347. Top K Frequent Elements

09 April 2022 05:41 PM

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]
```

1, 1, 1, 2, 2, 3, 4, 4, 4, 4

K most frequency elements

↓

commonly occuring

(count)

↳ HashMap

1 → 3

2 → 2

3 → 1

4 → 4

To store count of each element we use HashMap (key-value pair)

key :	(count) value
1 :	3
2 :	2
3 :	1
4 :	4

1st Approach :

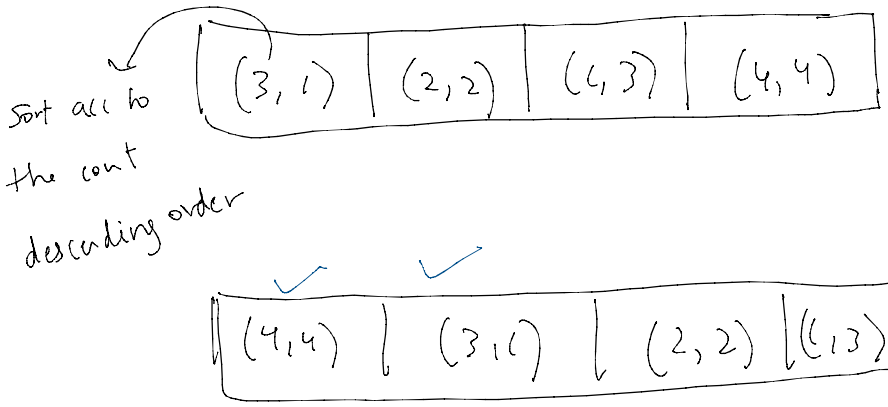
pair

1st Approach :

Sort

pair

(count, element)



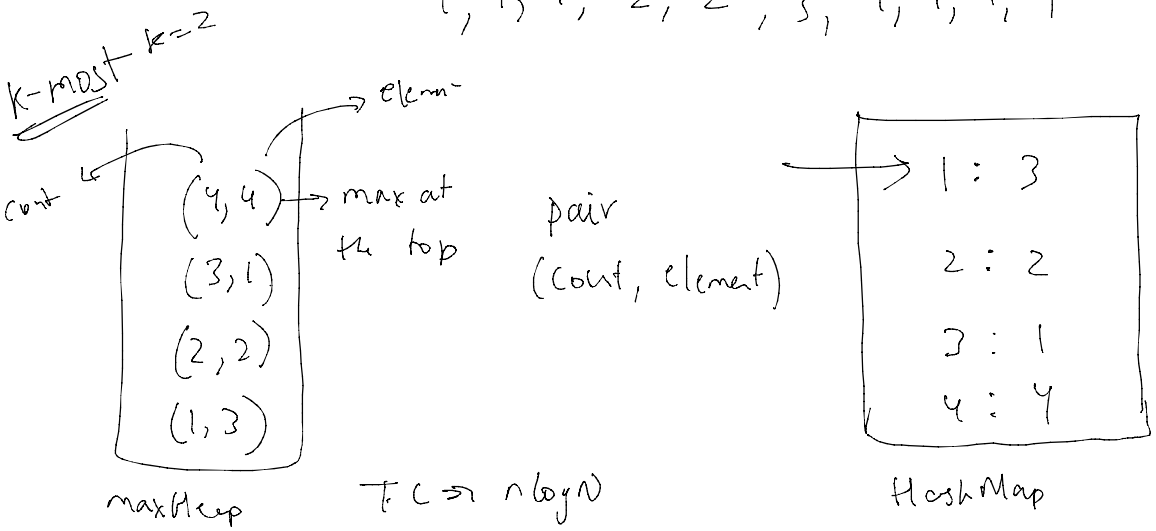
T.C ⇒ (n log n)

2nd Approach : Use Heap ⇒ automatically get sorted (miniHeap, maxHeap)

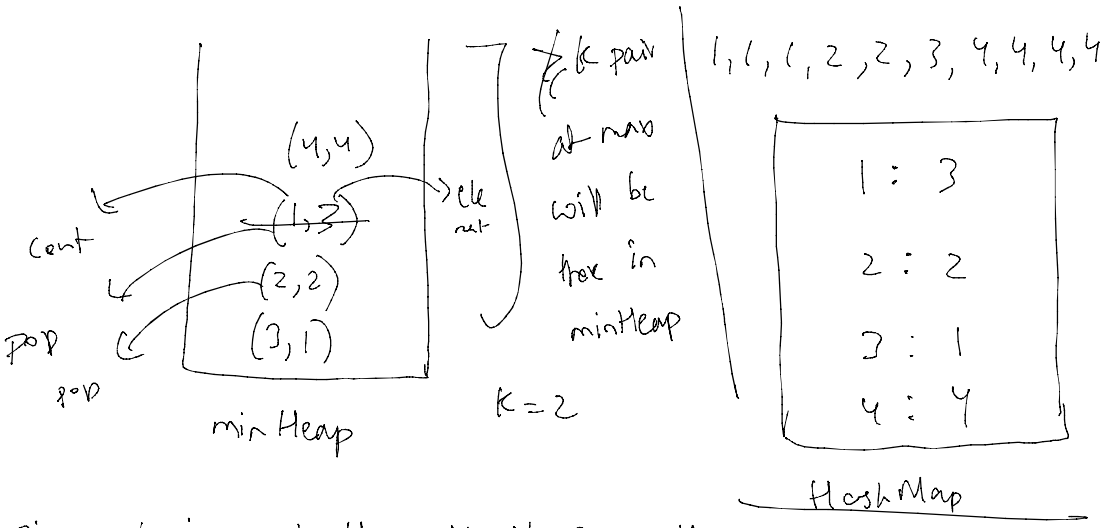
nlogk nlogN

max Heap :

1, 1, 1, 2, 2, 3, 4, 4, 4, 4



min Heap :



Size $\leq k$ in min Heap if it more then pop, we pop smaller count.

$1 < 2$ ✓

$2 \leq 2$ ✓

$3 < 2$ ✗ → pop

popped ⇒ (1,3), (1,2)

Ans will be in minHeap

We Choose minHeap when we need to find largest, most frequent

maxHeap when we need to find smallest

TC ⇒ $O(n \log k)$

```
class Solution {
public:
    vector<int> topKFrequent(vector<int>& nums, int k) {
        unordered_map<int, int> counts;
        priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int,
int>>> min_heap;
        for(auto i: nums){
            counts[i]++;
        }

        for(auto &i: counts){
            min_heap.push({i.second, i.first});
            if(min_heap.size() > k){
                min_heap.pop();
            }
        }
        vector<int> res;
        while(k--){
            res.push_back(min_heap.top().second);
            min_heap.pop();
        }
        return res;
    }
};
```