Dan Ruley
u0956834
1/24/19
CS 2420

Assignment 2 Analysis

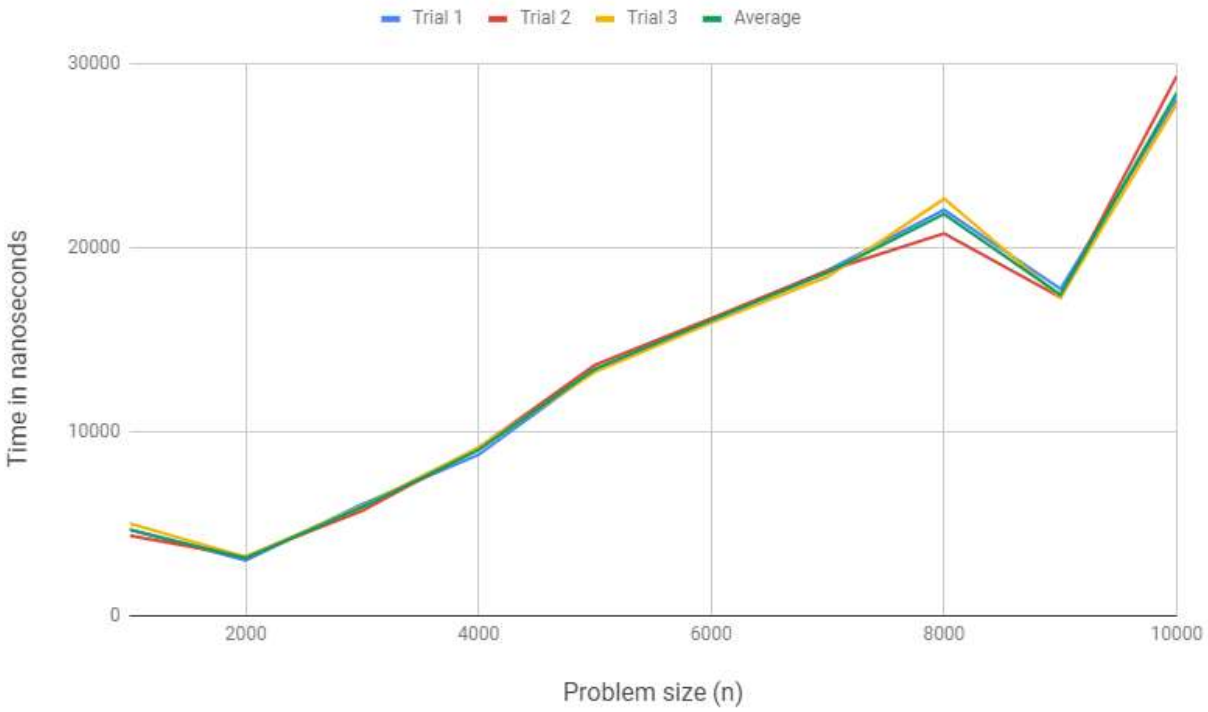
This assignment was a quite enlightening. We learned a large amount about how to use generics, and also became acquainted with the Comparator interface and the way it can be used to sort data using custom comparisons.

My partner and I seemed to work well together. I think we were fairly efficient, due to the fact that we sketched out a lot of things on paper and designed tests before implementing methods. Sketching things out on paper seems to be quite helpful and reduces the amount of coding time required. We seemed to complement each other pretty well, the navigator would often have good suggestions or spot errors and this saves quite a bit of time compared to programing by myself. I would work with Aric again on another assignment. I probably preferred being the "driver" a bit more, just because it feels like I'm doing more.

Comparable and Comparator are both interfaces that give you a way to sort data based on comparisons made between different items. Comparable not as adaptable as Comparator, because you are limited to the one compareTo method, whereas Comparator allows you to write many different compare functions for many different situations. I do not think that we would have been able to fulfill all of the specifications of the LibraryGeneric class without Comparator because we needed to be able to define multiple comparison functions, and Comparator is critical for this. One situation where Comparable is nice is for classes with already defined compareTo methods, such as String. We used the compareTo defined in the String and Gregorian Calendar classes within our Comparators, and this was really useful.

Runtime analysis showed the time increased more or less linearly with the problem size. I believe the lookup method has an O(N) runtime. Looking at the algorithm, it has only one for loop, so it's definitely not quadratic or greater. There is no halving or doubling either, so this would suggest that it is not logarithmic. It simply goes through the items one by one, so a linear O(N) runtime is the most likely here. Included on the next page is a chart of three trials, as well as the average of the three runs.

Figure 1:


Trial 1
Trial 2
Trial 3
Average
30000
20000
10000
0
Time in nanoseconds
2000
4000
6000
8000
10000
Problem size (n)