# 40. 最小的 K 个数

## 题目链接

[牛客网](https://www.nowcoder.com/practice/6a296eb82cf844ca8539b57c23e6e9bf?tpId=13&tqId=11182&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking&from=cyc_github)

## 解题思路

### 大小为 K 的最小堆

* 复杂度：O(NlogK) + O(K)
* 特别适合处理海量数据

维护一个大小为 K 的最小堆过程如下：使用大顶堆。在添加一个元素之后，如果大顶堆的大小大于 K，那么将大顶堆的堆顶元素去除，也就是将当前堆中值最大的元素去除，从而使得留在堆中的元素都比被去除的元素来得小。

应该使用大顶堆来维护最小堆，而不能直接创建一个小顶堆并设置一个大小，企图让小顶堆中的元素都是最小元素。

Java 的 PriorityQueue 实现了堆的能力，PriorityQueue 默认是小顶堆，可以在在初始化时使用 Lambda 表达式 (o1, o2) -> o2 - o1 来实现大顶堆。其它语言也有类似的堆数据结构。

public ArrayList<Integer> GetLeastNumbers\_Solution(int[] nums, int k) {
 if (k > nums.length || k <= 0)
 return new ArrayList<>();
 PriorityQueue<Integer> maxHeap = new PriorityQueue<>((o1, o2) -> o2 - o1);
 for (int num : nums) {
 maxHeap.add(num);
 if (maxHeap.size() > k)
 maxHeap.poll();
 }
 return new ArrayList<>(maxHeap);
}

### 快速选择

* 复杂度：O(N) + O(1)
* 只有当允许修改数组元素时才可以使用

快速排序的 partition() 方法，会返回一个整数 j 使得 a[l..j-1] 小于等于 a[j]，且 a[j+1..h] 大于等于 a[j]，此时 a[j] 就是数组的第 j 大元素。可以利用这个特性找出数组的第 K 个元素，这种找第 K 个元素的算法称为快速选择算法。

public ArrayList<Integer> GetLeastNumbers\_Solution(int[] nums, int k) {
 ArrayList<Integer> ret = new ArrayList<>();
 if (k > nums.length || k <= 0)
 return ret;
 findKthSmallest(nums, k - 1);
 /\* findKthSmallest 会改变数组，使得前 k 个数都是最小的 k 个数 \*/
 for (int i = 0; i < k; i++)
 ret.add(nums[i]);
 return ret;
}

public void findKthSmallest(int[] nums, int k) {
 int l = 0, h = nums.length - 1;
 while (l < h) {
 int j = partition(nums, l, h);
 if (j == k)
 break;
 if (j > k)
 h = j - 1;
 else
 l = j + 1;
 }
}

private int partition(int[] nums, int l, int h) {
 int p = nums[l]; /\* 切分元素 \*/
 int i = l, j = h + 1;
 while (true) {
 while (i != h && nums[++i] < p) ;
 while (j != l && nums[--j] > p) ;
 if (i >= j)
 break;
 swap(nums, i, j);
 }
 swap(nums, l, j);
 return j;
}

private void swap(int[] nums, int i, int j) {
 int t = nums[i];
 nums[i] = nums[j];
 nums[j] = t;
}