**题目描述：数组中有一个数字出现的次数超过数组长度的一半，请找出这个数字。**

**例如输入一个长度为9的数组{1,2,3,2,2,2,5,4,2}。由于数字2在数组中出现了5次，超过数组长度的一半，因此输出2。如果不存在则输出0。**

代码(mine):

class Solution {

public:

int MoreThanHalfNum\_Solution(vector<int> numbers) {

int i, mark ,count;

int length = numbers.size();

if (length == 0)return 0;

if (length == 1)return numbers[0];

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

mark = numbers[0];

count = 1;

for (i = 1; i < length; i++){

if (mark == numbers[i]){

count++;

}

else{

if (count>length / 2)return mark;

mark = numbers[i];

count = 1;

}

}

return 0;

}

};

**主要思想:先排序，然后再逐个统计。算法时间复杂度较大，至少为[N\*logN+N=(logN+1)\*N].**

代码(2):

class Solution {

public:

int MoreThanHalfNum\_Solution(vector<int> numbers) {

int len=numbers.size();

int num=numbers [0];

if(len==0) return 0;

int count=1;

//假定存在这样的数并且采用"抵消法"

for (int i=1; i<len;i++)

{

if(num==numbers[i]){count++;}

else count--;

if (count==0){num=numbers[i]; count=1 ;}

}

//逐个数累加

count=0;

for (int i=0; i<len; i++) {

if(num==numbers[i]) count++;

}

if (2\*count>len) return num;

else return 0;

}

};

**分析：算法时间复杂度为N.**

代码(3):

class Solution {

public:

int MoreThanHalfNum\_Solution(vector<int> numbers) {

if (numbers.empty()) return 0;

map<int, int> map;

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

map[numbers[i]]++;

}

cout << map.size()<<endl;

cout << map[12] << endl;

int len = (numbers.size() >> 1);

for (auto &it : map){

if (it.second > len){

return it.first;

}

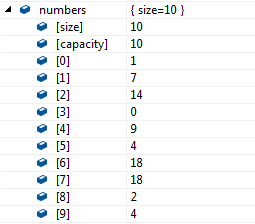
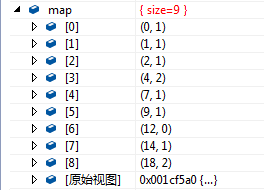
}

return 0;

}

};

分析:**map容器以first的值作下标**

如果访问不存在的下标值则新建一个(按键值从小到大排序)并将新元素插入其中(初值置为0),如果存在匹配的下标元素则将其值加1。访问只能以迭代器或是map[12](=0)的形式进行。