Eric Furukawa

CPTS 223

Section 01

Programming Assignment 1 Report

A. In this assignment the goal was to create a program which would read in integers from a user specified file and then sort and place them into a sorted singly linked list; this data would be analyzed so that the maximum, minimum and median values of the file could be printed back to the user. The goal of this problem was more directed at familiarizing ourselves to the new coding and submission methods for the course.

B. MIN- The minimum was the easiest of the three to find. As the list would already be sorted when looking for this value, it is possible to grab the data registered to the pHead of the linked list. This would be extremely efficient compared to any other method as it always has theta(1) efficiency. POOR METHOD: The min could be found by iterating through the list and finding which value was the lowest by comparison (tracking the min and comparing to it).

MAX- The maximum was found by simply iterating through the linked list using a pointer to a node as an iterator. It would stop on the last in the sorted list and return its data value. Since the list is singly linked and not doubly, this was the seemingly most efficient was to do this w/o pre-calculating it when populating the list as it has theta(n) efficiency. POOR METHOD: The max could be found by iterating through the list and finding which value was the highest by comparison (tracking the max and comparing to it).

MED- The median was found by calculating the length of the list and then iterating half way through the list. If the list had an odd amount of values, only the middle value would be taken (as it is the median). In cases which there was an even amount of values, the two middle values would be averaged for the median. Since the length would be calculated by iterating and the values found by iterating half way, the efficiency for the algorithm is O(n) (O(n+n)) which makes it the least efficient. POOR METHOD: The median in an even length list could be found by determining the size of the linked list and if it was even or odd, then if the list was even, averaging every pair starting from the 1st position to the length – 1 position and then iterating to the length divided by 4.

C.

PC Specs:

Type: Laptop

OS: Windows 10 Home

Processor: Intel (R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHZ

RAM: 12.0 GB (11.9 usable)

System Type: 54 bit OS, X64 based processor

Condition: About 3 Yrs old (heavy use)

Compiled and tested using PuTTY(linux) g++

Timings for input1.txt were computed 3 times.

Timings for input2.txt was attempted 5 times (each aborted due to length).

D.

**For input1.txt**

**Trial 1:**

Inserted in 19 ms

Max is 4000 in 27 µ

Min is 1 int 5 µ

Median is 2058.5 in 79 µ

**Trial 2:**

Inserted in 6 ms

Max is 4000 in 40 µ

Min is 1 int 8 µ

Median is 2058.5 in 133 µ

**Trial 3:**

Inserted in 3 ms

Max is 4000 in 22 µ

Min is 1 int 4 µ

Median is 2058.5 in 59 µ

Insertion takes the most time as it must iterate over and over to find the placement of each data. Median comes next as it must calculate the length of the list (which requires iteration) and then calculate the midpoint and iterate to it. Max is the second fastest as it only requires one iteration. Min is the fastest as it only needs to return the head’s data. 1000 elements is a fairly small amount for the given algorithms to handle, resulting in very fast iterations and short times. The variation in the times could come from many variables such as other programs taking up computation power.

**For input2.txt**

For input2.txt, the algorithms for inserting and analyzing the data could not perform the operations in a reasonable time. When tested, the program reached insertion of about 400000 elements in 2 hours time. Thus, the program was stopped as it would not be able to complete all operations in a reasonable time. Since insertion takes iterating through the list over and over, the time it takes to insert would only grow as more were inserted in the list. 1000000000 elements display the scaling nature of the algorithm’s efficiency (as it becomes much longer to complete).