Caden MacKenzie

Analysis of Algorithms

March 31, 2016

Lab Book Analysis

After implementing and completing the analysis of several sorting algorithms, it is clear that each one has strength’s and weakness’s when compared to the others. The first algorithm analyzed was Exchange sort. Exchange sort is not an ideal sorting algorithm for large sets of data. While exchange sort is unique in the fact that the best case in O(n) when the list is already sorted, the worst case and average case are O(n^2).

The second algorithm that was analyzed was Quicksort. While the worst case takes O(n^2). The worst case is faced when the pivot is either the highest or lowest number in the list creating a list with all the items on one side and a list of zero items on the other. If this continues to happen, each call will create a list that is one smaller than the previous instead of the best case where each call splits the list into two even sub lists. The best and average case take O(n logn).

The next algorithm analyzed was merge sort. Merge sort turned out to be the most efficient sorting algorithm. Merge Sort has a performance of O(n logn) for its worst, average and best case. Because it does so many fewer comparisons than say, quicksort, its worst case is as good at quicksort’s best case. Merge sort uses the divide and conquer algorithm explaining the order in which it runs. When the list of items being sorted is doubled, the algorithm only has to split the list one more time.

Lastly, insertion sort was more efficient than exchange sort, however fell behind both merge sort and quicksort in running time analysis. The best case performance of O(n) can be explained by understanding that algorithm doesn’t make any moves of items or insertions into the data if the list is already in order. The worst and average case performance increasing dramatically to O(n^2), similar to that of exchange sort.

It can be seen in the graph that exchange sort and insertion sort show a exponential line of best fit when the size of n increases. The data for merge sort and quicksort have a few more outliers when zoomed in due to the act that the change in time from one set to another is so small. From a distance the O(nlogn) line can be seen for both algorithms.