William James

Rene German

Data Structures and Algorithms

5/20/2016

Time Comparison Report

In this assignment, I had to compile sorting functions in C++ for sorting an array of doubles and record the time it takes for each sorting algorithm. The algorithms I used were quicksort, insertion sort, and shell sort. The differences in sorting times were much more drastic then I expected. In order of fastest to slowest, the algorithms were quicksort, insertion sort, then shell sort.

Quicksort amazed me the most with how fast it sorted the algorithms as it is average BigO (n(logn)). This algorithm works well when you want to sort a large list as fast as possible. Insertion sort also was very impressive with its speed, but was not as fast as quicksort since it is average BigO (n^2). Insertion sort would work better for smaller input file compared to the other two algorithms. Also, insertion sort works the best when the array is partially sorted. Shell sort was the slowest of the three sorting algorithms as its average BigO ((n(logn))^2). Shell sort will work better than the others if you are trying to sort a very large amount of data. Shell sort is also more effective if the array items are shuffled far apart from their original order. The time of these sorting functions would have been different if I were using another programming language. If I were to have used Java, the time to sort these algorithms would have been slower. Since I used C++ I was able to sort these algorithms more efficiently than I would have if I used Java.

The shortcomings of the empirical analysis of these sorting algorithms were the fact that we had to implement and run quicksort, insertion sort, and shell sort. We could only compare the sorting algorithms in C++ and on a Linux based system. So we don't know what would happen with other languages and other operating systems. Overall I learned a lot from this assignment, and sorting algorithms are very helpful when sorting data and the time differences are very interesting.