Week 7 Lab Report

A screenshot of a cell phone

Description automatically generatedTime Complexity Analysis

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedHelper Method

Cases

\*Based on SortOfSortTester.java\*

Best Case: I concluded that my program works best when the input is an even amount non-repeating numbers, based on Test Case #3 which had a runtime of 4 ms which was the lowest of all the test cases.

Average Case: The average case is when the input is even or odd amount of numbers with repeating and non-repeating numbers, based on Test Cases #2,4, and 5 which had runtimes of either 5 or 6 ms, which was not the lowest but average runtime of all the other tests.

Worst Case: The worst case is when the input is even amount of positive integers, based on Test Cases #1, which had runtimes of 34ms, which was the highest of all test cases.

Answer Justification

To answer the sortOfSort question, I used the logic behind a bubble sort, and iterating through the array and finding the Max number and replacing it at the desired position, but reducing the scope of search by one each time, for example if the max number was to be stored on the right side of the array, I would reduce the scope of search by one so that in the next iteration it would only search the array but exclude the last index where the largest number is already stored. I also used a Boolean and a counter to help determine the position where the variable would be stored i.e. the front or end of the array. I decided to use helper methods instead of one big method because when calculating the time complexity, I found it easier to do it in parts, instead of one big calculation.