题目 数组中出现次数超过一半的数字

考点 时间效率 热点指数 54491 通过率 27.01%

具体题目

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。

```
import java.util.Arrays;
public class Solution {
    public int MoreThanHalfNum_Solution(int [] array) {
        Arrays.sort(array);
        int count=0;

        for(int i=0;i<array.length;i++){
            if(array[i]==array[array.length/2]){
                count++;
            }
        }
        if(count>array.length/2){
            return array[array.length/2];
        }else{
            return 0;
        }

    }
}
```

时间复杂度O（n）方法一：用hashMap public class Solution { public int MoreThanHalfNum_Solution(int [] array) { HashMap<Integer,Integer> map = new HashMap<Integer,Integer>();

```
    for(int i=0;i<array.length;i++){

        if(!map.containsKey(array[i])){
           map.put(array[i],1);
        }else{
            int count = map.get(array[i]);
            map.put(array[i],++count);
        }
    }
    Iterator iter = map.entrySet().iterator();
    while(iter.hasNext()){
        Map.Entry entry = (Map.Entry)iter.next();
        Integer key =(Integer)entry.getKey();
        Integer val = (Integer)entry.getValue();
        if(val>array.length/2){
            return key;
        }
    }
    return 0;

}
```

方法二：基于快排思想

public class Solution { public int MoreThanHalfNum_Solution(int [] array) { if(array.length<=0) return 0;

```
    int start = 0;
    int length = array.length;
    int end  = length-1;
    int middle = length>>1;

    int index = Partition(array,start,end);


 while(index!=middle){
 if(index>middle){  index = Partition(array,start,index-1);  }  else{  index = Partition(array,index+1,end);  }  }  int result
= array[middle];
int times = 0; for(int i=0;i<length;++i){ if(array[i] == result) times++; } if(times*2<length){ System.out.println(times);
return 0; }else{ return result; } }

    public int Partition(int[] array,int start,int end){
     int flag = (array[start]+array[end])/2;

     while(start<end){
         while(array[end]>flag){
             end--;
         }
         swap(array,start,end);
         while(array[start]<=flag){
             start++;
         }
         swap(array,start,end);
     }
     return start;
 }
 public void swap(int[] array,int num1,int num2){
     int temp =array[num1];
     array[num1] =array[num2];
     array[num2] =temp;
 }

}
```

方法三：基于数组特点

public class Solution { public int MoreThanHalfNum_Solution(int [] array) { if(array.length<=0){ return 0; } int result =
array[0]; int times = 1;

```
    for(int i=0;i<array.length;i++){
        if(times==0){
            result = array[i];
            times =1;
        }else if(array[i]==result)
            times++;
         else
            times--;
    }
    int time = 0;
    for(int i=0;i<array.length;++i){
        if(array[i] == result)
            time++;
    }
    if(time*2<array.length){
        System.out.println(time);
        return 0;
    }else{
```

```
            return result;
        }
    }

}
```

import java.util.HashMap; import java.util.Map; /*

- 利用map存值，找出存在最多的数字，若大于长度一半，返回此数，否则返回0 */ public class Solution { public int MoreThanHalfNum_Solution(int [] array) { if(array.length==0||array==null) return 0; Map<Integer,Integer> map=new HashMap<Integer,Integer>(); for(int i=0;i<array.length;i++){ if(map.containsKey(array[i])){ map.put(array[i], map.get(array[i])+1); }else{ map.put(array[i], 1); } } for (Map.Entry<Integer, Integer> entry : map.entrySet()) { if(entry.getValue()>array.length/2) return entry.getKey(); } return 0; } }