Assignment 6 Report

David Ulriksen

12/14/18

A Empirical Analysis on the runtimes of various Sorting Algorithms

1 Time Differences

1.1 Initial Thoughts

I started the project expecting there to be far less difference between what became the slowest and fastest sorting algorithm. I had expected that, perhaps, quick sort would be a third of the length of bubble sort.

1.2 After Thoughts

Table 1: Average Time Results

Bubble	37
Selection	26
Insertion	5
Comb	1
Quick	1

The time differences were so drastic I had believed that there was something wrong with my code. I had to keep bumping up the amount of data for quicksort to even take a second. However, with bubblesort, I had to keep the amount of data low so I could get a result without waiting for too long and thinking my code was mentally struggling.

2 Tradeoffs

There are of course tradeoffs with regards to sorting algorithms. Time complexity vs space complexity vs implementation complexity. Obviously, faster sorting times and algorithms are preferred, but there may be restrictions that depend on the system and memory.

3 Language Choice

I wasnt aware we could pick other languages. I read the instructions before we started and figured it was leftover from a previous class that was allowed to pick different languages.

However, using c++ as the language allows for more control over member management, so long as one knows what they are doing. Not that I do. I used double linked lists in my algorithms, which I imagine true sorting algorithms dont do. Or at least use much more efficient ones. In using double linked lists, I probably lost all tradeoff of using the c++ language.

4 Shortcomings of the Empirical Analysis

These algorithm tests were analysed using Empirical Analysis. Each algorithm was developed and allowed time to process information. It took time to develop, implement, and prove what can be predicted via other means and with far less effort and cost.