ANALYSIS OF SORTING ALGORITHMS

March 19, 2018

Scott McCoy
108786190

Contents

Introduction

The goal of this project was to learn about several different sorting algorithms and the relationship between their time complexities and their real world run times. This was achieved by implementing each algorithm in C++ and observing, graphing, and comparing the changes in run times for each algorithm when using different lengths and configurations for the array. A statistical analysis of the run times of each configuration and algorithm was also performed.

Methods

The algorithms that were implemented were the insertion sort, selection sort, bubble sort, merge sort, and quick sort. Each sorting algorithm was ran with arrays of lengths 10, 1000, 10,000, 100,000, and 1,000,000. The four different configurations for the arrays were already sorted, randomly shuffled, sorted in reverse, and shuffled at 10 percent. Each combination of array length and configuration was sorted 100 times using each algorithm to produce the data for the statistical analysis. A menu system to allow a user to choose the type of sort, the length of the array, and the type of the array was also implemented.

Results

The results of the tests are represented by the following tables.

Discussion

In a broad sense, the results from the tests were as to be expected. Bubble and insertion sort were expected to run in n^2 time in most cases with the exception of the already sorted array being sorted in n time. The 10 percent shuffled array also did slightly better but still not as well as the fully sorted array as seen in tables 1 and 3. The selection sort also performed with the expected n^2 time in every case. You can see this in table 2 because every test for a single size will be raised to the same power signifying that the complexity for each test is approximately the same. Merge sort also exhibited this behavior as seen in table 4. This is to be expected because merge sort is in θ(n log n).

Statistical Analysis

Conclusion