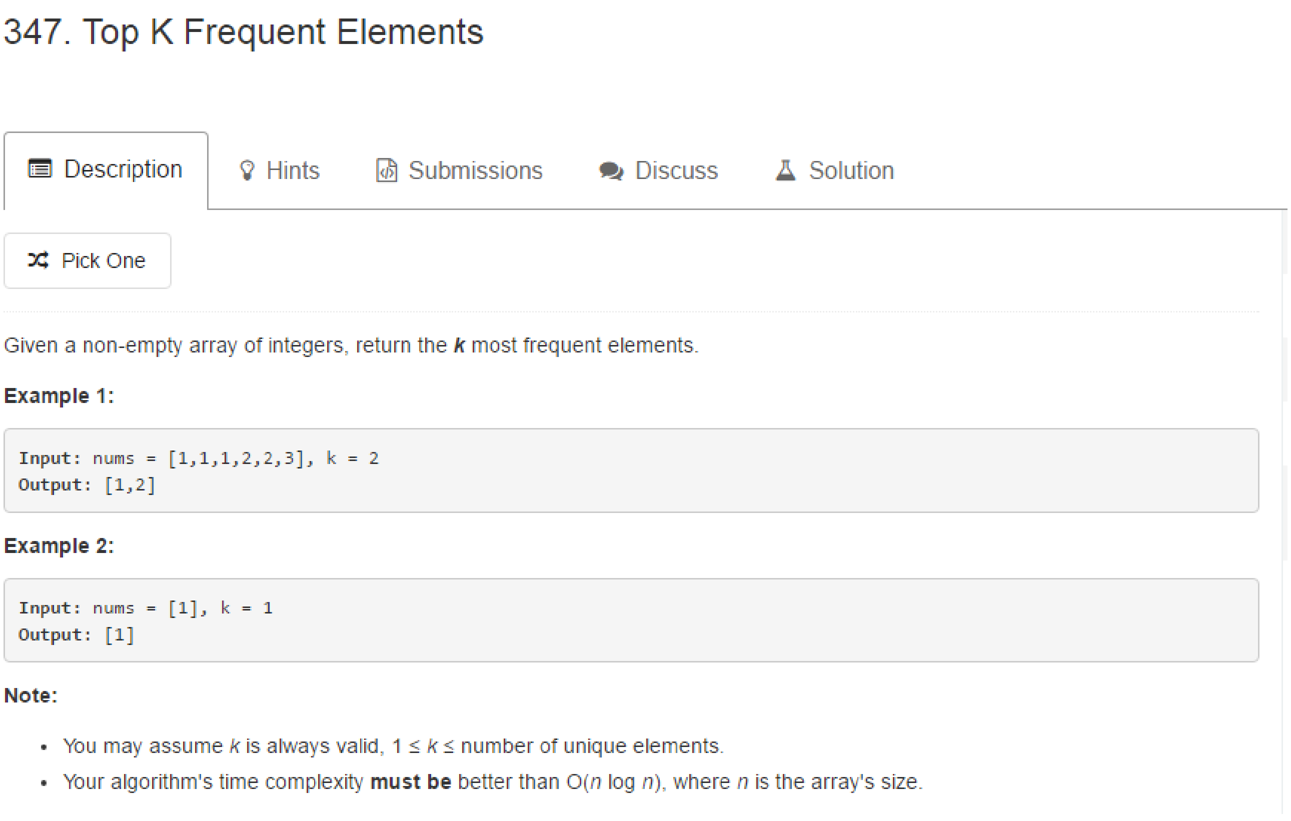


# 347. Top K Frequent Elements

Given a non-empty array of integers, return the ***k*** most frequent elements.

