# 题目

在未排序的数组中找到第 k 个最大的元素。请注意，你需要找的是数组排序后的第 k 个最大的元素，而不是第 k 个不同的元素。

**示例 1:**

输入: [3,2,1,5,6,4] 和 k = 2

输出: 5

**示例 2:**

输入: [3,2,3,1,2,4,5,5,6] 和 k = 4

输出: 4

**说明:**

你可以假设 k 总是有效的，且 1 ≤ k ≤ 数组的长度。

# 分析

## 方法一：排序

可以采用快排，然后取第k个数值。

## 方法二：堆/优先队列

代码（参考LeetCode703+剑指Offer II59）：

class Solution {

public:

int findKthLargest(vector<int>& nums, int k) {

priority\_queue<int,vector<int>,greater<int>> pri\_que;

for(int i=0;i<nums.size();i++)

{

pri\_que.push(nums.at(i));

if(pri\_que.size()>k)

{

pri\_que.pop();

}

}

return pri\_que.top();

}

};

或：

class Solution {

public:

int findKthLargest(vector<int>& nums, int k) {

priority\_queue<int, vector<int>, greater<int>> pri\_que;

for (int i = 0; i < nums.size(); i++) {

pri\_que.push(nums[i]);

if (i >= k) { // 错误：i>k

pri\_que.pop();

}

}

return pri\_que.top();

}

};

或：

class Solution {

public:

int findKthLargest(vector<int>& nums, int k) {

std::priority\_queue<int,std::vector<int>,std::greater<int> > max\_queue;

for(size\_t i=0;i<nums.size();i++){

if(max\_queue.size() < k){

max\_queue.push(nums.at(i));

}else{

if(nums.at(i) > max\_queue.top()){

max\_queue.pop();

max\_queue.push(nums.at(i));

}

}

}

return max\_queue.top();

}

};