Antonio E. Ramirez

8 March 2020

Project 2

CMSC 451

Table of Contents

[Introduction 3](#_Toc34586865)

[Pseudocode 3](#_Toc34586866)

[Big-Θ Analysis 4](#_Toc34586867)

[Iterative 4](#_Toc34586868)

[Recursive 4](#_Toc34586869)

[JVM Warm-up 4](#_Toc34586870)

[Critical Operation 4](#_Toc34586871)

[Results Analysis 5](#_Toc34586872)

[Graph 5](#_Toc34586873)

[Comparison of Performance 5](#_Toc34586874)

[Comparison of Critical Operation Results 5](#_Toc34586875)

[Coefficient of Variance 6](#_Toc34586876)

[Big-Θ Comparison 6](#_Toc34586877)

[Conclusion 6](#_Toc34586878)

# Introduction

For my project I decided to use the Insertion Sort Algorithm. It is a simple sorting algorithm that works one item at a time. This algorithm is only efficient on smaller data sets. This is due to the fact the algorithm only works one item at a time. Some better algorithms for larger data sets are quicksort, heapsort, or merge sort.

## Pseudocode

The pseudocode for the iterative approach is as follows:

for i = 1 to data.length

x = data [i]

for j = i – 1 to j >= 0 and data [j] > x

data [j+1] = data [j]

end

data [j+1] = x

end

The pseudocode for the recursive approach is as follows:

If n > 0

recursive(data, n-1)

x = data [n]

for j = n-1 to j >= 0 and data[j] > x

data[j+1] = data[j]

end

data[j+1] = x

end

## Big-Θ Analysis

### Iterative

Within the iterative pseudocode we can see that there are two for loops that will affect the Big-Θ of the algorithm. Beyond this, there is nothing else that will affect the analysis. For the iterative application, the Big-Θ will be Θ(n^2).

### Recursive

For the recursive approach, there are two factors that will affect the Big-Θ analysis. The first is the recursive call of the function. The second will be the for loop within the function. This ends up being the same Big-Θ as the iterative approach. The Big-Θ of the recursive approach is Θ(n^2).

## JVM Warm-up

To warm-up the JVM, multiple instances of the program are run, 250 to be exact. These runs were done to properly warm-up the JVM and remove that variable from the data. 250 was chosen because is enough repetition to establish a solid baseline, but not so many repetitions to affect the overall runtime of the program too greatly. Anything more, the program would take a considerable amount of time to execute, due to warming the JVM up, and less would not be enough to properly warm-up the JVM. In other instances, I have seen people use 100 or even 1000 runs to warm the JVM up. There are advantages and disadvantages to both.

## Critical Operation

For this project I decided that the critical operation would be when an element was shifted. This was chosen as the critical operation because of the overall impact on the algorithm itself. If no shift was need, what was the impact to our time or space complexity? I determined that if no shift was needed then the impact was so minimal that it should not be counted. However, if a shift was required, that is proof of work for our algorithm, and should therefore be counted. This applies to both the iterative and recursive approaches.

# Results Analysis

## Graph

The x-axis represents the data size

The y-axis represents the average critical operation count

## Comparison of Performance

When looking at the graph, we can see that the two approaches are highly comparable. When we pair this with the assessed Big-Θ of both the iterative and recursive approaches, this is to be expected. Both take similar approaches to the critical operations and how they are counted. This would cause the graph to be very similar. Furthermore, both applications were assessed to have the same Big-Θ of Θ(n^2). Within the given data set sizes, both the iterative and recursive are highly comparable, if not the same.

## Comparison of Critical Operation Results

This ties into the precious section that covered the performance of the two applications of the algorithm. Overall, it was concluded that both applications grow at an exponential and comparable rate. The same goes for the execution time. Both the critical operation count and the execution time grows at an exponential rate. From these data sizes we can assume that this will be the case for any data size. This means that there is no real preference regarding an iterative or recursive approach and the application of the Insertion Sort.

## Coefficient of Variance

Coefficient of Variance, also known as the relative standard of deviation, is the ratio of the standard deviation to the mean of a given data set. This is many times expressed as a percentage. As it applies to our application of the insertion sort algorithm, we can see that this percentage became smaller as the data set grew. This fits the mold of basic data structures and statistics. With smaller data sets, the probability for outliers is much higher. With a larger “sample size”, we can establish a stronger benchmark, or mean, and inherently have a much smaller coefficient of variance. From this, it is reasonable to assume that smaller data sets are much more volatile with smaller data sets, and much more consistent with larger data sets.

## Big-Θ Comparison

Given the previous analysis of the Big-Θ of both the recursive and iterative applications of the insertion sort algorithm, we concluded that both are Θ(n^2). This is supported by the data from multiple runs of the program and the graph included. Both applications grow comparably in both critical operation count and execution time. This only applies to the current applications and how the critical operations were counted. This could change if the critical operations were determined to be something different, or a more efficient implementation of the algorithms was used.

# Conclusion

Insertion sort is an algorithm that is efficient when dealing with smaller data sets. As previously mentioned, when dealing with larger data sets there are more efficient algorithms such as the quicksort, heapsort, and merge sort algorithms. With regards to the insertion sort algorithm, both the iterative and the recursive approach were comparable in the execution time and critical operation count. We have also concluded that with extremely small data sets, the coefficient of variance is unnaturally high. This is due to the highly volatile data and the inability to establish a reliable mean. Regardless, both approaches remain relatively efficient and should be used with these small data sets.