Coursera Algorithms week3 归并排序 练习测验: Counting inversions

题目原文：

An inversion in an array a[] is a pair of entries a[i] and a[j] such that i<j but a[i]>a[j]. Given an array, design a linearithmic algorithm to count the number of inversions.

分析：

如果没有性能限制，用插入排序算法可以实现。题目性能被限制在nlogn，又是归并排序的练习题，很显然要实现个归并排序，并在里面计数。通过一个例子来看下计数方法，例如{6, 5, 2, 3, 4, 1}这个序列，归并过程中会分为分为如下几步：

1. 归并{6, 5}，存在一次右侧元素5移至左侧的操作，变成{5, 6}，归并前逆序对<6,5>

2. 归并{5, 6, 2}，存在一次右侧元素2移至左侧的操作，变成{2, 5, 6}，归并前存在两对逆序对<5, 2>和<6, 2>

3. 归并{3, 4}，不存在右侧元素左移操作

4. 归并{3, 4, 1}，存在一次右侧元素1移至左侧的操作，变成{1, 3, 4}，归并前存在两对逆序对<3, 1>和<4, 1>

5. 归并{2, 5, 6, 1, 3, 4}，存在三次1、3、4右侧元素移至左侧的操作，归并前存在七对逆序对<2,1><5,1><6,1><5,3><6,3><5,4><6,4>

由上述分析可见，当右侧元素a[j]与左侧元素a[i]进行比较后需要移至左侧时，a[j]与区间 [a[i] , a[mid]]中的所有元素都可以组成逆序对，这个区间的元素个数为mid+1–i个。

因此代码如下：

```
import java.util.Arrays;

/**
 * @author evasean www.cnblogs.com/evasean/
 */
public class CountInversions {
    private static Comparable[] aux;
    private static int num = 0; // 逆序对计数

    private static boolean less(Comparable v, Comparable w) {
        return v.compareTo(w) < 0;
    }

    public static int inversionsNum(Comparable[] a) {
        aux = new Comparable[a.length];
        sort(a, 0, a.length - 1);
        return num;
    }

    private static void sort(Comparable[] a, int lo, int hi) {
        if (hi <= lo)
            return;
        int mid = lo + (hi - lo) / 2;
        sort(a, lo, mid);
        sort(a, mid + 1, hi);
```

```
26          merge(a, lo, mid, hi);
27      }
28
29      private static void merge(Comparable[] a, int lo, int mid, int hi) {
30          int i = lo;
31          int j = mid + 1;
32          for (int k = lo; k <= hi; k++) {
33              aux[k] = a[k];
34          }
35          for (int k = lo; k <= hi; k++) {
36              if (i > mid)  // i>mid表示aux的左半侧已经被全部被放于a中，直接将右半侧部分放入a
37                  a[k] = aux[j++];
38              else if (j > hi) // j>hi表示aux的右半侧已经被全部被放于a中，直接将左半侧部分放入a
39                  a[k] = aux[i++];
40              else if (less(aux[j], aux[i])){ // 右侧元素小于左侧元素时，将右侧元素放入
41                  num += mid+1-i; //此时右侧的这个元素a[j]和[a[i],a[mid]]整个区间的元素都是逆序对
42                  a[k] = aux[j++];
43              }else a[k] = aux[i++];
44          }
45          //System.out.println(Arrays.toString(a));
46      }
47
48      public static void main(String[] args) {
49          Integer[] a = { 6,5,2,3,4,1 };
50          System.out.println(inversionsNum(a));
51          System.out.println(Arrays.toString(a));
52      }
53  }
```