Ryan Williams
Assignment 3 Analysis Doc
2-5-15

1. Lonnie Lund was my partner. And I (Ryan Williams) was the one who submitted the code.

2. We honestly didn't switch much, I did most of the work, and all the testing. I wish we had switched a bit more on this, yes.

3. This time around I was honestly not too happy with Lonnie's participation of this assignment There were a total of 4 days that I worked on this assignment (Mon. - Thurs.). I worked on it for a minimum of 5-6 hours a day. Out of those 4 days, lonnie worked on it once with me, yes he was a help at getting code typed up on my computer, but as mentioned before I did the majority of the testing which involved changing around a lot of code to get things working a bit better. For example I went through and did my best to comment, add java docs, make the code more efficient in certain areas by writing methods to handle tasks (i.e. comparatorBinarySearch() and comparableBinarySearch()).

4. Using a Java list instead of an array to back our SortedSet implementation would have had its ups and downs, in my opinion. But difficulty wise It don't think it would have been much harder. Simply because of all the functions that are built into array lists and such, such as get(), equals(), etc. Thinking about it now though, maybe it would have been easier to implement because we could have just called certain methods instead of having to keep track of whether the set is empty of not. For example, with an array you have to keep track of where you are, where with an array list you can just call list.indexOf() and it would tell you right where you were at.

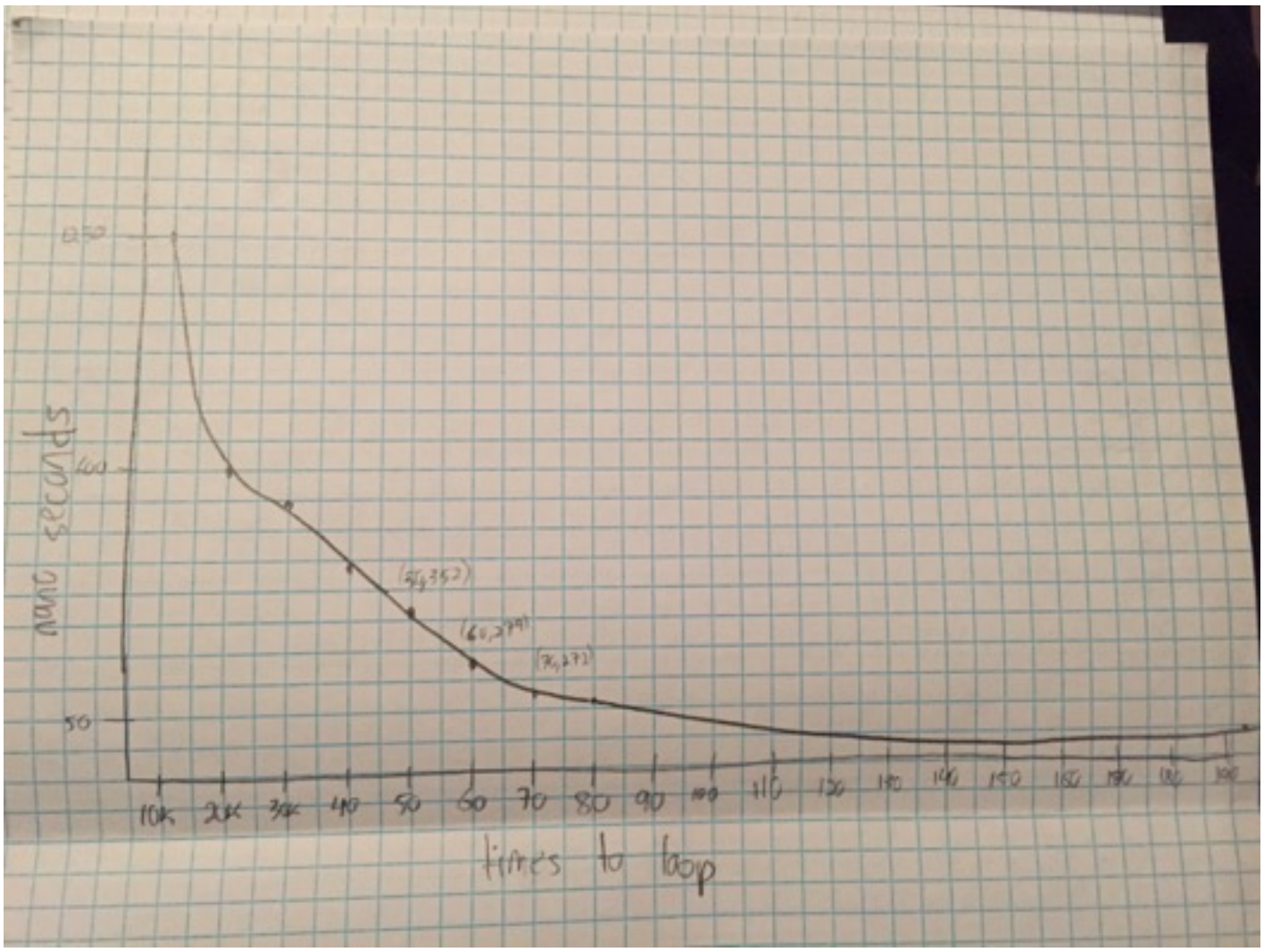
5. Finding the big O of the contains method is going to rely heavily on the fact that it is running the binary search algorithm. With this being said, I would say that the big O would be log(n). Simply because of the fact that it cuts the array in half each time it needs to run down a new sub array depending on what .compareTo() would return. It seems to be somewhat like sell sort, but instead of sorting it is searching.

Another reason I believe it to be O(log(n)), is because the array doesn't have any inversions, because we were sorting it as we inserted items. And log(n) is a fairly good complexity, which makes sense if it is running something with no inversions.

6. If you look at my attached picture of my plotted points (on the second page), you can see that the nanoseconds drop significantly with large N. It started at about 1,400 nanoseconds with small N (10,1000 loops), then dropped to about 62 nanoseconds with large N (200,000 loops). So yes this definitely looks like it backs my decision of O(log(n)).

7. Again, just like the contains method this method implements the binary search algorithm, which is O(log(n)). Although this one takes a bit longer due to everything else that is going on in the add method, such as needing to double the array, see if the object is already in the set before adding, etc.

8. All in all I would say that I worked on this assignment for 25 -28 hours.


nano seconds
250
100
50
(57,357)
(66,299)
(75,272)
10k
20k
30k
40
50
60
70
80
90
110
120
130
140
150
160
170
180
190
200
times to loop