# 描述

给定一个非空的整数数组，返回其中出现频率前 k 高的元素。

**示例 1:**

输入: nums = [1,1,1,2,2,3], k = 2

输出: [1,2]

**示例 2:**

输入: nums = [1], k = 1

输出: [1]

**说明：**

你可以假设给定的 k 总是合理的，且 1 ≤ k ≤ 数组中不相同的元素的个数。

你的算法的时间复杂度必须优于 O(n log n) , n 是数组的大小。

# 分析

方法 1：堆

k = 1 时问题很简单，线性时间内就可以解决。只需要用哈希表维护元素出现频率，每一步更新最高频元素即可。

当 k > 1 就需要一个能够根据出现频率快速获取元素的数据结构，这就是优先队列。

首先建立一个元素值对应出现频率的哈希表。在 Java 中使用 HashMap，但需要手工填值。在 Python 中提供一个字典结构用作哈希表和在 collections 库中的 Counter 方法去构建我们需要的哈希表。

这个步骤需要 O(N)O(N) 时间其中 NN 是列表中元素个数。

第二步建立堆，堆中添加一个元素的复杂度是 O(\log(k))O(log(k))，要进行 NN 次复杂度是 O(N)O(N)。

最后一步是输出结果，复杂度为 O(k\log(k))O(klog(k))。

在 Python 中可以使用 heapq 库中的 nlargest 方法，可以在相同时间内完成，但只需要一行代码解决。

# 代码