CA284

Sorting algorithm project

Name: Gary Murphy

Student number: 19310381

Introduction

For this project I was required to implement three different sorting algorithms in the C programming language and test them against various datasets of different size. For this task there are many suitable algorithms, some faster than others. Some have simple code, and some require a more complex implementation.

For this project I chose to test the following three algorithms, insertion sort, quick-sort and heapsort. I chose these three algorithms as they use different sorting techniques and they also include varying levels of sorting speed and complexity. For example, insertion sort is rather simple algorithm that is very slow on large datasets whereas quick and heap sort are more complex but sort large datasets much faster.

Datasets

Another aspect of this program that is equally important as the algorithms themselves are the datasets. Essentially, the datasets are the numbers we task the algorithms with sorting. It is Important in an analysis such as this to use a variety of datasets to test the strengths and weaknesses of our algorithms.

For this project I chose to use the following three types of data sets. Randomised numbers sorted (low-high) and reverse sorted (high-low). I chose these datasets for the following reasons:

* Randomised numbers provide an unbiased and realistic view on the performance of the algorithms.
* Sorted(low-high) to see how the algorithms perform when their job is already done. This is particularly useful for detecting unnecessary comparisons in an algorithm. i.e. insertion sort, selection sort
* Reverse sorted(high-low) to see performance when dealing with a backwards dataset. This can hamper the performance of a given algorithm.

Negatives

With any project there are negatives. Here are some aspects of my implementation and analysis which I believe could be improved upon.

* Experiment with more unique datasets. While the three datasets chosen provided ample evidence, I believe the use of more unique datasets such as a partial sort, or array of a single repeating number would have provided a broader insight into the performance of my algorithms.
* Insertion sort efficiency. While I understand that insertion is a O(n^2) algorithm, in the future I should attempt to make my implementation more efficient as it was taking upwards of a day to sort 5,000,000 integers.