CS303 Lab 2 – Insertion Sort PDF

Problem Specification:

The goal is to take in multiple input files ranging from 100 to 500000 random numbers and use insertion sort to sort each file and record the time that it takes to sort each array. All numbers are assumed to be integers greater than 0. This assignment differs from the previous one as we are implementing a sorting algorithm while in the previous assignment, we were implementing searching algorithms.

Program Design:

This program solely requires two files, insertionSort.java and sortDriver.java. Insertion sort has the source code for only the sorting function that sorts an array using the insertion sort algorithm. The sortDriver file reads in every input file and turns it into an array of n integers, n being the number of numbers in every file. Then it makes a call to the insertionSort file to sort each array. Using the function system.nanoTime() we calculate the time it takes for each array to sort.

Testing Plan:

The plan for testing is to take each array that has all the numbers from each input file and to sort them. We start by defining file variables that hold all the data of numbers. Then use a scanner to read each number in and placing it into an array of strings. Turning each input into the string array into an integer. Afterwards, you use this new array of integers and call the insertion sort on it. Using System.nanotime() before and after the call we are able to figure out the time that it takes.

Test Cases:

A picture containing text, scoreboard, plaque

Description automatically generated

The image above shows results of the insertion sort and how long each sort took.

|  |  |
| --- | --- |
| Array Size | Time(ns) |
| 100 | 15700 |
| 1000 | 75800 |
| 5000 | 321401 |
| 10000 | 631501 |
| 50000 | 2915000 |
| 100000 | 1516200 |
| 500000 | 5961101 |

Each case takes up more time than the previous except for 50000 and 100000. I do not know why these two are an exception to the case. There are multiple reasons why it could have happened, but I will discuss this in my analysis.

Analysis/Conclusion:

As discussed above, it seems as though the insertion sort works perfectly. The only anomaly is the time that it took for 50000 and 100000 numbers took to sort. There are two main ways that this could have caused the results that it did. One of which is that the 100000 number file is more sorted in comparison to the 50000 number file. It could also be the opposite being that the 50000 file is just more unsorted or closer to being ordered in largest to smallest causing more switching of numbers. Another reason could be that there is some system lack of resources during the run of the program. I believe that this is false since I tested multiple times causing it to show that there is a difference like this every time it is run which helps in showing that my first idea is more likely.

References:

I did not use any outside resources; all the code came from my own creation.

A screenshot of a computer

Description automatically generated with medium confidenceBelow are the pictures of my code.

Text

Description automatically generated

Text

Description automatically generatedText

Description automatically generated