Homework 2 - Problem

Algorithm that finds the inversions, and adds the inversions, when they satisfy i < j when indices are (i,j), and returns the sum of said inversions

```
Let Set A be an Set holding the sequence of integers
Let left be the left index of the sequence (start at 0)
Let right be the right index of the sequence (start at size - 1)

countInversions:
      Let sum = 0
      If left < right then
            Let mid = ( left + right ) / 2
            Sum = Sum + COUNTINVERSIONS(A, left, mid)
            Sum = Sum + COUNTINVERSIONS(A, mid + 1, right)
            Sum = Sum + MERGE(A, left, mid, right
      Return sum

merge:
      Let L be the left set of A
      Let R be the right set of A

      Set i, j = 0
      Set k to left index

      Set count, sum and total sum = 0

      While i < R length and j < L length then
            If L(i) <= R(j) then
                  sequence(k++) = L(i++),
                  sum = sum + count * L(i - 1) + totalSum
            Else
                  sequence(k++) = R(j++),
                  Count = Count + 1
                  totalSum = totalSum +  L(j - 1)
      While (i < L.length) {
            sequence(k++) = L(i++)
            sum += count * L(i - 1) + totalSum;
      While (j < R.length) {
            sequence(k++) = R(j++);

      Return sum
```

This program produces the right output because it divides and conquers, solving the problem recursively while also properly merging the two subarrays after they have been divided

Time estimate: O(n Log n)

Proof (Master Theorem):

Using Case 2: $f(n)$ = Θ(n^(log a/ log b)), $2T(n/2) + n$

Considering a: 2 , and b: 2

This means using Case 2 that $f(n)$ = Θ(n^(log 2/ log 2)) = n^1.