数组中的逆序对\_剑指Offer\_51

# 数组中的逆序对\_剑指Offer\_51

## 题目描述

\* 在数组中的两个数字，如果前面一个数字大于后面的数字，

\* 则这两个数字组成一个逆序对。输入一个数组,求出这个数组中的逆序对的总数P。

\* 并将P对1000000007取模的结果输出。 即输出P%1000000007

\* 示例：输入 1,2,3,4,5,6,7,0 输出 7

## 思路分析

\* 思路：归并思想。归并思想需要复制一个该数组作为辅助数组，空间复杂度O(N)。

\* 递归分割，原数组和辅助数组交替交换角色使用，直至为子数组长度为1时，

\* 逐层返回，返回之前需要统计数目和将两个子数组归并排序。

\* 递归思想：递归终止条件low==high，返回0；

\* 首先获取左子数组内部的逆序对数；(递归)

\* 然后获取右子数组内部的逆序对数；（递归）

\* 最后将两个子数组合并，并合计所有的逆序对数。（归并）

\*

\* 由于数字比较大，在求余之前，利用long存储数组，否则无法通过。

\*/

## Java代码

public int **InversePairs**(int [] array) {

if(array == null||array.length <= 1) return 0;

int[] copyArray = Arrays.copyOf(array,array.length);//创建辅助数组

long count = countInversePairs(array,copyArray,0,array.length-1);

return (int)(count%1000000007);

}

/\*\*

\* 递归方法

\* 主要是利用了归并思想

\*/

public long **countInversePairs**(int[] array,int[] copyArray,int low,int high){

if(low == high){//复制这一个即可

copyArray[low] = array[low];

return 0;

}

int halfLen = (high - low + 1)>>1;//一般的长度

//递归调用获取左子数组中的逆序对数

long leftCount = countInversePairs(copyArray,array,low,low+ halfLen -1);

//递归调用获取右子数组中的逆序对数

long rightCount = countInversePairs(copyArray,array,low+halfLen,high);

//合并两个子数组为一个排序数组

long count = leftCount + rightCount;

//归并思想

int i = low+halfLen-1, j = high, indexCopy = high;

while(i >= low &&j >= low+halfLen){

if(array[i] > array[j]){

copyArray[indexCopy--] = array[i--];

count += j-low-halfLen + 1;

}else{

copyArray[indexCopy--] = array[j--];

}

}

while(i >= low) copyArray[indexCopy--] = array[i--];

while(j >= low+halfLen) copyArray[indexCopy--] = array[j--];

return count;

}