RUNNING TIME ANALYSIS PAPER

In project 3, we are supposed to create a Sort Race that uses 3 different algorithms in order to test which algorithm is the fastest. By using three O(N*logN) sorting algorithms such as Mergesort, Quicksort, and Selection sort, we compared each one of them within each other in order to find which one is the fastest. However, we also found out that each algorithm has different run times.

For the merge sort, it has an average and best-case running time of O(nlogn), which means it is the fastest algorithm, so no algorithm can be faster than this. However, if there is more complexity within the algorithm, the time for the algorithm will take longer, which could in some instances will not be desired. The performance of the merge sort is 2 * T(N/2) + N.

As for quicksort, similarly to the mergesort, it is a recursive method, which uses the O(nlogn) as well. It doesn’t get any faster than this. Even in its worst case, quicksort is still one of the fastest methods around.

Finally, we have the selection sort. For the selection sort, we have 3 parts of the loop that we have to find the running time. For the first part, we are finding the minimum index, which takes a constant time to complete. So the running time is O(n^2). As for the other parts of the loop that we use, it will take just O(n) since we are testing one value at a time.

When we run our program, we can obviously see that merge sort runs at a more efficient than quicksort if we are to deal with bigger number size in arrays; where as quicksort is more efficient and faster than mergesort if we were to deal with small number size arrays since quicksort is an internal algorithm while mergesort is quite the opposite. As for selection sort, we believe it to be the slowest among all the algorithms since it is required for us to scan the whole array for the desired element, rather than scanning until we reached the desired element, which will run at a slower pace than the other algorithms.