**An Analysis of the Interactions between Programmer, Language, and Algorithm**

**Andrew Combs**

**Cody Jenkins**

**Department of Computer Engineering/Computer Science**

**University of Louisville**

**4-9-2018**

**An Analysis of the Interactions between Programmer, Language, and Algorithm**

Andrew Combs

Cody Jenkins

Department of Computer Engineering/Computer Science

University of Louisville

4-9-2018

**Executive summary**

The purpose of this experiment is to analyze the relationships between programmer, programming language, and algorithm. It was executed by having the insertion sort, selection sort, and bubble sort algorithms written in C#, Java, and Python by Cody and Andrew. The running time of each algorithm to sort a given list of floating-point numbers is the response.

**Table of Contents**

Executive Summary i

1. Introduction 1

2. Experiment Design 1

3. Experiment Results

4. Analysis

5. Results and Conclusion

References

**1. Introduction**

Sorting algorithms are a well-studied field, due to their common use in everyday applications. Because the runtime of an application is important to users, efficient algorithms are highly desired. In addition, the programming language an application is written in can also affect how quickly it runs. Compiled languages are generally faster than interpreted languages, due to their ability to modify the program before it is run to help enable it to run faster. Finally, it is worthwhile to explore the interaction between programmer and programming language. Of course, if a particular programmer has years of experience with a particular language, he or she will be well-versed in the various nuances of that language. Thus, it is the goal of this experiment to analyze the relationships and effects of programming language, algorithm, and programmer on the execution time of a given problem.

**2. Experiment Design**

For this experiment, the students decided to analyze the effects of different programming languages, different algorithms, and different programmers on program runtime. To evaluate this experiment, they decided to have two programmers implement three different sorting algorithms in three different languages. The three algorithms implemented are:

* Insertion sort
* Selection sort
* Bubble sort

The three languages used are:

* C#
* Java
* Python

The two programmers are:

* Andrew Combs
* Cody Jenkins

In order to analyze the relationship between the different factors, the students employed a full factorial design. Programs implementing each of the algorithms in each of the languages were written by both programmers, and their execution time was measured. Each program read in a file of 10,000 floating-point numbers into a list every time it runs, and sorts the list using one of the sorting algorithms. To minimize the effect of the environment in which the programs are running, the students elected to run them all on the same machine. The time each program took to sort the given set of numbers, in milliseconds, is the response. A separate program was written that would call the other programs and track their execution times.

**3. Experiment Results**

**4. Analysis**

**5. Results and Conclusion**

The runtime of various algorithms are well-defined and have been studied frequently. Divide-and-conquer search algorithms, which divide the input into smaller pieces and recombine them later, perform much better than comparison-based search algorithms. The three algorithms were chosen because they all had polynomial time complexity.

As was expected, python took much longer to run than both C# and Java. This is due to the fact that Python is interpreted, whereas both C# and Java are compiled. Compiled languages benefit from utilizing compiler optimizations, which help make the code run faster. However, what was not expected was that Java was faster than C#. This could be due to the fact that C# relies on the common-language runtime and just-in-time compiling to increase its flexibility as part of the .Net framework.

**References**

Algorithm definitions were obtained from:

Selection sort. (2018, April 09). Retrieved April 9, 2018, from <https://en.wikipedia.org/wiki/Selection_sort>

Insertion sort. (2018, April 09). Retrieved April 9, 2018, from <https://en.wikipedia.org/wiki/Insertion_sort>

Bubble sort. (2018, April 09). Retrieved April 9, 2018, from <https://en.wikipedia.org/wiki/Bubble_sort>