CPE 349: Counting Inversions

Consider the following problem that arises in analyzing rankings to find people with similar tastes. (Once it has been decided two people have similar tastes then their previous choices can be used to make recommendations based on what the other person has liked.) So given a ranking and a database of rankings from other people, how do you find people who have similar rankings? Clearly identical rankings are good and completely reversed ranking is not so good.

We will simplify by assuming everyone is ranking the same n things – say books. A natural approach would be to label the books with integers and label them in the order of the first person's ranking. Then reorder these labels by the ordering of another customer's preference. Then count the number of books that are "out of order."

So consider the following problem, given n numbers $a_1, \ldots, a_n$, we want a measure of how scrambled the numbers are. Think of these numbers as the rankings of the second person. We will do this by counting the number of inversions. The definition of an inversion is given on page 41 of Cormen et. al. in the problems for section Chapter 2

Your assignment is to develop and implement an algorithm to count inversions whose worst case computational complexity measured by the number of comparisons is $\Theta(n \log n)$. You may consult the web and other students for ideas. However all code most be your own.

0. Get an idea of how to solve the problem. Test it on small subproblems, Finally write the pseudo code for your algorithm to count inversions. **MergeSort is a good place to start**.
1. **Implement your algorithm in Java using Divide and Conquer and recursion**. The algorithm must be clear and as simple as possible.
2. Determine the recurrence relation that describes the number of comparisons of the entries of the array that contains the permutation as a function of the length of the array.
3. Solve the recurrence relation by back substitution.

Deliverables:

1. Your class Inversions.java must contain a method, **int invCounter (int [] ranking)** that takes as input a permutation as an array positive integers. Your program should then return the number of inversions in the array. Assume that the input is a valid permutation of positive integers.
2. Submit your Java code for a class **Inversions.java** by 10:00 p.m. on the assignment due date. The method invCounter **must** be designed to contain the divide step, the conquer step, and the combine step. Code must contain good comments. Your class should **not** contain a main method. Use "handin gradertk cpe349assign3 Inversions.java"
3. Also submit a pdf file, CountAnalysis.pdf , that contains
   a. the recurrence relation for the number of comparisons of array entries needed by the algorithm to count the inversions.
   b. Show the derivation of the closed form solution for the number of comparisons as a function of the number of elements in the ranking list using back substitution.
   c.

Input:

```
6 4 3 1
```

Output:

```
6
```

Input:

```
2 3 8 6 1
```

Output:

```
5
```

As usual you can assume the input is as specified. Be sure to test you program thoroughly.