Sorting Algorithms

Justin Ewoldt, *Student*, *Chapman University*

Abstract**-- In computer science, sorting algorithms are used to put elements of a list in a certain order. There are many different types of sorting algorithms, and depending on the situation variables, one algorithm will perform more efficient than its counterparts. Choosing the optimal algorithm is crucial, as this step will trickle down and make future operations increasingly efficient.**

I. Introduction

During this assignment, my skills as a software engineer were put to the test as I demonstrated my ability to compare and contrast the efficiency of many different sorting algorithms on a list of doubles. I put the list of doubles into my four sorting algorithms-- Quick Sort, Insertion Sort, Selection Sort, and Bubble Sort-- and determined which algorithm ran most efficiently. I did this by running each algorithm on the list, using the built-in C++ clock as a stopwatch. After each algorithm, I stopped the clock and recorded the time. Some algorithms performed faster than their counterparts, and the main focus of this paper is understanding why this was my result.

II. Efficiency of Algorithms

Insertion sort is a sorting algorithm that works by taking elements one by one and inserting them in the list where they fit. A real-world comparison to this algorithm is similar to holding a deck of cards. Insertion sort will be most efficient when dealing with small or mostly sorted lists. Selection sort works by taking the minimum element in the list and moving it to the beginning, and repeats this until the entire list is sorted. As you can probably assume, this is very slow when taking on large lists. Quick sort is described as a "divide and conquer" algorithm. It begins by selecting a number near the median as its "pivot". It then proceeds to move anything greater than the pivot on the left side of the list to the right side of the list, and vice versa. Bubble sort is the simplest of algorithms, starting at the beginning of the list and comparing each element to the element in front of it, if the first element is greater, then the elements are swapped.

III. My Findings

Based on my tests, I found that in a list of five thousand random doubles, my Quick sort algorithm worked most efficiently, with a time of 0.0015 seconds. Its counterparts performed slower at-- *Bubble Sort: 0.3689, Select Sort: 0.030, Insert Sort: 0.3621.* This confirms my predictions, as Quick sort's divide and conquer method is superior on large lists compared to its counterpart algorithms.