user: sakura

第17题——数组中出现次数超过一半的数字

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。

你可以假设数组是非空的，并且给定的数组总是存在多数元素。

示例 1:

```
输入: [1, 2, 3, 2, 2, 2, 5, 4, 2]
```

```
输出: 2
```

限制：1 <= 数组长度 <= 50000

哈希（1）

理解题意

```
这道题，这一看，不就是比较出现次数嘛，我第一反应就是哈希表，因为C语言并没有像C++那样封装好的容器
```

```c
int majorityElement(int* nums, int numsSize){
    int i;
    int pos[10000]={0};//正数数组
    int neg[10000]={0};//负数数组
    int mid=numsSize/2;
    for(i=0;i<numsSize;i++)
    {
        if(nums[i]>=0){
            pos[nums[i]]++;//哈希表对应下标元素+1
            if(pos[nums[i]]>mid)   //大于numsSize/2返回
            {
                return nums[i];
            }
        }
        else if(nums[i]<0){
            neg[-nums[i]]++;//哈希表对应下标元素+1
            if(neg[-nums[i]]>mid)//大于numsSize/2返回
            {
                return nums[i];
            }
        }
    }
    return -1;
}
```

这个代码，只要输入超过我分配空间的数值，就无法为对应的哈希表+1，难搞。

我使用了一个相当蠢的办法，但问题确实解决了

```
int majorityElement(int* nums, int numsSize){
    int i;
    int hash[20000]={0};
    //考虑正负，>=0存在10000以后,负数存在10000以前
    int max=2147483647;//定义最大值
    int mid=numsSize/2;
    for(i=0;i<numsSize;i++)
    {
        hash[nums[i]%max+10000]++;//哈希表对应下标元素+1
        if(hash[nums[i]%max+10000]>mid)   //大于numsSize/2返回
        {
            return nums[i];
        }
    }
    return -1;
}
```

利用取余的特性，完成了这道题，但这真的很蠢，只是骗过了编译器，实际上仅仅只能处理2147483647而已，在10000-2147483647之间的数我们仍然无法处理，无奈，放弃哈希表的办法，另辟蹊径。

摩尔投票法（2）

核心理念为 票数正负抵消 。

那么，对于我们的这道题，有两个推论：

1. 假设众数的票数为 +1 非众数的票数为 -1，那么，众数的和一定 >0
2. 如果数组的前n个数字的票数和0，那么数组剩下的 （size-n）的票数和一定 >0

接下来我们依照这个推论来解题

假设数组为 [1 ,2 , 3 , 2 , 2 , 2 , 5 , 4 , 2] 票数count=0 i=0 众数 为x ;

第 1 次，我们假设第 i 个数就是众数 ， 那么，x=1,count=1 , i++;

```
接着和下一个数比较，1≠2，count=0 ， i++ ;
```

第 2 次，我们假设第 i 个数就是众数 ， 那么，x=3,count=1 , i++;

```
接着和下一个数比较，3≠2，count=0 ， i++ ;
```

第 3 次，我们假设第 i 个数就是众数 ， 那么，x=2,count=1 , i++;

```
接着和下一个数比较，2=2，count=2 ， i++;

接着和下一个数比较，2≠5，count=1 ， i++;

接着和下一个数比较，   2≠4，count=0 ， i++;
```

第 4 次，我们假设第 i 个数就是众数 ， 那么，x=2,count=1 , i++;

```
后面没有数了，那么返回这个数，也就是 nums[ i ] 即可。
```

代码实现

```
int majorityElement(int* nums, int numsSize){
    if(numsSize){
        int count=0;
        int x;
        for(int i=0;i<numsSize;i++){
            //若count==0，则把第i个数当作众数
            if(count==0){
                x=nums[i];
                count++;
            }
            //让x与下一个数比较，相等则票数count++
            else if(x==nums[i]){
                count++;
            }
            //不相等则票数count--
            else{
                count--;
            }
        }
        return x;
    }
    return -1;
}
```

可以看到，这个方法优雅且简洁。

排序法（3）

如果将数组 nums 中的所有元素按照单调递增或单调递减的顺序排序，那么下标为 [n/2]的元素一定是众数

所以我们可以先给数组排序，然后直接返回nums[n/2]即可。

提供比较函数，然后使用内置快速排序qsort排序，再返回nums[n/2]。

代码如下：

```
int cmp(void *a,void *b)
{
    return *(int*)a-*(int*)b;
}
int majorityElement(int* nums, int numsSize){
    qsort(nums,numsSize,sizeof(int),cmp);
    return nums[numsSize/2];
}
```