14.标题：数组中的逆序对

有一组数，对于其中任意两个数组，若前面一个大于后面一个数字，则这两个数字组成一个逆序对。请设计 一个高效的算法，计算给定数组中的逆序对个数。

给定一个int数组**A**和它的大小**n**，请返回A中的逆序对个数。保证n小于等于5000。

测试样例：

**[1,2,3,4,5,6,7,0],8**

返回：**7**

import java.util.\*;

public class AntiOrder {

public int count(int[] A, int n) {

if (A == null || n == 0) {

return 0;

}

return mergeSortRecursion(A, 0, n - 1);

}

public static int mergeSortRecursion(int[] arr, int l, int r) {

if (l == r) {

return 0;

}

int mid = (l + r) / 2;

//逆序对的总数=左边数组中的逆序对的数量+右边数组中逆序对的数量+左右结合成新的顺序数组时中出现的

逆序对的数量；

return mergeSortRecursion(arr, l, mid) + mergeSortRecursion(arr, mid + 1, r) +

merge(arr, l, mid, r);

}

public static int merge(int[] arr, int left, int mid, int right) {

int[] temp = new int[right - left + 1];

int index = 0;

int i = left;

int j = mid + 1;

int inverseNum = 0;// 新增，用来累加数组逆序对

while (i <= mid && j <= right) {

if (arr[i] <= arr[j]) {

temp[index++] = arr[i++];

} else {

// 当前一个数组元素大于后一个数组元素时，累加逆序对

// s[i] > s[j] 推导出 s[i]...s[mid] > s[j]

inverseNum += (mid - i + 1);

temp[index++] = arr[j++];

}

}

while (i <= mid) {

temp[index++] = arr[i++];

}

while (j <= right) {

temp[index++] = arr[j++];

}

for (int k = 0; k < temp.length; k++) {

arr[left++] = temp[k];

}

return inverseNum;

}

}