Top K/Kth Frequent Coding Pattern

LeetCode 215 - Kth Largest Element in an Array *[medium]*

Find the **k**th largest element in an unsorted array. Note that it is the **k**th largest element in the sorted order, not the **k**th distinct element.

Example 1:

```
Input: [3, 2, 1, 5, 6, 4] and k = 2
Output: 5
```

Example 2:

```
Input: [3, 2, 3, 1, 2, 4, 5, 5, 6] and k = 4
Output: 4
```

Note:

You may assume **k** is always valid, 1 ≤ **k** ≤ array's length.

```java
public class TopKFrequent {
    //top k positon value print
    public int[] topKFrequent(int[] nums, int k){
        Map<Integer, Integer> map = new HashMap<>();
        for (int i : nums){
           map.put(i, map.getOrDefault(i, 0) + 1);
        }
        PriorityQueue<Map.Entry<Integer, Integer>> pq = new PriorityQueue<>((a, b)
-> b.getValue() - a.getValue());
        for (Map.Entry entry: map.entrySet()){
            pq.add(entry);
        }
        int[] result = new int[k];
        for (int i = 0; i < k; i++){
            result[i] = pq.poll().getKey();
        }
       return result;
    }
    //only kth positon value print
    public int topKthFrequent(int[] nums, int k){
        Map<Integer, Integer> map = new HashMap<>();
        for (int i : nums){
            map.put(i, map.getOrDefault(i, 0) + 1);
        }
        PriorityQueue<Map.Entry<Integer, Integer>> pq = new PriorityQueue<>((a, b)
-> b.getValue() - a.getValue());
        for (Map.Entry entry: map.entrySet()){
            pq.add(entry);
        }
        for (int i = 0; i < k; i++){
            pq.poll().getKey();
        }
        return pq.poll().getKey();
    }

    public static void main(String[] args) {
        TopKFrequent k = new TopKFrequent();
        int[] a = {1, 1, 1, 2, 2, 3};
        System.out.println(Arrays.toString(k.topKFrequent(a, 2)));

        int[] b = {3, 2, 1, 5, 6, 4};
        System.out.println(k.topKthFrequent(b, 2));
    }
}
```

LeetCode 973 - K Closest Points to Origin [*medium*]