Patrick Polcuch

CPSC350-02

Assignment 6 Write up

On this assignment, I decided to format the time as hours:minutes:seconds. I generated 100 and copy pasted them to reach sufficiently large numbers. Not surprisingly, all the sorting algorithms ran under a second for inputs that were less than about NUMBER. When I sorted one hundred thousand, I got the following output:

Quick Sort: 0:0:1
Merge Sort: 0:0:0
Selection Sort: 0:0:9
Insertion Sort: 0:0:7
Bubble Sort: 0:0:16

I was surprised that Merge Sort seemed to sort it instantly. I ran it a few more times with the similar results. I had to check to make sure that it actually sorted the numbers. But apparently merge sort is just that fast.

I doubled the input to two hundred thousand numbers. As expected, Selection sort, Insertion sort, and Bubble sort took about four times as long. I was a little surprised that Quick sort took four times as long as well. Merge sort still took under a second.

I was suspicious about this, so I regenerated my random numbers so that they repeated after ten thousand. I then sorted two hundred thousand numbers again. Selection sort and insertion sort remained close to their previous time. Bubble sort was almost twice as slow. This was unexpected but makes sense after some thought. Quick sort and Merge sort are now both under a second.

While Quick sort normally has a runtime of nlog(n), the worst-case runtime is n^2, as shown by the runtimes of my two different data sets. Merge sort has a runtime of nlog(n) for all cases, but it has the downside of taking up n additional memory. I watched my memory usage spike when my program executed. Selection sort has a runtime of n^2 and is relatively simple. Insertion sort also has a runtime of n^2, but runs closer to n on nearly sorted lists. Bubble sort is by far the slowest of these sorting algorithms, taking almost twice as long as the next slowest algorithm. It has a runtime of n^2, but it is by far the easiest algorithm to code.

This analysis of these five algorithms was simple and easy, but not perfect. It would be more accurate to compare the amount of CPU ticks each algorithm took, but reading minutes and seconds is easier. It would also be nice to test the efficiency of these algorithms on many different data sets, with varying degrees of sorted-ness.

I tried sorting 5 million numbers and I got these results:

Quick sort: 4 seconds
Merge sort: 1 second
Selection sort took over 25 minutes, so I cancelled it.