# 最小的K个数

## 题目

[牛客网](https://www.nowcoder.com/practice/6a296eb82cf844ca8539b57c23e6e9bf?tpId=13&tqId=11182&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking&tPage=2)

输入n个整数，找出其中最小的K个数。例如输入4,5,1,6,2,7,3,8这8个数字，则最小的4个数字是1,2,3,4,。

## 解题思路

### Partition

该算法基于 Partition

public ArrayList<Integer> GetLeastNumbers\_Solution\_Partition(int[] input, int k) {  
 ArrayList<Integer> res = new ArrayList<>();  
  
 if (k > input.length || k < 1) {  
 return res;  
 }  
  
 int start = 0, end = input.length - 1;  
 int index = partition(input, start, end);  
 while (index != k - 1) {  
 if (index > k - 1) {  
 end = index - 1;  
 index = partition(input, start, end);  
 } else {  
 start = index + 1;  
 index = partition(input, start, end);  
 }  
 }  
  
 for (int i = 0; i < input.length && i < k; i++) {  
 res.add(input[i]);  
 }  
 return res;  
}  
  
private int partition(int[] nums, int start, int end) {  
 int left = start, right = end;  
 int key = nums[left];  
  
 while (left < right) {  
 while (left < right && nums[right] > key) {  
 right--;  
 }  
 if (left < right) {  
 nums[left] = nums[right];  
 left++;  
 }  
  
 while (left < right && nums[left] <= key) {  
 left++;  
 }  
 if (left < right) {  
 nums[right] = nums[left];  
 right++;  
 }  
 }  
  
 nums[left] = key;  
  
 return left;  
}

### 小根堆算法

该算法基于小根堆，适合海量数据，时间复杂度为：n\*logk

public ArrayList<Integer> GetLeastNumbers\_Solution(int[] input, int k) {  
 ArrayList<Integer> res = new ArrayList<>();  
 if (k > input.length||k==0) {  
 return res;  
 }  
  
 for (int i = input.length - 1; i >= 0; i--) {  
 minHeap(input, 0, i);  
  
 swap(input, 0, i);  
  
 res.add(input[i]);  
 if (res.size() == k) break;  
 }  
 return res;  
}  
  
private void minHeap(int[] heap, int start, int end) {  
 if (start == end) {  
 return;  
 }  
  
 int childLeft = start \* 2 + 1;  
 int childRight = childLeft + 1;  
  
 if (childLeft <= end) {  
 minHeap(heap, childLeft, end);  
  
 if (heap[childLeft] < heap[start]) {  
 swap(heap, start, childLeft);  
 }  
 }  
  
 if (childRight <= end) {  
 minHeap(heap, childRight, end);  
  
 if (heap[childRight] < heap[start]) {  
 swap(heap, start, childRight);  
 }  
 }  
}  
  
private void swap(int[] nums, int a, int b) {  
 int t = nums[a];  
 nums[a] = nums[b];  
 nums[b] = t;  
}