题目 旋转数组的最小数字

考点 查找和排序 热点指数 89987 通过率 31.97%

具体题目

把一个数组最开始的若干个元素搬到数组的末尾，我们称之为数组的旋转。 输入一个非减排序的数组的一个旋转，输出旋转数组的最小元素。 例如数组{3,4,5,1,2}为{1,2,3,4,5}的一个旋转，该数组的最小值为1。 NOTE：给出的所有元素都大于0，若数组大小为0，请返回0。

```
public class Solution {
 /*
 * 传进去旋转数组，注意旋转数组的特性：
 * 1.包含两个有序序列
 * 2.最小数一定位于第二个序列的开头
 * 3.前序列的值都>=后序列的值
 * 定义把一个数组最开始的若干个元素搬到数组的末尾，我们称之为数组的旋转。
 * ^_^这个旋转思想是很经典的
 * 旋转数组实例：
 * {123456}旋转后{456123}
 */

 //用到了快速排序的快速定位范围的思想，
 public int minNumberInRotateArray(int [] array) {
    if(array==null||array.length==0)
    {  return 0;
    }
    int low=0;
    int up=array.length-1;
    int mid=low;

 // 当low和up两个指针相邻时候，就找到了最小值，也就是
 //右边序列的第一个值

 while(array[low]>=array[up]){
 if(up-low==1){
 mid=up;
 break;
 }
 //如果low、up、mid下标所指的值恰巧相等
 //如：{0,1,1,1,1}的旋转数组{1,1,1,0,1}
 if(array[low]==array[up]&&array[mid]==array[low])
 return MinInOrder(array);
    mid=(low+up)/2;
 //这种情况，array[mid]仍然在左边序列中
 if(array[mid]>=array[low])
 low=mid;//注意，不能写成low=mid+1；
 //要是这种情况，array[mid]仍然在右边序列中
 else if(array[mid]<=array[up])
 up=mid;
 }

 return array[mid];

    }
 private int MinInOrder(int[] array) {
 // TODO Auto-generated method stub
 int min =array[0];
 for(int i=1;i<array.length;i++){
```

```
 if(array[i]<min){
 min=array[i];

 }
 }
 return min;
 }
 public static void main(String[] args) {

 }
}
```

```
public class 旋转数组的最小数字 {
    public int minNumberInRotateArray(int[] array) {
        if (array == null || array.length == 0) {
            return -1;
        }
        int len = array.length;
        int index1 = 0;
        int index2 = len - 1;
        int indexMid = index1;
        while (array[index1] >= array[index2]) {
            if (index1 == index2 - 1) {
                indexMid = index2;
                break;
            }
            indexMid = index1 + (index2 - index1) / 2;
            if (array[index1] <= array[indexMid]) {
                index1 = indexMid;
            } else if (array[indexMid] <= array[index2]) {
                index2 = indexMid;
            }
        }
        return array[indexMid];
    }
}
```

```
import java.util.ArrayList;
public class Solution {
    public int minNumberInRotateArray(int [] array) {
        if(array.length==0)
            return 0;
        else
            return partition(array,0,array.length-1);
    }
    //递归的目的是寻找乱序的子数组
    private int partition(int [] array,int start,int end){
        if( array[start] < array[end] || start == end )  //如果第一个元素小于最后一个元素，说明数组从头到尾都是非减的；如果只剩下一个元素，则直接返回
            return array[start];
        else {
            int mid=start+(end-start)/2;
            if( array[mid] < array[end]){    //如果中间值下于最后的值，说明后半部分为非减序列，所以在前半部分继续寻找；
                                //另外，之所以是mid而不是mid-1，是为了防止出现越界的情况,例如，array=[3,4],那么start=0，mid=0,end=1; (mid-1)等于-1,不可行
                return partition(array,start,mid);
            }else if(array[mid] == array[end]){    // 如果array=[1,0,1,1,1]或者[1,1,1,0,1]，那没办法判断乱序子数组的位置，所以只能削减一步
```

```
                return partition(array,start,end-1);
            }else{                  //如果中间值大于最后值，那么说明乱序的部分在后半段，所以在后半段寻找。可以使用mid+1是因为，中间值都比最后值大了，那还要它干嘛？
                return partition(array,mid+1,end);
            }
        }
    }
}
```