题目 调整数组顺序使奇数位于偶数前面

考点 代码的完整性 热点指数 79796 通过率 25.80%

具体题目

输入一个整数数组，实现一个函数来调整该数组中数字的顺序，使得所有的奇数位于数组的前半部分，所有的偶数位于数组的后半部分，并保证奇数和奇数，偶数和偶数之间的相对位置不变。

时间复杂度为O（n），空间复杂度为O（n）的算法

```
/*
整体思路：
首先统计奇数的个数
然后新建一个等长数组，设置两个指针，奇数指针从0开始，偶数指针从奇数个数的末尾开始  遍历，填数
*/

public class Solution {
    public void reOrderArray(int [] array) {
        if(array.length==0||array.length==1) return;
        int oddCount=0,oddBegin=0;
        int[] newArray=new int[array.length];
        for(int i=0;i<array.length;i++){
            if((array[i]&1)==1) oddCount++;
        }
        for(int i=0;i<array.length;i++){
            if((array[i]&1)==1) newArray[oddBegin++]=array[i];
            else newArray[oddCount++]=array[i];
        }
        for(int i=0;i<array.length;i++){
            array[i]=newArray[i];
        }
    }
}
```

从题目得出的信息： 相对位置不变--->保持稳定性；奇数位于前面，偶数位于后面 --->存在判断，挪动元素位置； 这些都和内部排序算法相似，考虑到具有稳定性的排序算法不多，例如插入排序，归并排序等；这里采用插入排序的思想实现。

```
public class Solution {
    public void reOrderArray(int [] array) {
        //相对位置不变，稳定性
        //插入排序的思想
        int m = array.length;
        int k = 0;//记录已经摆好位置的奇数的个数
        for (int i = 0; i < m; i++) {
            if (array[i] % 2 == 1) {
                int j = i;
                while (j > k) {//j >= k+1
                    int tmp = array[j];
                    array[j] = array[j-1];
                    array[j-1] = tmp;
                    j--;
                }
                k++;
            }
        }
    }
}
```

```
public class Solution {
    public void reOrderArray(int [] array) {
        for(int i=0;i<array.length-1;i++)
            for(int j=0;j<array.length-i-1;j++){
                if(array[j]%2==0 && array[j+1]%2==1){
                    int temp=array[j];
                    array[j]=array[j+1];
                    array[j+1]=temp;
                }
            }
    }
}
```