Sorting Algorithms

Jillian Anderson

I. Abstract

This paper will analyse the results of the implementations of four different sorting algorithms.

II. Introduction

There are many types of sorting algorithms to choose from depending on your current situation. Some are quicker than other while some are more space efficient. Each of the sorting algorithms have their own pros and cons. In this paper, we will be addressing and discussing the benefits and fall backs of Bubble Sort, Selection Sort,Insertion Sort, and Quick Sort

III. Time Differences

The standard Big O run times for the algorithms is question are:

- Bubble Sort: $O(n^2)$
- Selection Sort: $O(n^2)$
- Insertion Sort: Worst: $O(n^2)$ Best: $O(N)$
- Quick Sort: $O(n * log(n))$

When I implemented these algorithms I needed a long list of numbers to test them out. I created a text file and filled it with a random list of 100,000 double numbers. I then ran the program using the text file and recorded the results.

Here are the results of my test in order from slowest to fastest:

- Bubble Sort: 47.8015seconds
- Quick Sort 11.7952seconds
- Selection Sort 11.6677seconds
- Insertion Sort 7.02327seconds

Looking at my test results, some of their are to be expected but some of them are a surprise to me. Bubble sort was expected to be the slowest. This is because bubble sort uses a very brute force technique and doesn't have the most coherent implementation.

However, Insertion Sort performing the best in terms of time complexity was definitely a surprise to me. I wondered how this sorting method could beat Quick sort, which is supposed to have a faster worst case run time, but also how it could beat Selection sort by so much when their run times are supposed to be comparable. After some thought I realised that there was an obviously explanation.

The run time of Insertion sort has the possibility of increasing to linear depending on the data. More specifically, if the data set contains somewhat presorted data, then Insertion sort run time can change from $O(n^2)$ to O(n). This is significantly faster, and therefore makes Insertion sort the obvious choice for partially presorted data.

During the testing of my sorting algorithms, I used a randomize data list. Because the numbers were generated with no specific pattern it was entirely possible that some of the numbers were in fact in a sorted order.

IV. Trade Offs Of Sorting Algorithms

Some algorithms are definitely more efficient at sorting under certain conditions than others, as was seen in the trial I ran with my own sorting algorithms earlier.

Here are the algorithm and some of their advantages:

- Bubble Sort: very simplistic and easy to memorize
- Selection Sort: performs very well on a small data set
- Insertion Sort: very good for presorted data
- Quick Sort: usually a very fast run time and also an in-place algorithm (meaning no additional storage used up)

Every sorting algorithm is imperfect, however. Here are a list of their disadvantages:

- Bubble Sort: not very fast or efficient
- Selection Sort: still relatively slow
- Insertion Sort: less efficient on larger data sets
- Quick Sort: more complicated and takes more CPU power to run

V. Impact of Programming Language

C++ was a great programming language to write these algorithms in because it has many features that other programming languages do not. For example, Java does not use pointers as efficiently. This is important because I dynamically allocated a lot of memory while writing my algorithms. Also C++ is a compiled language which makes it faster. Programming languages such as python are interpreted, which means they do not have a compiler.

VI. Shortcomings of Empirical Analysis

Empirical Analysis is not the most efficient type of analysis. It is not time or cost effective. The reason behind this is the fact that it relies on so many different variables. This type of analysis can easily be skewed if the platforms, hardware, or compilers are not exactly the same. Mathematically analysis on the other hand is time and resource efficient. However, mathematical analysis can be a little unrealistic. A good example is the tests on my algorithms, mathematically the Quick sort function should have been the fastest, but that was not the case. Empirical analysis does have its upsides: sometimes it shows us more of the real world than mathematical.

VII. Citations

- https://www.geeksforgeeks.org/is-there-any-concept-of-pointers-in-java/
- https://link.springer.com/chapter/10.1007/978-981-10-8848-39
- https://www.quora.com/What-are-the-advantages-and-disadvantages-to-an-insertion-sort

]