```
/*
给定一个大小为 n 的数组，找到其中的众数。众数是指在数组中出现次数大于 ⌊ n/2 ⌋ 的元素。

你可以假设数组是非空的，并且给定的数组总是存在众数。

示例 1:

输入: [3,2,3]
输出: 3

示例 2:

输入: [2,2,1,1,1,2,2]
输出: 2

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/majority-element
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 方法一:逐一遍历每个元素,统计之后与之相同的个数(vote),相同加1,不同减1,显然当vote的值大于等于0时,当前元素即众树
- 方法一:时间复杂度$O(n^2)$,果不其然超时了.
- 方法二：对方法一进行了优化，但是还是有硬伤，若测试用例逆序位1，2，1，2类似交换的也会超时。
- 方法三：先对数组进行快速排，由于原始数组的众数的个数一定过半，所以取数组的中间元素即可。（还是超时了）
- 方法四：优化快速排序算法，使用三路快排算法。

方法一：C, 投票筛选法

```
int majorityElement(int* nums, int numsSize)
{

    int i           = 0 ;
    int j           = 0 ;
    int ret_val     = 0 ;
    int vote        = 0 ;

    for(i = 0 ; i < numsSize  ; i++)
    {
        ret_val =   nums[i];
        vote    =   0;
        for(j=i;j<numsSize;j++)
        {
            if(nums[j]==ret_val)
```

```
            {
                vote++;
            }
            else
            {
                vote--;
            }
        }
        if(vote>=0)
        {
            break;
        }
    }
    return ret_val;
}

/*
执行结果：
超出时间限制
显示详情
最后执行的输入：
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1.
..
*/
```

```
int majorityElement(int* nums, int numsSize)
{

    int i       =   0       ;
    int j       =   0       ;
    int ret_val =   rand()  ;
    int vote    =   0       ;

    for(i = numsSize-1 ; i >=0  ; i--)
    {
        if(ret_val == nums[i])
        {
            continue;
        }
        ret_val = nums[i];
        vote = 0;
        for(j=i;j>=0;j--)
```

```
        {
            if(nums[j]==ret_val)
            {
                vote++;
            }
            else
            {
                vote--;
            }
        }
        if(vote>=0)
        {
            break;
        }
    }
    return ret_val;
}


/*
执行结果:
通过
显示详情
执行用时 :28 ms, 在所有 C 提交中击败了85.92% 的用户
内存消耗 :8.9 MB, 在所有 C 提交中击败了52.54%的用户
*/
```

方法三：先快排再取中间元素

```
void QuickSort(int* array, int start, int last)
{
    int i = start;
    int j = last;
    int temp = array[i];
    if (i < j)
    {
        while (i < j)
        {
            //
            while (i < j &&  array[j]>=temp )
                j--;
            if (i < j)
            {
                array[i] = array[j];
                i++;
            }

            while (i < j && temp > array[i])
                i++;
            if (i < j)
            {
                array[j] = array[i];
```

```
                j--;
            }

        }
        //把基准数放到i位置
        array[i] = temp;
        //递归方法
        QuickSort(array, start, i - 1);
        QuickSort(array, i + 1, last);
    }
}

int majorityElement(int* nums, int numsSize)
{


    int ret_val = 0;

    QuickSort(nums,0,numsSize-1);

    ret_val = nums[numsSize/2];

    return ret_val;
}


/*
执行结果:
超出时间限制
显示详情
最后执行的输入:
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1.
..
*/
```

方法四：C，三路优化快排+取中值

```
void swap(int*a , int* b)
{
    int c;
    c = *a;
    *a = *b;
```

```
    *b = c;
}

void __quicksort3(int* arr,int l,int r)
{

    int v   =   arr[l]  ;
    int lt  =   l       ;
    int gt  =   r+1     ;
    int i   =   l+1     ;

    if(l>=r)
    {
        return ;
    }

    while(i<gt)
    {
        if(arr[i]<v)
        {
            swap(&arr[i],&arr[lt+1]);
            lt++;
            i++;
        }
        else if(arr[i]>v)
        {
            swap(&arr[i],&arr[gt-1]);
            gt--;
        }
        else
        {
            i++;
        }
    }

    swap(&arr[l],&arr[lt]);
     __quicksort3(arr,l,lt-1);
     __quicksort3(arr,gt,r);


}



int majorityElement(int* nums, int numsSize)
{


    int ret_val = 0;

    __quicksort3(nums,0,numsSize-1);

    ret_val = nums[numsSize/2];
```

```
    return ret_val;
}


/*
执行结果:
通过
显示详情
执行用时 :28 ms, 在所有 C 提交中击败了85.92% 的用户
内存消耗 :9 MB, 在所有 C 提交中击败了26.37%的用户
*/
```

AlimyBreak
2019.08.11