

1.

Aaron Smith, I will submit the code.

2.

We switched roles every time one of had an idea or solution to a problem we were having, this occurred about every ten minutes. I would prefer to switch less often but that is only because I like to actually code.

3.

My programming partner was good, I plan on working with him again

4.

The iterator was most difficult for me.

5. Download [TimeArrayCollection.java](#), which is a starting point for your timing
Graphs below

6. Plot the performance of your `contains` method vs the performance of

7.

Our `toSortedList` has a complexity of N^2 because it used a `selectionSort` so it has to check every instance of N with every other instance of N , making it N^2 . Our plot appears to have exponential growth so it does support this expectation.

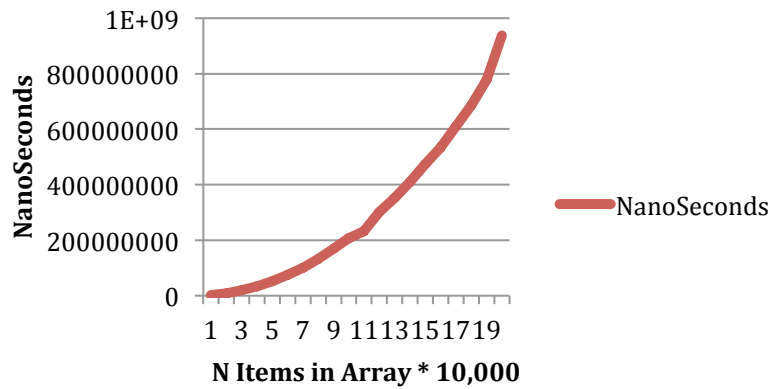
8.

Our `binarySearch` algorithm has the complexity of $\log(n)$. Our graph appears to have logarithmic traits, starts off with a decent growth rate then levels out unless you greatly increase N . On our graph its kind of hard to tell for such small values of N because increasing N by only a thousand ever time barely does anything against a logarithmic growth so it appears constant.

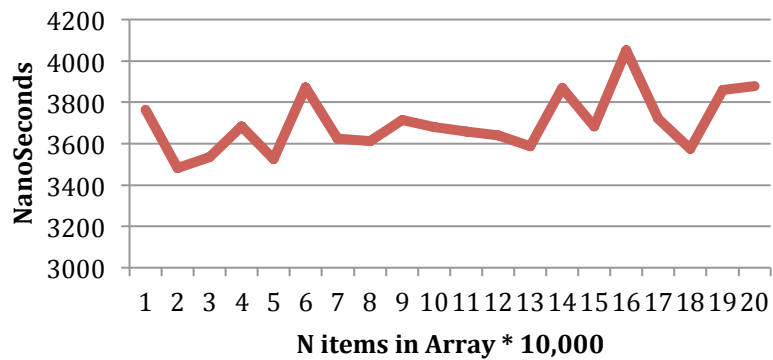
9. There are about 2^{31} possible random integer values 

The best, worse, and average case for `contains` is N . This is because we only check every item once in the array to see if it matches, if N increases the algorithm only has to check N more times. This is linear growth.

ToSorted



Binary Search



Contains

