Experiment 6.3

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 : Top K Frequent Elements

Problem statement: Given an integer array nums and an integer k, return *the* k *most frequent elements*. You may return the answer in any order.

Example 1:
Input: nums = [1,1,1,2,2,3], k = 2
Output: [1,2]
Example 2:
Input: nums = [1], k = 1
Output: [1]

Algorithm:

1. **Count the Frequency of Each Element**:
   - Create an **unordered_map** to store the frequency of each element in the array nums.
   - Iterate through each element in nums and update its frequency in the map.
2. **Use a Min-Heap to Store the Top K Frequent Elements**:
   - Create a **min-heap** (priority queue) that stores pairs of (frequency, element). The heap will help us efficiently track the top k frequent elements.
   - The heap is implemented as a **min-heap** using greater<pair<int, int>> so that the element with the smallest frequency is at the top.
3. **Push Elements into the Min-Heap**:
   - For each element and its frequency in the frequency map:
     - Push the pair (frequency, element) into the min-heap.
     - If the size of the heap exceeds k, pop the element with the smallest frequency. This ensures that the heap only stores the k most frequent elements.
4. **Extract the K Most Frequent Elements**:
   - Create a **result** vector to store the top k frequent elements.
   - While the heap is not empty, pop the top element (which contains the most frequent elements) and push the element (second value of the pair) into the result vector.
5. **Reverse the Result**:
   - Since the heap stores elements in increasing order of frequency, the result vector will have the elements in reverse order of their frequencies.

- Reverse the result vector to place the most frequent elements first.

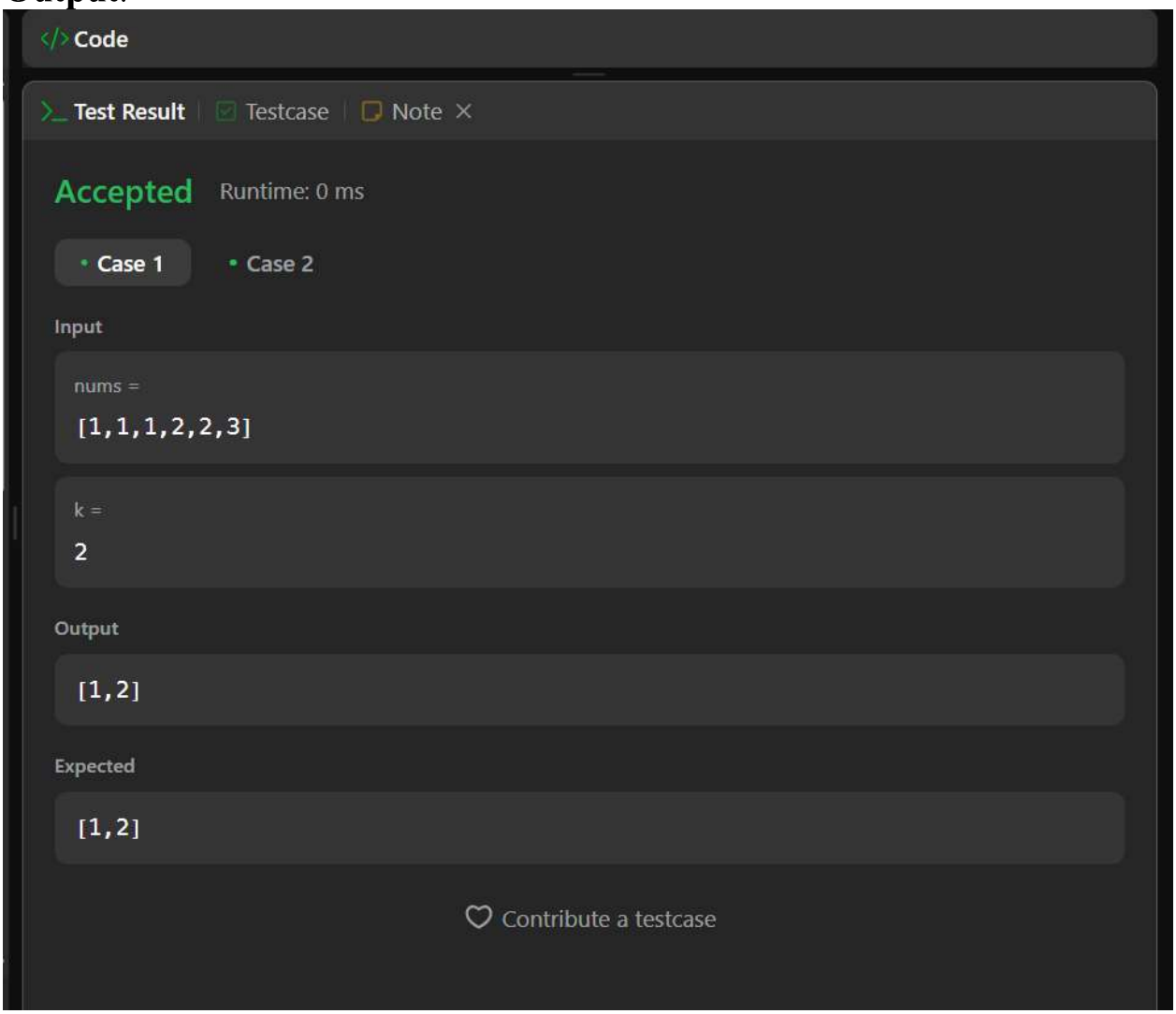
6. **Return the Result**:
   - Return the result vector containing the top k most frequent elements.

Code:

```
#include <iostream>
#include <vector>
#include <unordered_map>
#include <queue>
using namespace std;

class Solution {
public:
    vector<int> topKFrequent(vector<int>& nums, int k) {
        // Step 1: Count the frequency of each element in the array
        unordered_map<int, int> frequencyMap;
        for (int num : nums) {
            frequencyMap[num]++;
        }

        // Step 2: Use a min-heap to store the top k frequent elements
        // Min-heap stores pairs of (frequency, element)
        priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>>
      minHeap;

        // Step 3: Push the elements into the heap
        for (auto& entry : frequencyMap) {
            minHeap.push({entry.second, entry.first});

            // If the heap size exceeds k, remove the element with the smallest frequency
            if (minHeap.size() > k) {
                minHeap.pop();
            }
        }

        // Step 4: Extract the k most frequent elements from the heap
        vector<int> result;
        while (!minHeap.empty()) {
            result.push_back(minHeap.top().second);
            minHeap.pop();
        }
```

```
        // Since the heap stores elements in increasing order of frequency, reverse the result
        reverse(result.begin(), result.end());
        return result;
    }
};
```

Output:



Code

Test Result | Testcase | Note

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =
[1]

k =
1

Output

[1]

Expected

[1]

Contribute a testcase