Sorting Algorithms in Practice

Donner Hanson
Schmidt College of Science and Technology
Chapman University
Department of Computer Science

May 16, 2019

Abstract

This paper will discuss the differences of the sorting algorithms Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, and Recursive Quick Sort.

Outline This report is organized as follows:
Section 1 : Introduction
Section 2 : Bubble Sort
Section 3 : Selection Sort
Section 4 : Insertion Sort
Section 5: Merge Sort
Section 6 : Quick Sort
Section 7 : Conclusion

1 Introduction

Assignment 6 is a demonstration of how differing sort algorithms work. To construct the txt file included I used the standard distribution functionality from C++11. I generated random doubles in a range and output to the txt file. The program also skips any whitespace and characters that may be included in the data to sort. The internal data structure I created to hold the values of the data to sort is a custom Dynamic Array class. When the

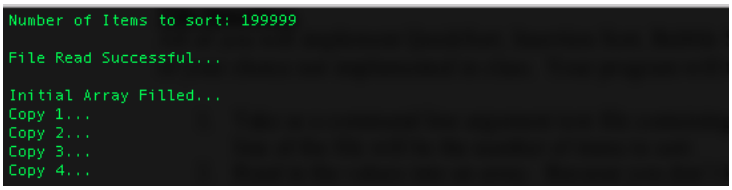

Figure 1: Number of Items Sorted

base array has reach its max length it is copied to a new array, pointers are swapped, and the old array deleted. To time the sorting of the data I made use of the C++11 std::chrono library to calculate the time of each sort in microseconds.

2 Bubble Sort Results

Bubble sort is great for a quick solution to a small set of numbers. However, when the size of the data reaches into the hundreds of thousands, or millions, my hardware (2012 Macbook Pro) runs at max capacity slowing to a sluggish pace. This was my first time sorting large amounts of data so I was quite shocked to see that I could actually get some chores done and eat dinner while I waited for the results while waiting for this $O(n^2)$ algorithm to finish.

Figure 2: Bubble Sort Results

3 Selection Sort Results

Selection sort seemed to run slightly faster than bubble sort with large sets of data but the change was not nearly that remarkable. Again, an $O(n^2)$ algorithm has surprised me.

Figure 3: Selection Sort Results

4 Insertion Sort Results

Insertion sort actually seemed to run far quicker than expected - especially after watching Bubble and Selection sorts run at a snails pace, even though Insertion Sort is also considered $O(n^2)$. This could have been due to the uniformly distributed random numbers that were loaded into the doubles.txt file being somewhat sorted. However, the sort still took a while.

Figure 4: Insertion Sort Results

5 Merge Sort Results

Merge sort wowed me. It brought a better understanding of what an O(nlogn) algorithm can complete compared to the previous algorithms. In under a second, 199999 items had been sorted.

Figure 5: Merge Sort Results

6 Quick Sort Results

To my surprise, Quick Sort was, well... quick! Again, impressed by the speed of the recursive algorithms.

Figure 6: Quick Sort Results

7 Conclusions

Recursion is useful and, given the right environment and specs of data, should be embraced for efficiency. I was pleasantly surprised by the efficiency of Quick Sort as well as Merge Sort and will be implementing them more frequently where appropriate.