1. Kth Largest Element in an Array

Find the **k**th largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.

**Example 1:**

Input: [3,2,1,5,6,4] and k = 2  
Output: 5

**Example 2:**

Input: [3,2,3,1,2,4,5,5,6] and k = 4  
Output: 4

**Note:** You may assume k is always valid, 1 ≤ k ≤ array’s length.

**解法1** 直接调用sort()函数，返回第k大的数字

class Solution {  
public:  
 int findKthLargest(vector<int>& nums, int k) {  
 sort(nums.begin(), nums.end());  
 return nums[nums.size() - k];  
 }  
}

**解法2** 用quick\_sort的思路

**解法2.1** 自己写partition函数，为了避免1 vs n-1的划分导致的性能下降，可以采用random\_partition

class Solution {  
public:  
 int findKthLargest(vector<int>& nums, int k) {  
 return search(nums, k, 0, nums.size() - 1);  
 }   
 int search(vector<int>& nums, int k, int l, int r){  
 int order = random\_partition(nums, l, r);  
 if(order == k)return nums[l+order-1];  
 else if(order < k){  
 return search(nums, k - order, l+order, r);  
 }else{  
 return search(nums, k, l, l + order - 2);  
 }  
 }  
 int random\_partition(vector<int>& nums, int l, int r){  
 srand((unsigned)time(NULL));  
 int idx = rand() % (r-l+1)+ l;  
 swap(nums[l], nums[idx]);  
 return partition(nums, l, r);  
 }  
 int partition(vector<int>& nums, int l, int r){  
 int pivot = nums[l];  
 int i = l, j = r;  
 while(i < j){  
 while(nums[j] < pivot && j > i)j--; // 注意不要丢掉i < j的条件  
 nums[i] = nums[j];  
 while(nums[i] >= pivot && i < j)i++; // 注意不要丢掉i < j的条件  
 nums[j] = nums[i];  
 }  
 nums[i] = pivot;  
 return i - l + 1;  
 }  
};

**解法2.2** 调用stl中的partition函数。原型：

iterator partition(nums.begin(), nums.end(), cond)，其中cond是一个函数，满足cond条件的元素会被放到前一段，不满足的放到后一段

static int pivot;  
 static bool cmp(int x){  
 if(x >= pivot)return true;  
 else return false;  
 }  
   
 int random\_partition(vector<int>& nums, int l, int r){  
 srand((unsigned)time(NULL));  
 int idx = rand() % (r-l+1)+ l;  
 swap(nums[l], nums[idx]);  
 pivot = nums[l];  
 auto it = partition(nums.begin() + l, nums.begin() + r + 1, cmp);  
 return it - nums.begin();  
 }