PA1 CPTS 233 Jocelyn Strmec

A: *Problem statement*. In one or two sentences, summarize the goal of the problem.

The goal of this project was to sort a given list of integers into a linked list. This linked list would be sorted at all times, have accessible min, max, and median, and be return a time for each method or retrival time_insert, time_min ,time_max ,time_med to occur.

B: *Algorithm design*. This project does not have a lot of algorithm design decisions. However, there are few areas that you will need to make a decision about how to compute the output values such as 'min', 'max', and 'med'. Within one page, provide a brief description of the main ideas behind your algorithms for finding 'min', 'max', and 'med' within the list. Discuss any potential alternative approaches that could be used to find these values which could have resulted in a poor timing performance.

Finding min would be easy whatever the 0 index was that would be or getFirst() or getLast(). Despite knowing this it was decided it would be preferred to sort each number as it was added to the linked list, find the greatest and lowest values as the list was read, updating a global variable. The max value was used to get the lastIndexOf() so I could divide that index by 2 to find the median. Because of the global variables calling the max and min are at a zero millisecond execution time and about 3911 nano and 666 nano seconds, though it might add a bit of run time to the adding of the list as the global variable is updated every time a higher or lower variable are found. Med is a bit longer in nano seconds but milliseconds it is 0 because lastIndexOf (max) is used within the get() function to set the median.

C: *Experimental setup*. In this section, you should provide a description of your experiment setup, which includes but is not limited to:

Machine specification - CPU maker, CPU speed, RAM available, hard drive type How many times did you repeat each experiment before reporting the final timing statistics?

MacBook Pro (16-inch, 2019) -Apple-2.3 GHz 8-Core Intel Core i9-920.49 GB available-1TB SSD

Well for an initial run the statistics are only run once, there is an option to run the program as many times as wanted and an array list will add the data and can return an average for the number of runs.

O/S and environment used during testing: Windows or Unix? Also mention which compiler environment (e.g., javac version). This information will help the TAs determine where to run your programs during grading.

macOS Catalina Version 10.15.6 and eclipse was used for a compiler and run environment.

D: *Experimental Results & Discussion*. In this section, you should report the performance (running time) and outputs based on your observations, and provide justification for your observations. For your testing and reporting, conduct the following two experiments using the two provided input files, namely 'input1.txt' and 'input2.txt'.

Input1.txt	milliseconds	milliseconds	milliseconds	milliseconds	milliseconds	Average
time_insert	22	21	31	43	50	33
time_min	0	0	0	0	0	0
time_med	0	0	0	0	0	0
time_max	0	0	0	0	0	0
Input2.txt						
time_insert	1319509	1267914	-	-	-	1293711 Or 21.56min
time_min	0	0	0	0	0	0
time_med	10	9	0	0	0	10
time_max	0	0	0	0	0	0

Also I used the following to know how to time the time for a method to run because I have not yet had to : https://www.techiedelight.com/measure-elapsed-time-execution-time-java/#:~:text=Java%20System%20class%20also%20provides,elapsed%20time%20of%20the%20program. 1.System.nanotime()

```
import java.util.concurrent.TimeUnit;

// Program to measure elapsed time in Java
class Main
{
        public static void main(String[] args) throws InterruptedException {

                long startTime = System.nanoTime();
                /* ... the code being measured starts ... */

                // sleep for 5 seconds
                TimeUnit.SECONDS.sleep(5);

                /* ... the code being measured ends ... */

                long endTime = System.nanoTime();

                // get difference of two nanoTime values
                long timeElapsed = endTime - startTime;

                System.out.println("Execution time in nanoseconds  : " + timeElapsed);

                System.out.println("Execution time in milliseconds : " +
                                                        timeElapsed / 1000000);
        }
}
```