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

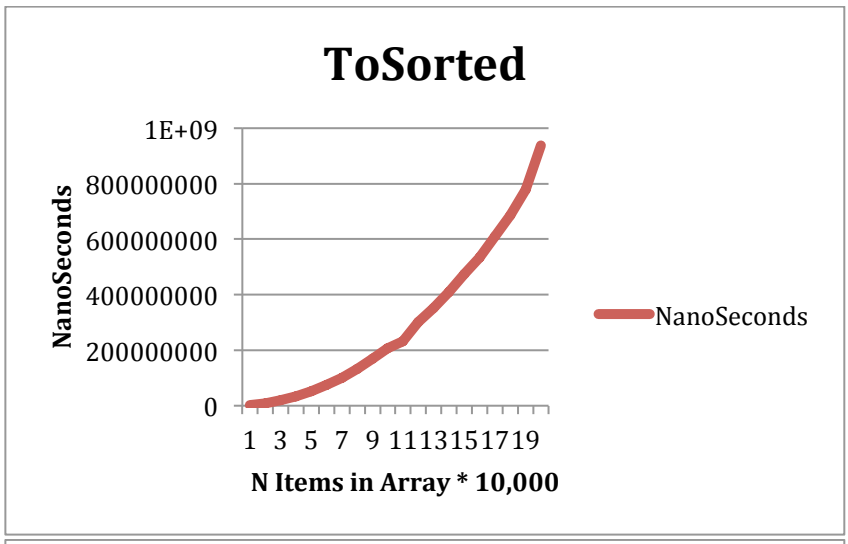
7.

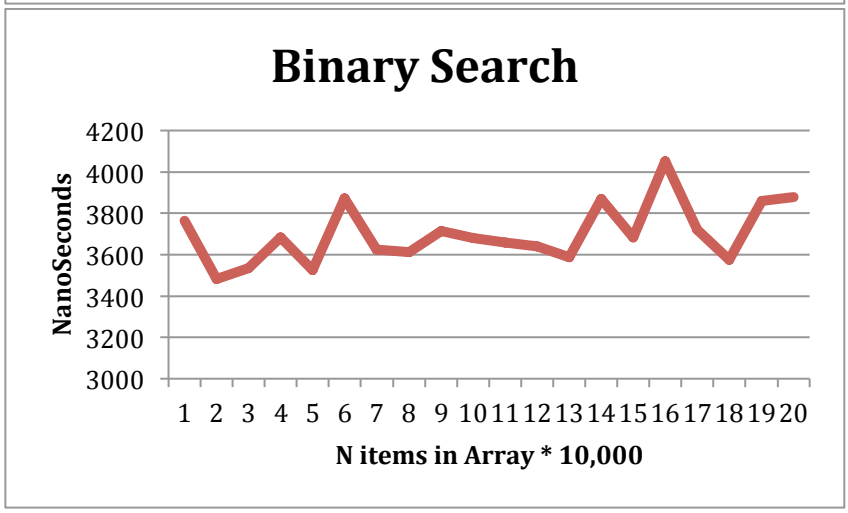
   Our toSortedList has a complexity of N^2 because it used a selectionSort so it has to check every instance of N with everyother instance of N, making it N^2. Our plot appears to have exponential growth so it does support this expectation.

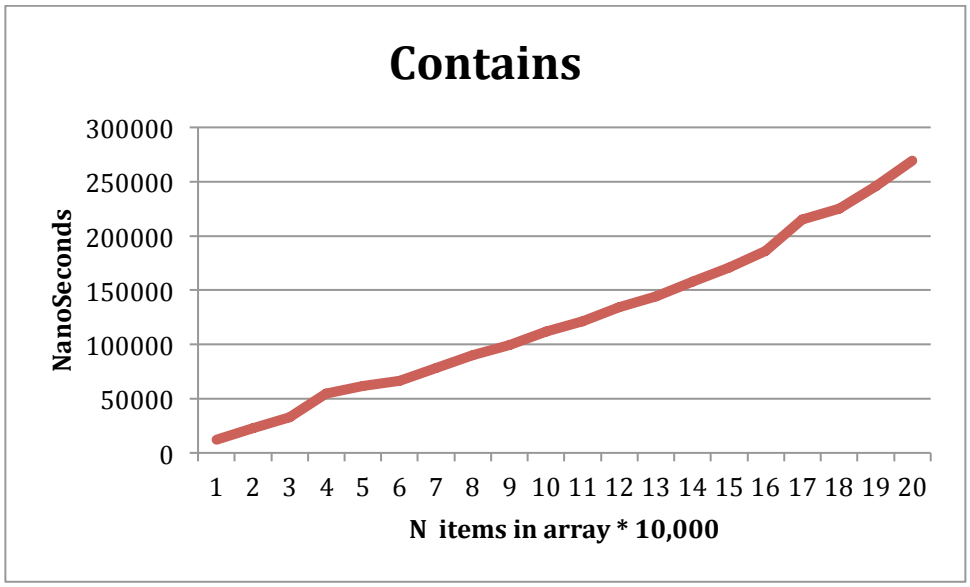
8.

   Our binarySearch algorithm has the complexity of log(n). Our graph appears to have logirthmic 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 logorithmic 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
1E+09
800000000
600000000
400000000
200000000
0
NanoSeconds
1 3 5 7 9 11 13 15 17 19
N Items in Array * 10,000
NanoSeconds


Binary Search
4200
4000
3800
3600
3400
3200
3000
NanoSeconds
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
N items in Array * 10,000


Contains
300000
250000
200000
150000
100000
50000
0
NanoSeconds
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
N items in array * 10,000