📘 LeetCode 347 – Top K Frequent Elements

🔧 Goal:

> Given an integer array arr, return the k most frequent elements.

🧠 Approach Used:

🔶 HashMap (for frequency) + Min-Heap (for top K)

✅ Step-by-Step Logic :

1. **Count Frequency:**
    - Use `unordered_map<int, int>` to store frequency of each element.
    - Key = number, Value = count.
2. **Use Min-Heap:**
    - Make a `min-heap` of pairs `{frequency, value}` using `priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>>>`.
    - Keep heap size ≤ `k`. So, if size > `k`, pop the top (least frequent).
3. **Build Answer:**
    - Remaining elements in heap will be `k` most frequent.
    - Pop them one by one, store their `value` in a vector.
4. **Return the vector.**

🔄 Code Implementation:

```
class Solution {
public:
    vector<int> topKFrequent(vector<int>& arr, int k) {
        typedef pair<int, int> p;
        unordered_map<int, int> mp;

        // Step 1: Count frequency
        for (int ele : arr) {
            mp[ele]++;
        }

        // Step 2: Min-heap to track top K frequent
        priority_queue<p, vector<p>, greater<p>> pq;

        for (auto x : mp) {
            int value = x.first;
            int frequency = x.second;
            pq.push({frequency, value});
            if (pq.size() > k) {
                pq.pop(); // Remove least frequent
            }
        }

        // Step 3: Collect result
        vector<int> ans;
        while (pq.size()>0) {
            ans.push_back(pq.top().second);
            pq.pop();
        }

        return ans;
    }
};
```

🧮 Time & Space Complexity:

Aspect	Value	Reason
Time Complexity	`O(N log K)`	N = size of array, log K for heap ops
Space Complexity	`O(N + K)`	Map for frequency, heap of K elements

🧾 Summary Table:

Step	Tool Used	Purpose
Count frequency	`unordered_map`	Store count of each element
Keep top `k`	Min-heap	Always keep `k` highest frequencies
Final answer	Vector	Push values from heap to result array

💡 Intuition:

> "Zyada frequency wale elements chahiye?
>
> Map me gin le, heap me sambhal le, aur `top k` chhant le."

🧠 Interview Tip:

- Agar **"top K"** ya **"most frequent"** dikhe — turant **hashmap + heap** sochna.
- Heap size `K` ka rakho — time efficient hoga.