数组中占比超过一半的元素称之为主要元素。给定一个整数数组，找到它的主要元素。若没有，返回-1。

示例 1：

输入：[1,2,5,9,5,9,5,5,5]

输出：5

示例 2：

输入：[3,2]

输出：-1

示例 3：

输入：[2,2,1,1,1,2,2]

输出：2

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/find-majority-element-lcci

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。

**思路:**

1). 先把数组中的所有数存入到set中

2). 依次使用set存储的值,与数组中的比较是否相同

|  |
| --- |
| public int majorityElement(int[] nums) {  //创建一个Set集合  HashSet<Integer> set = new HashSet<Integer>();  //存储到set集合里面  for (int i = 0; i < nums.length; i++) {  set.add(nums[i]);  }  int count = 0; //记录元素的个数  int result = -1; //结果  for (Integer data : set) {  for (int i = 0; i < nums.length; i++) {  if (data == nums[i]) {  result = nums[i];  count++;  }  }  //如果个数大于数组长度的一般,直接结束循环  if (count > nums.length /2) {  break;  }else {  result = -1; //重置为-1  count = 0; //重新记录  }  }  return result;  } |

**思路2:**

1). 使用摩尔算法,先得出一个当前数组最多相同的值

2). 用这个最多相同的值,进行比较,看看是否相同的值大于总值的一半

解释(来源于知乎https://www.zhihu.com/question/49973163):

摩尔算法: 核心就是对拼消耗。

例如:

玩一个诸侯争霸的游戏，假设你方人口超过总人口一半以上，并且能保证每个人口出去干仗都能一对一同归于尽。最后还有人活下来的国家就是胜利。

那就大混战呗，最差所有人都联合起来对付你（对应你每次选择作为计数器的数都是众数），或者其他国家也会相互攻击（会选择其他数作为计数器的数），但是只要你们不要内斗，最后肯定你赢。

最后能剩下的必定是自己人。

|  |
| --- |
| public int majorityElement2(int[] nums) {  int value = nums[0]; //假设一开始值  int count = 1; //计数  //摩尔投票算法  for (int i = 0; i < nums.length; i++) {  if (value == nums[i]) {  count++;  }else {  count--;  }  if (count == 0) {  value = nums[i];  count = 1;  }  }  //比较相同的值  count = 0;  for (int i = 0; i < nums.length; i++) {  if (value == nums[i]) {  count++;  }    //相同的值是否大于总数的一般  if (count > nums.length/2) {  return value;  }  }  return -1;  } |