/\*

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

For example,

Given [3,2,1,5,6,4] and k = 2, return 5.

Note:

You may assume k is always valid, 1 ≤ k ≤ array's length.

way-1:排序……

way-2:利用快排的partition函数思想，选定一个数组内的值作为pivot，将小于pivot的数字放到pivot右边，大于等于pivot的数字放到pivot左边。接着判断两边数字的数量，如果左边的数量小于k个，说明第k大的数字存在于pivot及pivot右边的区域之内，对右半区执行partition函数；如果右边的数量小于k个，说明第k大的数字在pivot和pivot左边的区域之内，对左半区执行partition函数。直到左半区刚好有k-1个数，那么第k大的数就已经找到了

\*/

class Solution {

public:

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

{

//way-1

/\*

sort(nums.begin(),nums.end());

return nums[nums.size()-k];

\*/

//way-2

/\*

int high = nums.size();

int low = 0;

while (low < high) {

int i = low;

int j = high-1;

int pivot = nums[low];

while (i <= j) {

while (i <= j && nums[i] >= pivot)

i++;

while (i <= j && nums[j] < pivot)

j--;

if (i < j)

swap(nums[i++],nums[j--]);

}

swap(nums[low],nums[j]);

if (j == k-1)

return nums[j];

else if (j < k-1)

low = j+1;

else

high = j;

}

\*/

}

};