Arely De Leon

CPSC 335-02 Project #3: Sort Race

Merge Sort Complexity:

Merge sort has an average big-o notation of O(nlog(n)). However, within my merge sort function and helper function I have the **createRow()** function that creates a row within the merge sort table on the html canvas. The createRow function has a O(1) run time because it simply takes the array and prints it onto the canvas. So adding O(nlog(n)) + O(1) is still O(nlog(n)). Helper function: merger().

Quick Sort Complexity:

Quick sort is the same as merge sort where its average run time is O(nlog(n)) and it also uses the createRow function to add to its own table. So it also has a run time of O(nlog(n)). Helper functions: swap() and partition().

Insertion Sort Complexity:

Insertion sort is a little different as its average run time is O(n^2). However, it also uses the createRow function and adding that to it makes its run time still O(n^2). Has no helper functions.

CreateRow Complexity:

CreateRow has a run time of O(1). It takes the array or sections of the original array to display in each sorting functions respective table. For merge sort it takes each divided and recombined section of the array and displays them. For quick sort it takes the whole array after each swap is made. Insertion sort is like quick sort where it displays the whole array after each index of the array is checked and sorted.

Running Complexity:

Running is my manager function since it runs the three sorting algorithms and fills the three tables in the canvas. However the tables are made in the html code.