Assignment 6: Sorting Algorithms

stone188

May 2017

1 Time

Since the test data set utilized was no more than 1000 numbers, there appeared to be no significant differences in run time complexity. No real observable difference in time complexity will be observed when the set is under 10000 entries. Upon further research, I discovered that at 50000 entries and up, the bubble sort method takes significantly longer while (of course) quick sort remains at only a fraction of a second. Quick sort is the clear winner when it comes to duration as it hardly takes more time as the data set increases.

2 Trade Off

Quicksort (Avg. - n log(n) / Worst - $\mathbf{n}^2$)
Great for most cases, but loses performance in the worst case. Space complexity is O(log(n)), so great for larger data sets."

Insertion Sort (Avg. - $\mathbf{n}^2/Worst - n^2$)
Taxing with larger data sets. Best suited for small sets do to average and worst case run times."

Shell Sort (Avg. - n log(n)$^2/Worst - nlog(n)^2$)
Shell sort allows for index swapping that are far apart. Also easy on stack memory. Not as fast as quicksort, but easy to implement.

3 Conclusion

Empirical analysis runs into some serious problems due to the almost non-existent variation in run times between these sorting algorithms. Without knowing Big-O notation, we would have to test increasingly larger data sets and time each one in order to reach a decisive conclusion on which algorithm to employ. This is very taxing on time and could result in heavy CPU usage. On the other hand, asymptotic analysis allows programmers to make appropriate decisions immediately and without wasted effort.