## Problem Statement:

In this project, the goal was to implement a few algorithms to sort a set of data and find the statistics, min, max, and med while benchmarking the amount of time it took to run each algorithm. Furthermore, the goal is to learn to use benchmarking to analyse a program's performance rather than using other analysis software.

## Algorithm Design:

### Sorting:

First, to sort the list of numbers quickly and efficiently before insertion into the linked list, I placed all the numbers from the file into a TreeSet which orders them from least to greatest. Then, I traversed the set with an iterator while inserting each number into the linked list in the same order it came out of the TreeSet. Originally, when the term "sorting" came to mind I thought about using a HashSet as it is known to be one of the fastest sorting methods. I later chose to use a TreeSet instead because, although slightly slower, I would be able to insert all the values into the linked list in increasing, numerical order rather than HashSet order which may become pointless when inserted into a linked list. Overall, I chose to use a data structure to sort the values because traversing the linked list thousands of times in order to place each number in it's ordered index seemed far too time consuming when other methods were available.

### Min:

Calculating the minimum value was quite simple with the values ordered from least to greatest in the linked list. I simply took the data value from the first node with "getFirst()" as the minimum value. Traversing the linked list did not make sense to me when I know that the minimum value is at the head of the linked list given an increasing set of numbers.

### Max:

Calculating the max value was very similar to the min. As the numbers were increasing, I knew that the max is the last node in the linked list so traversing the list seemed unnecessary. I simply used the function "getLast()" to retrieve the data from the last node in the list.

### Median:

Calculating the median was not as simple as the min and max. Knowing the values were already sorted in increasing order, I knew that I had to simply retrieve the middle number(s). Again, I avoided traversing the list knowing that traversing 350,000 numbers is not very efficient if there is a way to avoid it, To find the middle indexes,' I found the size of the treeSet and divided it by 2. Then I added 1 if the size was even to get both middle indexes. Once I had the middle indexes for odd and even sizes, I could simply retrieve the median at the index or find the median from the average of the two middle indexes. This way I didn't have to traverse the linkedList, I simply calculated values.

## Experimental Setup:

**CPU maker:** Intel
**CPU speed:** 1.8Ghzb
**RAM available:** 16GB
**Hard drive type:** SSD
**O/S:** Windows
**Environment:** Pullman Apartment Wifi
**Compiler environment:** javac version
**Number of experiment trials:** 5

## Experimental Results and Discussion:

### Average Results with 3 trials for input2:

**Min Value:** 39
**Max Value:** 7999978
**Med Value**: 4004476.5

**Min calculation time:** 13940.0ns
**Max calculation time:** 1300.0ns
**Med calculation time:** 3115180 ns
**Sorting and inserting time:** 1135.29ms (1135290040ns)

**Total Time**: 1153.92ms (1153921800ns)

### Average Results with 3 trials for input1:

**Min Value:** 1
**Max Value:** 4000
**Med Value:** 2058.5

**Min calculation time:** 6066.66ns
**Max calculation time:** 1133.33ns
**Med calculation time:** 21100ns
**Sorting and inserting time:** 57.23ms (57230000ns)

**Total Time:** 55.5385ms (5.55385E7ns)

Within the results for both files, it is clear that most of the time is consumed by sorting the numbers into the TreeSet. In fact, the amount of time taken to calculate the min, max, and med is only a fraction of the time it takes to sort the numbers. This makes sense as to sort the numbers, the file has to be traversed many times in order to find the right spot for each number compared to simply retrieving numbers from indexes.The bigger file takes much longer to sort than the smaller one which is expected with this conclusion. What is interesting though is that in both experiments, it takes the program longer to retrieve data from the first index than from the last. I would expect the opposite as the first index can be identified right away.

## Bibliography


- Paul, Javin. *How to Read File in Java Using Scanner Example - Text Files*. www.java67.com/2012/11/how-to-read-file-in-java-using-scanner-example.html.
- Author not Listed. “TreeSet in Java.” *GeeksforGeeks*, 26 June 2020, www.geeksforgeeks.org/treeset-in-java-with-examples/.
- Author not Listed. “Java.util.LinkedList.get(), GetFirst(), GetLast() in Java.” *GeeksforGeeks*, 10 June 2020, www.geeksforgeeks.org/java-util-linkedlist-get-getfirst-getlast-java/.