

Experiment 6.3

Student Name: Amit Kumar

Branch: CSE

Semester: 6th

Subject: AP

UID: 22BCS16121

Section: 641/A

DOP: 25/02/2025

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:

- 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.
- 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.
- 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.
- 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.
- 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();
        }
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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




Output:





The screenshot displays a code editor interface with a dark theme. At the top, there's a tab labeled '</> Code'. Below it, a 'Test Result' section shows a green checkmark and the word 'Accepted' in green, followed by 'Runtime: 0 ms'. There are two tabs for test cases: 'Case 1' (selected) and 'Case 2'. Under the 'Input' section, there are two input fields: 'nums =' with the value '[1,1,1,2,2,3]' and 'k =' with the value '2'. The 'Output' section shows the result '[1,2]'. The 'Expected' section also shows '[1,2]'. At the bottom, there is a link with a heart icon that says 'Contribute a testcase'.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

 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](#)