Assignment 2: Library

Analysis Document by Qianlang Chen (u1172983)

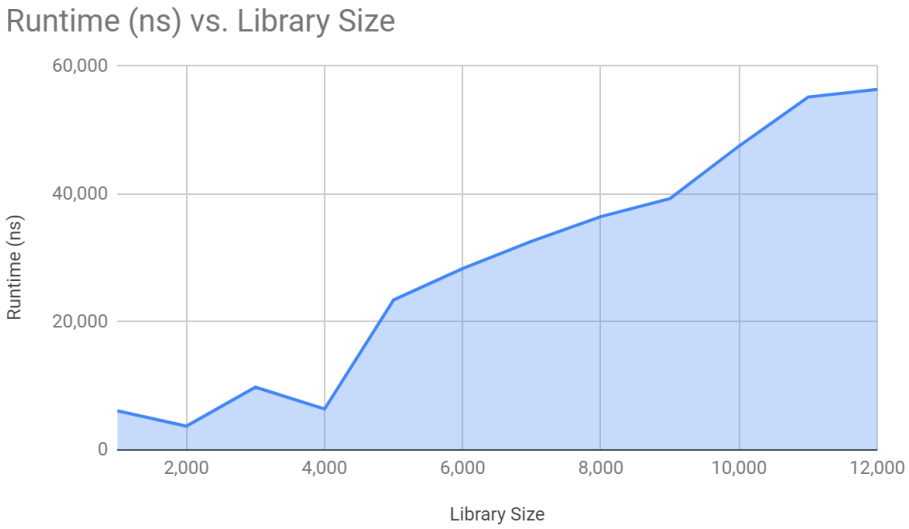
In this programming assignment, Brandon Walton was my coding partner, and I had been switching "roles" with him throughout the completion of this assignment. I enjoyed being in the navigator seat, because I had more chances to think about long-term strategies of writing our library, and that was a fun experience for me. I generally like being a navigator since I always think of myself as an experienced programmer so that I can usually point out errors that my partners make and can give them some tricks in coding.

Brandon was also an experienced programmer, to which I was happy that we did not have to worry about mistakes that most beginners make, and he could understand my navigations better. This made us working so efficiently that we finished the entire assignment (not including this analysis document) in under two hours and a half. I will definitely work with him in future assignments as long as I am allowed.

We encountered Java's *Comparable* and *Comparator* interfaces during this project and understood the difference with them. They are both interfaces, but the *Comparable* interface represents an object that has a *compareTo()* method so that the object can be compared to another object and tell the difference. A *Comparator*, on the other hand, has a *compare()* method that can be used to determine the order of two objects (which may not implement *Comparable*). In the case of our library sorting, we could have implemented our *LibraryBook* class to be *Comparable* but only for one specific sorting; so if we did that, we would still have to use *Comparators* for the rest of sorting (our library could be sorted with multiple ways, such as based on the books' ISBNs, titles, etc.).

To test and analyze our *lookup* methods within the *Library* class, I generated several different sized libraries, ran *lookup*, and timed them. Here is the diagram of the runtime for each library size:

Library Size	Runtime (ns)
1,000	6,109
2,000	3,706
3,000	9,800
4,000	6,386
5,000	23,450
6,000	28,364
7,000	32,621
8,000	36,484
9,000	39,300
10,000	47,527
11,000	55,181
12,000	56,389



The diagram shows that the Big-Oh behavior of our *lookup* method is **linear**, which makes sense, because how our lookup method (for ISBN) works is by going over the array of books **once** with a single for-loop, causing the runtime to increase proportionally to the size of the library. (Note: since the graph does not show a strong linear behavior for small library sizes, I increased the maximum test-case to contain 12,000 books just to see its behavior easier.)