**Example 1:**

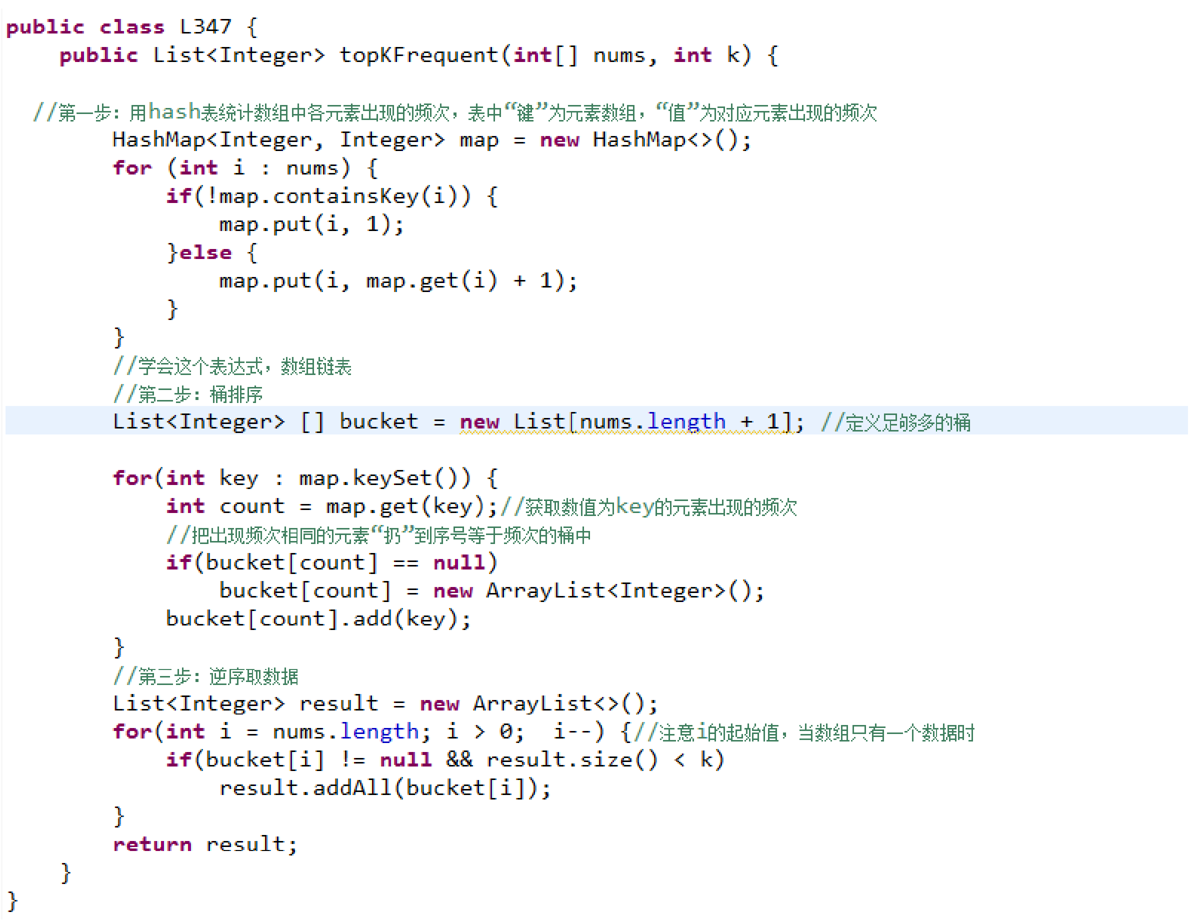
```
Input: nums = [1,1,1,2,2,3], k = 2
Output: [1,2]
```

**Example 2:**

```
Input: nums = [1], k = 1
Output: [1]
```

**Note:**

- You may assume *k* is always valid, 1 ≤ *k* ≤ number of unique elements.
- Your algorithm's time complexity **must be** better than O(*n* log *n*), where *n* is the array's size.



```
public class L347 {
    public List<Integer> topKFrequent(int[] nums, int k) {

  //第一步：用hash表统计数组中各元素出现的频次，表中“键”为元素数组，“值”为对应元素出现的频次
        HashMap<Integer, Integer> map = new HashMap<>();
        for (int i : nums) {
            if(!map.containsKey(i)) {
                map.put(i, 1);
            }else {
                map.put(i, map.get(i) + 1);
            }
        }
        //学会这个表达式，数组链表
        //第二步：桶排序
        List<Integer> [] bucket = new List[nums.length + 1]; //定义足够多的桶

        for(int key : map.keySet()) {
            int count = map.get(key);//获取数值为key的元素出现的频次
            //把出现频次相同的元素“扔”到序号等于频次的桶中
            if(bucket[count] == null)
                bucket[count] = new ArrayList<Integer>();
            bucket[count].add(key);
        }
        //第三步：逆序取数据
        List<Integer> result = new ArrayList<>();
        for(int i = nums.length; i > 0;  i--) {//注意i的起始值，当数组只有一个数据时
            if(bucket[i] != null && result.size() < k)
                result.addAll(bucket[i]);
        }
        return result;
    }
}
```