Clayton Warstler

25 February 2023

Programming Challenges

Assignment 2

Easy Sorting – Merge Sort:

For this program, I chose to use three functions. I had the main function, mergeSort function, and merge function. All the main function did was get the lines and call mergeSort. MergeSort then recursively called itself to split down further and further until it could call the merge function to build itself a fully sorted array. After merge was called, it would then take each array and compare both sides of the arrays to find out where to place the current item. The time complexity of this program is O(n log n).



Easy Sorting – Quick Sort:

For this program, I chose to use four functions. I had the main function, partition function, quicksort function, and swap function. The main function did the same thing as the last one, where it gets input and calls the quicksort function. The quicksort function, would then call partition and then recursively call itself. The partition function would find a pivot point and compare and swap. The time complexity of this program is at best O(n log n) and at worst O(n2).



Shake Shake Shaky:

For this program, I used just one main function. The beginning of this function would take the input from the user. After getting the input for the first test case, it would then call the built in “Arrays.sort(candiesInBox)” function to sort the boxes in ascending order. After that, I then moved on to a for loop that starts at the number of boxes minus 1 while it was greater than 0 and the amount of students was also greater than 0. In the for loop, I would then gather the total number of candies, and calculate the average number of candies. Using the maximum and average number of candies I used the built in “Math.max(maxCandies, avgCandies)” function to find the maximum amount of candies a student can have.

