. Working with sorted arrays efficiently.
- 4. Hash maps for frequency counting.
- 5. Bucket sort for frequency-based grouping.
- 6. Optimized O(n) approach instead of sorting  $(O(n \log n))$ .