Assignment 6 Algorithms Report

Colton Wedell
Assign6
Rene German
CPSC 350-03
12/13/2019

I. Introduction

This report analyzes the performance of five separate sorting algorithms used on a large dataset of integers. The algorithms discussed are bubble sort, selection sort, insert sort, quick sort, and merge sort.

II. Setup

A. Dataset

The dataset used for this test consisted of 200,000 integers on the interval 1 - 1,000,000, inclusive. The size was chosen to both allow a differentiation of performance between each algorithm and minimize the processing and memory requirements of an extremely large dataset.

B. Program

The program used to test these algorithms was written in C++. It takes a plaintext file containing the numbers and reads them into an array. It then duplicates this array four times for a total of five arrays, one for each sorting algorithm. Once each algorithm has sorted the dataset, it prints the starting time, ending time, and elapsed time for each algorithm.

III. Computer Performance

Before the test was run, the computer maintained a CPU utilization rate of about 5%, and 4.6 GB of RAM was available. During the test, CPU utilization rose and maintained a level between 20% and 30%, peaking at 53%. Utilization reached above 50% a total of three times. Memory usage remained steady at 4.5 - 4.7 GB available. The test lasted for 4 minutes, 52 seconds.

IV. Results

Bubble sort: 204 s
Selection sort: 56 s
Insert sort: 32 s
Quick sort: 0.038 s
Merge sort: 0.028 s

V. Discussion

On an initial comparison of the runtime of each algorithm, the advantages of quick sort and merge sort are clearly apparent. They outperform insert sort, the fastest non-recursive algorithm, by about a thousand times. It is obvious why recursion is the tool of choice for an algorithm built for speed.

A. Individual Analysis

1) Non-Recursive Algorithms: Bubble sort, the slowest algorithm, took over three minutes. This algorithm uses nested for loops and directly compares each value in succession, which slows the algorithm significantly. Selection sort also uses nested for loops; however, it partially sorts the data as it traverses the array, and if statements in the inner for loop improve the performance fourfold. The final non-recursive algorithm, insert sort, only uses a single for loop with a nested while loop, which improves runtime almost 50% over selection sort.

2) Recursive Algorithms: These algorithms blow the rest away. Quick sort is based on a pivot value at the right of each array and each item is compared to the pivot to sort it. The array is split in half at each recursion, and swaps are only done if necessary. Merge sort is similar, with the array split in half at each recursion. But, instead of a pivot, it compares values from the left and right sides of the array and merges them (hence the name) back into one. These improvements provide the additional gain in performance compared to quick sort.

B. Additional Comments

The results of this analysis, while expected, were still telling. The differences in performance from bubble sort to insert sort and then to merge sort were even wider than what I expected, with merge sort running almost 10,000 times faster than bubble sort. Quick sort or merge sort are clearly the best options, excepting cases in which memory is limited (merge sort is then a bad choice) or in which recursion is unavailable.

While this test was limited in scope, and it would have been intriguing to attempt with a much larger dataset and a more powerful processor, the amount of data used was more than enough to distinguish the performance of each unique algorithm.

The use of C++ for this test was certainly sufficient to demonstrate the expected results. Using a language with automatic memory management would have likely varied the end outcome, especially in the case of merge sort with its heavy memory requirements. It is telling that the SPEC benchmarks for computer performance are written mainly in C, and occasionally in C++ or FORTRAN; these languages have the power and capability required for standardized tests such as these.