Kurk Fisher

Computer Algorithms

Assignment 1

Discuss what your results mean regarding the theoretical run-time of the different algorithms.

As expected, each sorting algorithm perform pretty much the same when the number sets were low but as the number sets increase so did the differences between each algorithm. Quicksort did the best among the algorithms. It’s runtime never went above 1 millisecond. Insertion sort did the worst among algorithms. It had the highest run time among the algorithms. Heap and merge sort ran similarly in terms of times.

Do the sorts really take O(n2) and O(n lg n) steps to run?

No, it did not take worse case for algorithms to sort the number. I came to this conclusion by comparing the largest sorted sets numbers run times with their respective values of O(n2) and O(n lg n). All the sort algorithms didn’t even come close to their average case when I did my calculation.

Which of the sorts takes the most steps?

The two sorting algorithms that took the most steps were insertion sort.

Which of the O(n lg n) sorts takes the most steps?

The two sorting algorithms with O(n lg n) that took the most steps were heap sort and merge sort.

Why?

Heap sort used the most comparison in its sorting but didn’t move the data around a lot. Merge sort didn’t do a lot of comparison but move the data around a lot. This could be because that both heap sort and merge sort Best, Average, and worst case does not change with order of number set.

Under what circumstances might you prefer to use one of the sorts versus others?

If I wanted the fastest results, I would use quick sort because it performed the best out of the sorting algorithms with the least number of steps. If I valued consistency over speed, I would go with merge sort or heap sort because they always give the same run time matter the order of the data set.

In general, which sort seems preferable?

I would say that Quicksort is preferable because on average it will preform faster or just as fast a heap sort and merge sort while doing it with least steps.