**描述**

给定一个未排序的整数数组，找到其中位数。

中位数是排序后数组的中间值，如果数组的个数是偶数个，则返回排序后数组的第N/2个数。

**样例 1:**

输入：[4, 5, 1, 2, 3]

输出：3

解释：

经过排序，得到数组[1,2,3,4,5]，中间数字为3

**样例 2:**

输入：[7, 9, 4, 5]

输出：5

解释：

经过排序，得到数组[4,5,7,9]，第二个(4/2)数字为5

**挑战**

时间复杂度为O(n)

时间复杂度位O（n）这就没啥好说的了，我就直接上手用快排了。排完返回中间值就可以了。

void exchange(vector<int>& a, const int& i, const int& j) {

int tmp = a[i];

a[i] = a[j];

a[j] = tmp;

}

int partition(vector<int>& a,const int &lo,const int &hi) {

int i = lo, j = hi + 1;

int val = a[lo];

while (true) {

while (a[++i] < val)

if (i == hi)

break;

while (val < a[--j])

if (j == lo)

break;

if (i >= j)

break;

exchange(a, i, j);

}

exchange(a, lo, j);

return j;

}

void quicksort(vector<int>& a,const int&lo,const int&hi) {

if (lo >= hi)

return;

int j = partition(a, lo, hi);

quicksort(a, lo, j - 1);

quicksort(a, j + 1, hi);

}

void sortofquick(vector<int>& a) {

int n = a.size() - 1;

quicksort(a, 0, n);

}

int median(vector<int> &nums) {

// write your code here

sortofquick(nums);

int j=nums.size();

int a=0;

if(j%2==0)

a=j/2-1;

else

a=j/2;

return nums[a];

}

截图如下：

