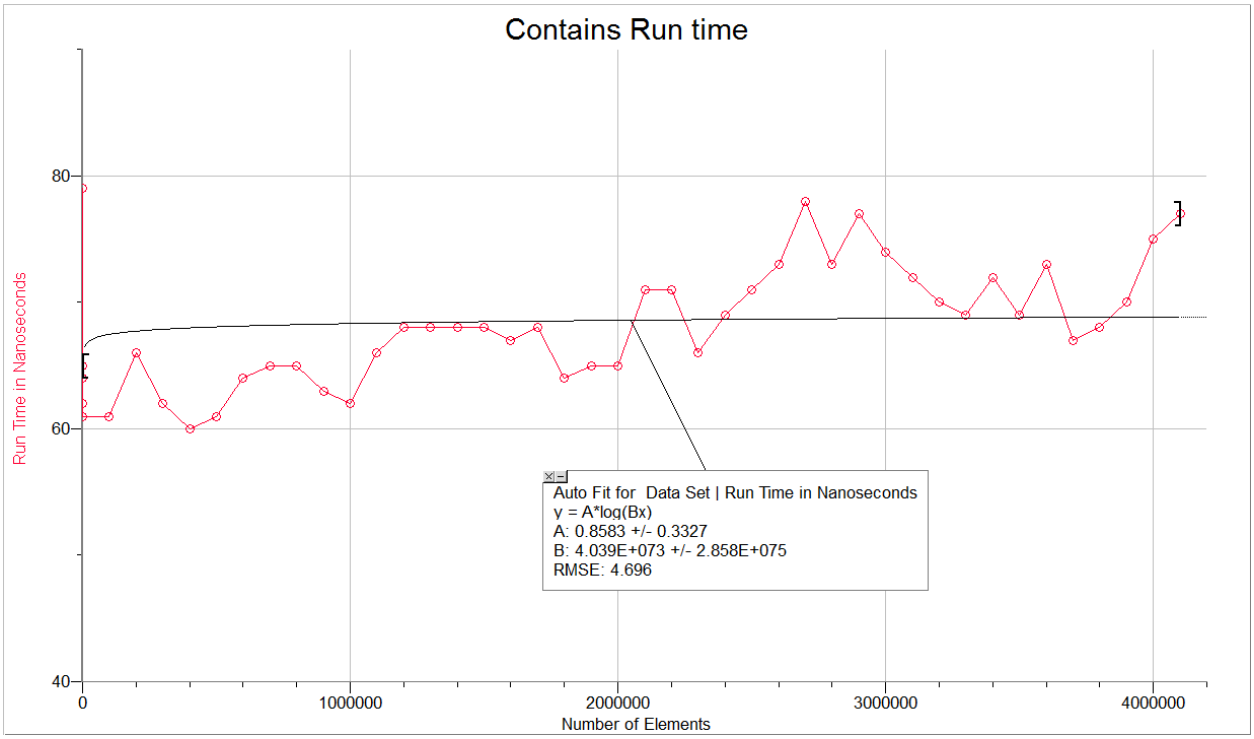
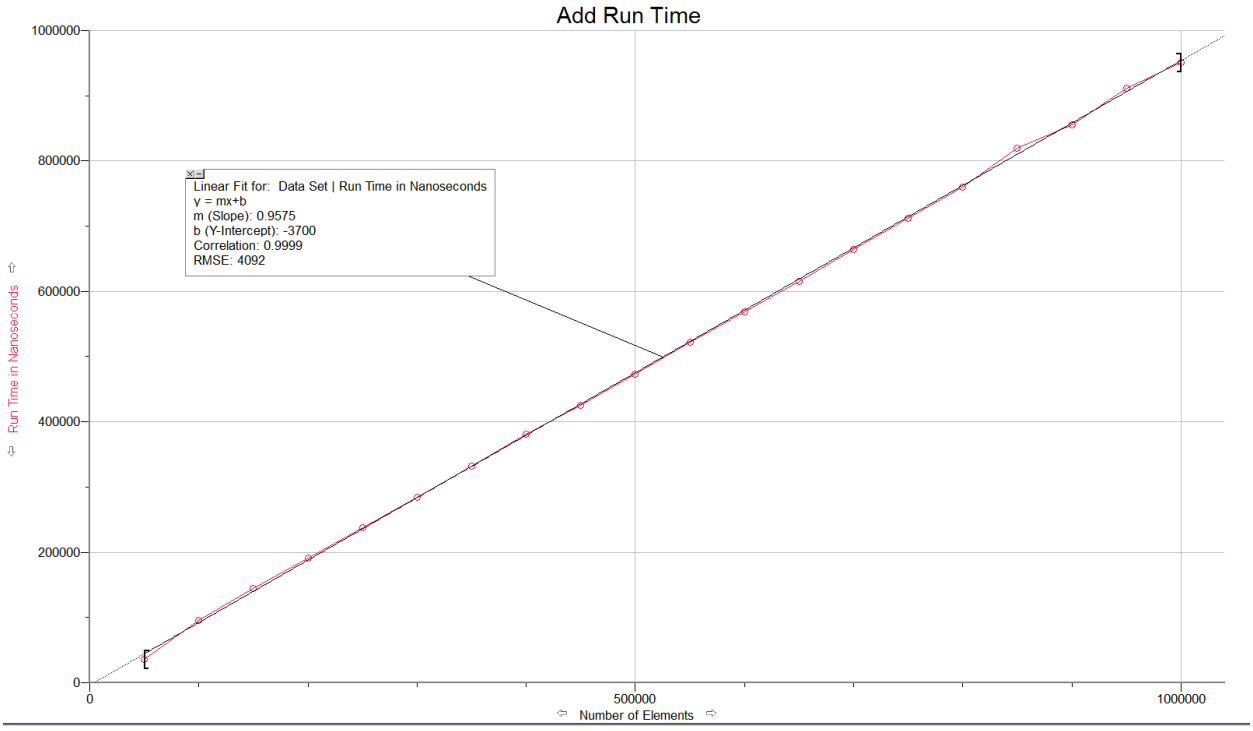
Zachary Warrell
CS 2420
Assignment 2 Analysis Document

