Ryan Kuhl
COP4531

Using Divide & Conquer to Calculate Inversion

As stated in the title, I used a divide and conquer method in tackling this inversion calculation. The merge sort was implemented in order to reorganize the lists into proper order, implementing a global variable, “inv” that counts the total number of inversions. Merge sort offered a means to calculate at or near an ideal O(n lg(n)). The merge is done via three while loops that traverse two lists separately, which are provided by the previous function.

I ran into several problems while creating the python code for this project, however they were mostly syntactic in nature as I am new to Python. I mainly had issues using map, split, strip and file reading operations. Past that a difficult part of the project was getting the correct area in the merge sort in order to output the correct inversion value. Initially I thought that it would be some combination of incremented values located in one or more of the three while loops in the merge sort operation. Upon further thought, the correct solution of taking the incremented value of the while loop which appends B to A subtracted from the length of list A was discovered to be correct. The last real hurdle was file input. Reading in the file, creating a list of strings and type casting those into individual integers proved to be challenging. After that was completed, the main function in my program was made into the unit test function for testing capabilities.

Past those small issues, Python was an extremely adaptable programming language that proved to be far simpler and more elegant in execution that a similar program in C or C++.