Sum Inversions

My algortihm to find the weights is nearly verbatim the class to find the total inversions with a few small modifications and additions. Knowing that the value of the numbers is central to finding the sum, the statement to increment the inversion count has to be replaced with a statement that increments the count by the sum of the two parts of the currently identified inversion. **However** the problem of leaving out future inversions when incrementing the right index is still real and cannot be solved by incrementing the count by the amount of remaining left elements like before.

For this reason, my algorithm includes a sum of the left elements that is decremented by the value of each left element as the algorithm indexes past it. Then when an inversion is found, the equation **middleWeight += leftSum + rightSorted[j]*(1+leftSorted.len-leftIndex)** is used to increment the weight. This equation represents adding an inversion corresponding to the right-indexed element for every element remaining in the left array (leftSum). Adding the weight of the right element by the amount of left elements yet to be counted ensures AND the leftSum ensures the weight added is equivalent to the weight of what each individual inversion would contribute totalled together.

Running Time:

The only non-constant time addition I made was a for loop the length of the left side that sums all left elements, which runs in O(n) time. This does not affect the overall running time of O(nlogn), because it does not affect the number of recurrences nor the running time of each recursion.