Sorting Report

Ori Chanael
Computer Science
Chapman University
Orange, CA
chana102@mail.chapman.edu

***Abstract*—This document is a report on Assignment 6 using LaTeX. This covers the time difference, tradeoffs, and programming language for the algorithms implemented and the shortcomings of empirical analysis.**

***Index Terms*—algorithm, sorting, programming**

I. Introduction

This document is a comparison of sorting algorithms Quick Sort, Insertion Sort, Gnome Sort, and Merge Sort.

II. Time Differences

(Testing was done twenty times with a text file of 10,000 values)

A. Quick Sort

Quick Sort lived up to its name as the fastest of the four algorithms. Generally consistent, it typically had a runtime of 0.096ms. It's fastest runtime was 0.085ms and the longest runtime was 0.125ms.

B. Insertion Sort

Insertion Sort was extremely consistent, and was the third fastest out of the four. It typically had a runtime of 134.2ms, with the slowest being 136.3 and the fastest being 133.7.

C. Gnome Sort

Gnome sort was predictably the slowest of all the algorithms, along with being the most inconsistent. It averaged a runtime of 520.2ms, with its fastest runtime being 504.3ms and its slowest runtime being 574.3ms.

D. Merge Sort

Merge Sort was the second fastest of the algorithms and very consistent, typically having a runtime of 3.16ms, with its fastest runtime being 3.09ms and its slowest being 3.41ms.

III. Tradeoffs

A. Quick Sort

1) Pros: Sheer speed is the benefit of Quick Sort, even its slowest times were faster than all the other algorithms.

2) Cons: Quick Sort was fairly difficult to implement, needing to be implemented recursively with the partition and swap methods. However once the first stumbling blocks were cleared it became much easier to understand.

B. Insertion Sort

1) Pros: A very straightforward algorithm, Insertion Sort was relatively easy to implement. With a smaller number values, it can work almost as fast as Quick Sort.

2) Cons: Slows down significantly when having to sort more than a few hundred values. With 10,000 values it was the third fastest.

C. Gnome Sort

1) Pros: Extremely easy to implement, requiring the least work of all algorithms.

2) Cons: By far the slowest of all algorithms, earning its nickname of 'Stupid Sort.' Slows down significantly when having to sort more than a hundred values.

D. Merge Sort

1) Pros: The recursion was similar to Quick Sort, so having done that first Merge Sort was easier to understand. It is the most consistent of all the algorithms, and the second fastest, vastly outstripping Gnome and Insertion Sort but falling short of Quick Sort.

2) Cons: The merge function was definitely the most confusing and time consuming function to write out of all the algorithms. Not quite worth it compared to the simpler and faster Quick Sort.

IV. Empirical Analysis

Empirical Analysis is useful for running multiple algorithms with the same input. The issue is it uses a significant amount of resources and as a result made the runtimes differ from test to test. It also puts a higher strain on RAM and CPU usage.