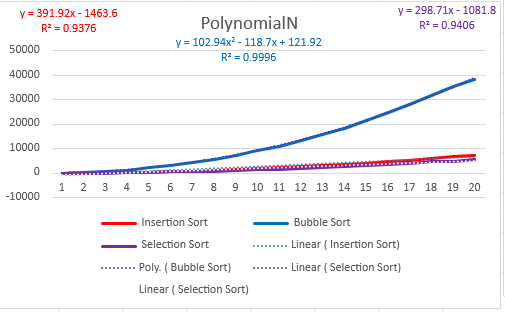
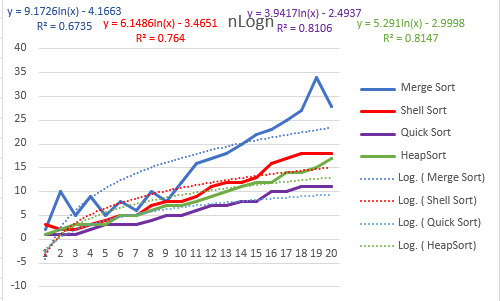
Jaylen Knight

CSCI 3103-001

Sorting Methods and time complexity report.

I have created a program that is designed to test and record the time it takes for the 7 more commonly employed sorting algorithms with the purpose of taking in an array or list of elements and arranging them in a given order. For this we targeted to arrange an array of numbers spanning from 10,000 – 200,000 in increments of 10,000 in order of least to greatest. The algorithms we utilized were insertion sort which iterates through each element of the list and compares it with the first 2 elements present, merge sort which constantly splits the array to then merge them back together after sort, shell sort which is a variant of insertion sort also utilizing forming a split in the middle of the array for sorting, bubble sort which constantly swaps given elements until they are in the correct position, selection sort which finds the smallest element immediately and puts it at the start then repeats for the second smallest, quick sort which randomly picks an element and its neighbors to be sorted around until it is done, and lastly heap sort which converts the array to a complete binary tree for searching purposes similar to selection sort.

I have compiled the times taken for each sorting algorithm on their respective array size of random integers from a random range of zero to 1 million to increase the likelihood of numerous comparisons for each algorithm. Graphs shown below are also divided into the 2 groups of which their average time complexity is O(n^2) and O(n log n) respectively.

The Polynomial of N algorithms include Insertion sort, Bubble sort, and Selection Sort while the n log n algorithms include Merge sort, Shell sort, Quick sort, and Heap sort. The time is recorded in milliseconds and included with each plot is the equation the plot represents within the bounds of the data and the variance value of the equation which represents what we found versus what we expected for a general time increase between each plot. The Polynomial N graph shows by far the longer sorting methods, mainly bubble sort which as we introduce bigger arrays of data the time taken massively increases due to its method of sorting. The problem with bubble sort is how as it iterates through the list it swaps the element with its next element once then moves on from there while starting from the beginning after it reaches the end of the list until every element is in the correct order. This results in an exponentially increasing number of loops within loops being required as we introduce bigger sets of data.

2 similarly paced algorithms Insertion sort and Selection sort are on the much lower end of the O(n^2) graph while still being considered O(n^2). Although these 2 algorithms sort using different methods they both achieve extremely similar results however one, being insertion sort, is the (almost) objectively stronger sorting method due to also being able to handle multi key order sorting well. What's known as a “Stable” sorting algorithm, insertion sort has the capacity to keep the order of an original key being our array numbers, and a second key for example alphabetical order. Say each number in the array also has a letter either A or B and we also wanted a sort regarding alphabetical order. Insertion sort can keep same letter elements together through the sorting process for example 3A, 5A, 1B, 2B, 4B.

Moving on to the n log n graphs, I want to highlight is the position of shell sort vs insertion sort from the other graph. Shell sort is an alternative to insertion sort by also employing sub arrays of shrinking sizes to perform insertion sort on until each sub array is ordered correctly. This change brings down its average time complexity massively to the point of being in a lower classification when dealing with larger and larger arrays of elements however its worst case scenario which can only happen if every single even index element is greater than the median of the array will it have a time complexity of O(n^2).

These calculations were performed on an AMD Ryzen 7 5800x CPU with other tasks running on it at the same time which may affect some outcomes of results.