CAB302 Assignment 2

Inventory Management Application

Student Name: Luke Reynolds, Johnathan Gonzalez

Student Number: n18023481, n9821112

Date Submitted:

# Summary

This report documents the functionality, theoretical predictions, and experimental results of a simple binary searching algorithm. The algorithm was implements using a Java Integrated Development Environment which was used to record the number of basic operations performed by the algorithm and its execution time. Once this was achieved the program was then used to create a graph displaying the data in an easy to view format.

Contents

[Summary 1](#_Toc511563205)

[1. Description of the Algorithm 3](#_Toc511563206)

[2. Theoretical Analysis of the Algorithm 3](#_Toc511563207)

[2.1 Algorithms Basic Operation 3](#_Toc511563208)

[2.2 Best Case Efficiency 3](#_Toc511563209)

[2.3 Worst Case Efficiency 3](#_Toc511563210)

[2.4 Average Case Efficiency 4](#_Toc511563211)

[3. Methodology, Tools, & Techniques 4](#_Toc511563212)

[3.1 Methodology 4](#_Toc511563213)

[3.2 Tools 4](#_Toc511563214)

[3.3 Techniques 4](#_Toc511563215)

[4. Experimental Results 4](#_Toc511563216)

[4.1 Functional Testing 4](#_Toc511563217)

[4.2 Best and Worst Number of Basic Operations 5](#_Toc511563218)

[4.3 Average Number of Basic Operations 5](#_Toc511563219)

[4.4 Average Execution Time 5](#_Toc511563220)

[References 5](#_Toc511563221)

[Figure 1. Algorithm to be analysed. 6](#_Toc511563222)

[Figure 2. Graph displaying number of basic operations relative to dataset size. 7](#_Toc511563223)

[Figure 3. Graph displaying execution times relative to dataset size. (No warm up) 8](#_Toc511563224)

[Figure 4. Graph displaying execution times relative to dataset size. (With warm up) 9](#_Toc511563225)

[Appendix A: The Algorithm (Code) 10](#_Toc511563226)

[Appendix B: Functionality Testing (Code) 11](#_Toc511563227)

[Appendix C: Counting Basic Operations (Code) 12](#_Toc511563228)

[Appendix D: Averaging Basic Operations (Code) 13](#_Toc511563229)

[Appendix E: Measuring Execution Times (Code) 14](#_Toc511563230)

# Technical Description

A technical description of your program architecture, drawing reference to object-oriented design concepts such as polymorphism and abstraction. You may want to use a diagram to illustrate your type hierarchies and interaction between classes.

# 1.1 Program Architecture

The most important basic operation in the binary search seen in figure 1 is the while loop that is used to check if the current value is equal to the search value. This means the best way to calculate the basic operations is a test that would require a variable that counts the number of loops required by the algorithm to complete the search. The algorithm works by splitting the sorted array down the middle meaning that the difference in number of basic operations is insignificant even for large arrays.

# 1.2 Type Hierarchies

The best-case efficiency scenario for this algorithm is when the current index matches the search value on the first iteration. This means that either the list is empty or the search value was found on the first iteration. In either of these cases the algorithm only requires one loop of the outermost while loop to complete its search. This can be expressed as .

# 1.3 Class Interaction

The worst-case efficiency scenario for this algorithm is when the array does not contain the search value. This causes the algorithm to execute the maximum number of basic operations for the array size until the left-most value is higher than the right-most value meaning that the search value is not in the array causing the algorithm to exit. This can be expressed as .

# 3. Graphics User Interface (GUI) Test Report

This section summarises the development environment used for this report to test the algorithm.

# Image result for image placeholder 3.1 Home Page

The algorithm and experiments were implements using a Java IDE called IntelliJ. IntelliJ Community is a free and open-source IDE.

# Image result for image placeholder3.2 Item List

- The experiments were performed on a custom-built desktop pc running windows 10.

- A built in random number generator was used to fill the datasets with variables.

- System.nanoTime() was used to calculate the execution times.

# Image result for image placeholder 3.3 Sales Logs

Graphs displaying the data were produced using a free and opensource Java library called JFreeChart. This library worked via storing all the results from the experiments in a dataset which is used by the library to plot the data.

# Image result for image placeholder 3.3 Cargo Manifest

Graphs displaying the data were produced using a free and opensource Java library called JFreeChart. This library worked via storing all the results from the experiments in a dataset which is used by the library to plot the data.

# References

* <https://docs.oracle.com/javase/tutorial/uiswing/components/table.html#simple>
* <https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>
* <https://stackoverflow.com/questions/20473325/gridlayout-java-center-alignment>