SortOfSort Method

First my SortOfSort method takes an array which will be used to be altered throughout the method. First I declare several integers: idx(index in the array), beg(the first position in the array), last(the last position in the array), and lastly path which will be used to sort the elements either in the first, or the end of the array accordingly.

The method first runs the main nested for loop which runs linear time and checks every element and then proceeds to sort it in a increasing order. Then goes trough the base case to check if the array length is of size one or empty, if met, then it just returns the array.

If not, then it proceeds to check the first condition to see whether path is either 2 or 3 which at first was declared as zero, this condition isn’t met so it runs the else condition which uses a helper method created in my Class that swaps two elements in the array, in this case it swaps the elements, in this case the element at position n(array length -1) minus the integer last with the element in the position at index. Then increments the integer last by one to move one more to the left in this case which will result in being next to the last element. From there I equal idx with beg to make this return to the first position on the array. The process repeats one more time until the condition of path being equal to 2 or 3 which then swaps now the max value after having placed the first two maximum elements in the end of the array to now be in the beginning of the array. Again, the helper method helps me swap the elements to the beginning with the elements in the array at position index. Again this condition repeats once more and beginning has now two additions as well as path. When path equals four, the last condition is met which resets path to 0 and begins the process of meeting the condition in the else statement which will again result in swapping the next maximum integer in the array to the next last position helped by having kept the next position by the integer last.

This method keeps repeating N times the number of the elements in the array.

The worst case in this method would be to have the number of inputs be a very large even length number in the array, since it runs in a linear time.

The best case scenario would be to have a short even number of inputs in the array, this way the positionings of the elements would be easier to understand since we are placing them by pairs in both sides of the array