Experiment-5

Student Name: Chaitanya Gaba **UID:** 22BCS15984
Branch: BE-CSE **Section/Group:** KPIT-901/B
Semester: 6th **Date of Performance:** 18/01/25
Subject Name: Advanced Programming Lab - 2 **Subject Code:** 22CSP-351

1. **Aim:** Divide and Conquer
   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;
    int j = n - 1;
    int k = m + n - 1;

    while (j >= 0)
      if (i >= 0 && nums1[i] > nums2[j])
        nums1[k--] = nums1[i--];
      else
```

```
        nums1[k--] = nums2[j--];
  }
};
```

2.)

```
struct T {
    int num;
    int freq;
  };

  class Solution {
   public:
    vector<int> topKFrequent(vector<int>& nums, int k) {
      const int n = nums.size();
      vector<int> ans;
      unordered_map<int, int> count;
      auto compare = [](const T& a, const T& b) { return a.freq > b.freq; };
      priority_queue<T, vector<T>, decltype(compare)> minHeap(compare);

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

      for (const auto& [num, freq] : count) {
        minHeap.emplace(num, freq);
```

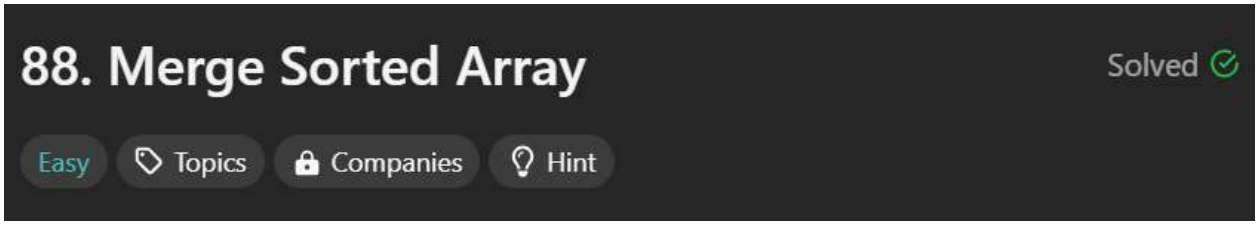
```
        if (minHeap.size() > k)
          minHeap.pop();
      }

      while (!minHeap.empty())
        ans.push_back(minHeap.top().num), minHeap.pop();

      return ans;
    }
  };
```

4. Output:

1.



Accepted 59 / 59 testcases passed

Chaitanya Gaba submitted at Mar 06, 2025 19:54

Editorial | Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

	Status	Language	Runtime	Memory	Notes
3	Accepted 3 minutes ago	C++	40 ms	61.6 MB	
2	Accepted Feb 27, 2025	C++	39 ms	61.5 MB	
1	Accepted Feb 27, 2025	C++	39 ms	61.5 MB	

2.

347. Top K Frequent Elements

Solved

Medium | Topics | Companies

Accepted 21 / 21 testcases passed

Chaitanya Gaba submitted at Mar 06, 2025 19:57

Editorial | Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

	Status	Language	Runtime	Memory	Notes
3	Accepted 3 minutes ago	C++	40 ms	61.6 MB	
2	Accepted Feb 27, 2025	C++	39 ms	61.5 MB	
1	Accepted Feb 27, 2025	C++	39 ms	61.5 MB	

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)).