Allison Stansberry
CPT_S 233
Nadra Guizani
Programming Assignment 1
09/16/2020

A: Problem Statement

The goal of the problem is to be familiar with the process of benchmarking programs. In doing so, it is important to record results and critically analyze those results to justify them and determine if there is a better, more efficient way to complete the task.

B: Algorithm Design

My program is written based on the understanding that the smallest value should be at the front of the LinkedList, “dataset”, while the largest value should be at the end. Because of this knowledge, it made the most sense to me that in order to find the smallest or largest value in the list, I would just have to get the data from the first node in the list and the last node in the list respectively. As for the median, I used the same math to find the middle of the list as I had in my insertion algorithm, and used that to get to the middle index of the linked list which is the median.

Another way of doing this could have been to include max, min, and median integer variables that can be accessed by the whole class. With these, during every insertion, you could check to see if the new number to be added was less than the current min or larger than the current max and replace the max or min accordingly. This could have potentially made the calls for the max, min, and median values since the list would not have to be traversed to find them. However, the reason I chose not to do this was because it would add another few steps to the insertion algorithm making the time it takes to add all the numbers to the list longer. Because of the nature of a sorted list, you know how the list is organized so you know where the maximum value and minimum and median values are already.

C: Experimental Setup

Machine Specification:

- CPU: AMD Ryzen 5 3600 6-Core 3.95 GHz
- RAM: 16 GB
- Hard Drive Type: SSD

Times Experiment (input1.txt) Was Ran:

- Before Working Version: ≈ 10 times
- After Working Version: 9 times

O/S and Environment Used During Testing:

- Windows 10
- IDE: Visual Studio Code w/ javac 14.0.2
- Compiling done inside Git Bash terminal

D: Experimental Results & Discussion (input1.txt)

Insert_time: 0.0692 to 0.0719 seconds (average: 0.0702)

Time_min: 0.0020 to 0.0026 seconds (average: 0.0026)

Time_max: 0.0001 to 0.0002 seconds (average: 0.0002)

Time_median: 0.0002 to 0.0003 seconds (average: 0.0002)

run #	time_insert	time_min	time_max	time_median
1	0.0703769	0.0023396	1.51E-04	1.57E-04
2	0.0698249	0.0021685	1.64E-04	1.60E-04
3	0.0707396	0.0021473	2.39E-04	1.89E-04
4	0.0691759	0.002037	1.44E-04	2.73E-04
5	0.0692044	0.0024311	1.71E-04	2.10E-04
6	0.0703378	0.00212	1.55E-04	1.65E-04
7	0.0719539	0.0023981	1.71E-04	2.19E-04
8	0.0706659	0.0023793	1.58E-04	1.64E-04
9	0.070237	0.0025539	1.73E-04	2.03E-04
Average	0.07027958889	0.002286088889	1.70E-04	1.93E-04

I desperately tried to make it run fast enough to do input2.txt but it just took way too long. To the point I contemplated scrapping my whole program and starting over again (It is still trying to run as I am turning this in).

I think the time varies at all because different things are happening in the computer at different instances. I have other applications open on my computer while running this program so things could simply take slightly more time because less of the computer's power is being directed to it at one instance than it would be at another. Something I was curious about however, was that the time to find the minimum was more time than finding the maximum values. I would have assumed that traversing through the list would have taken more time than just grabbing the first node's value. I would think that finding the minimum value would actually take the least amount of time since it is the first node in the list.