1. I worked with Cashe Rumsey on assignment 3. I submitted the code for our program.
2. We switched off quite frequently especially when someone had a new idea to solve a problem we were stuck on. I think that we switched at the correct times I would not have wanted to be on a specific clock as that would have interrupted the driver's train of thought.
3. My programing partner was very easy to work with and extremely competent and I will definitely work with him again.
4. If the set had been implemented with a list things would have been much different. Lists are already a type of collection so it would be very easy to implement as far as coding goes. To provide an example to add something you would simply call list.add() after checking for duplicates. Similarly you would call list.remove() and list.contains() and for other functions. So as far as implementation goes it would be much easier to use a list. But as far as running time it would be less efficient. For example if an ArrayList was used then you are creating another object which essentially does the job with a few minor tweaks. You will be dealing with more fields and objects if you used a list rather than an array however implementing generics would be slightly easier. Sorting the list would be less efficient than an array as well.
5. I would expect the Big-O behavior of our contains method to be log(N)+c. This is because the operations in the contains method are all constant except for the binary search which is log(n) so doing them together would result in log(N)+c;

6. This is a graph of the run times for the contains() method over a large input range. According to the graph this looks like a constant algorithm. This is incorrect. This graph should look like a logarithmic function due to the binary search. Our contains method simply runs a binary search and returns true if the value is found and false if it is not in the set. In addition the code is surrounded in a try catch block to return false if the objects are of different types. I ran the debugger several times in many different cases and found no problems in our binary search or contains methods therefore, something in our timing code must be incorrect. My partner and I consulted with other students in the class, John Clukey during TA office hours, and people in upper level computer science classes to try and resolve this timing problem. Unfortunately no one had any suggestions saying they thought the code was correct and they didn't know what was wrong. We timed different ranges and extended the data well past the assignment requirements in an attempt to produce a log(N) shaped graph, but we had no success. We also ran it on both of our computers in case it was a problem with our machines but again no results matched a log(N) shape. We included out timing code so hopefully we can get some feedback on how to better time our algorithms in the future. We used the same technique to test the add algorithm and it provided accurate results so it remains a mystery as to what the problem is timing this specific algorithm.



7. Our add() method will take N + log(N) to execute. The N is from shifting all the elements in the array over one position. The log(N) is due to the binary search add does in order to find the correct location to insert the element. This run time is depicted by the graph below.



8. I spent about 7 to 8 hours on this assignment. In addition I spent 6 to 7 hours trying to debug my timing code. I would be very grateful for any advice you can give me about timing code in the future. My personal email is zwarrell@peakinfo.com. If you have any suggestions on how to time algorithms in the future I would greatly appreciate some suggestions.