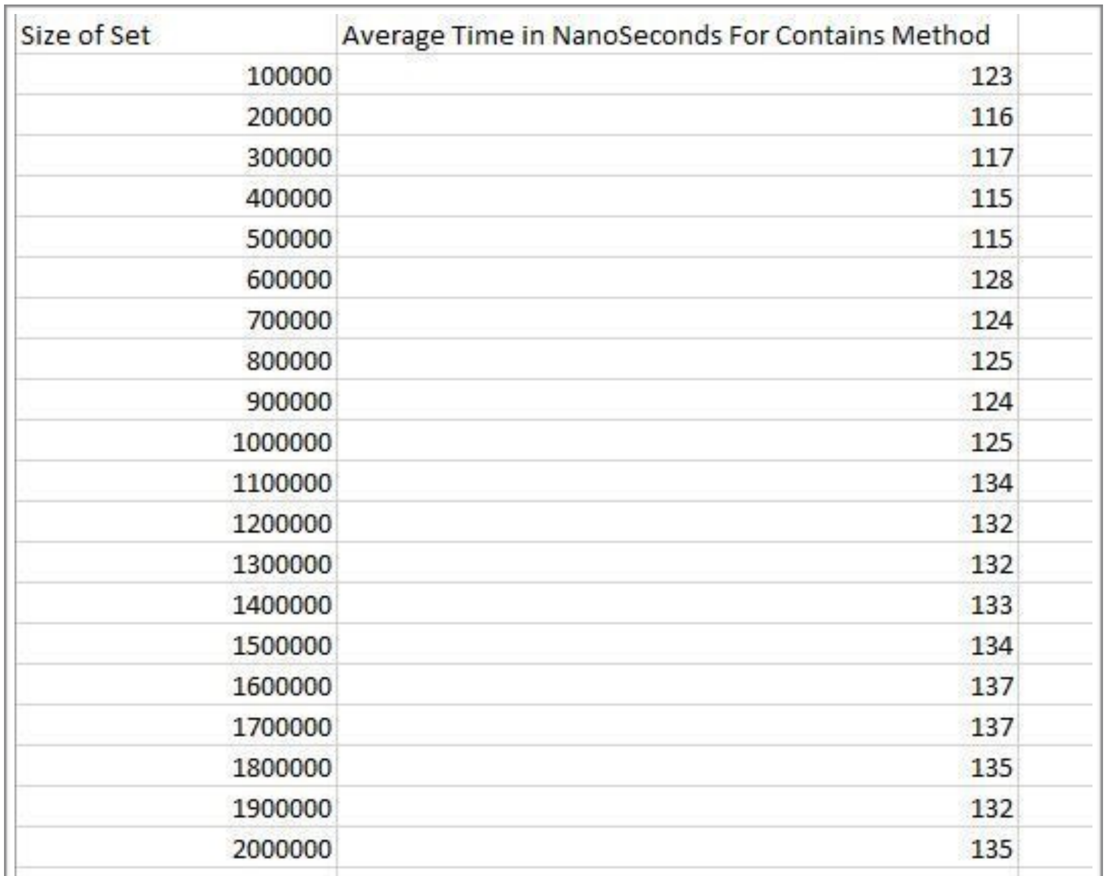
Makenzie Elliott
U0928473
Due 2/5/15 @ 11:59pm

Assignment 3 Analysis Document

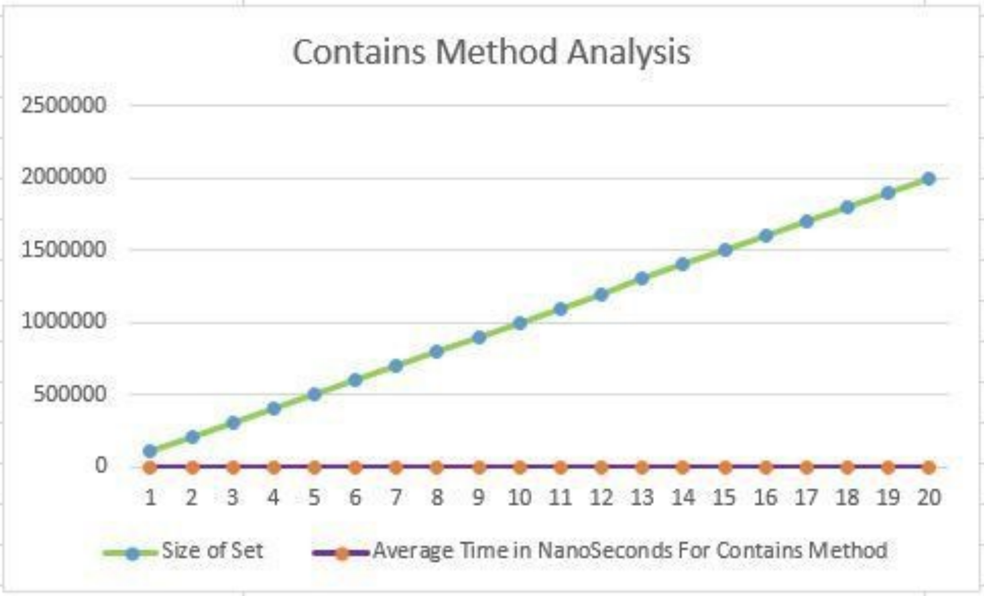
1. My programming partner was Robert Weischedel. Robert has submitted the source code for our program.
2. My partner and i switched roles constantly. If one of our ideas wasn't working out we allowed the other to take control and try something different. We spent most of our our time discussing ways to go about the problems in this assignment. I don't think switching less often wouldn't have helped. It was incredibly helpful to switch up roles and it made the assignment easier for the both of us.
3. Robert is great. He and i worked on the last assignment together and we worked well. I plan to work with him again as much as possible.
4. If the program had been backed by a Java List instead of a basic array then the implementation would have been different in many ways. If we had used an ArrayList instead we would have had access to the methods add, addAll, remove, removeAll, clear, contains, containsAll, isEmpty, lastIndexOf(similar to last), indexOf(which could be used instead of a binary search), sort(which takes in a comparator for a parameter), get(could be used for first or last), size, iterator, toString and toArray. All of these methods have been already defined for us in the ArrayList generic class. We wouldn't even have had to do a sort method with a different type of comparator because the .sort() in ArrayList already takes care of that. The sort method takes in null as a parameter if the sorting method should be the default comparable, or takes in a comparator if you want a different defined order. In addition we would no longer need the binary search method because the implementation is being taken care of with he sort method in ArrayLists as well. It would be more efficient to use the Java List because it does everything that we implemented for this assignment. The program development time would have been close to nothing because everything is already done for us. The running time for each class would be different because the ArrayList class uses the fastest algorithms whereas our program is not as advanced and takes longer to process.
5. I expect the Big-O behavior of the contains method to be of O(Log(N)) because the code is constant other than the binary search method that is used inside of the contains method. The binary search method is of type O(Log(N)) because it halves the sorted list over and over again until the correct index is found. The constant O(C) complexity from the contains method combined with the Log(N) complexity of the binary search equals cLog(N) where c is the constant and doesn't affect the run time of the Log(N) algorithm by much. The overall Big-O behavior would be O(Log(N)).

Makenzie Elliott
U0928473
Due 2/5/15 @ 11:59pm

6. For the contains method i tested sizes from 100,000 to 2,000,000 by steps of 100,000. Based upon all of the data collected the graph behaves as expected with the Big-O behavior of Log(N) discussed in question 5



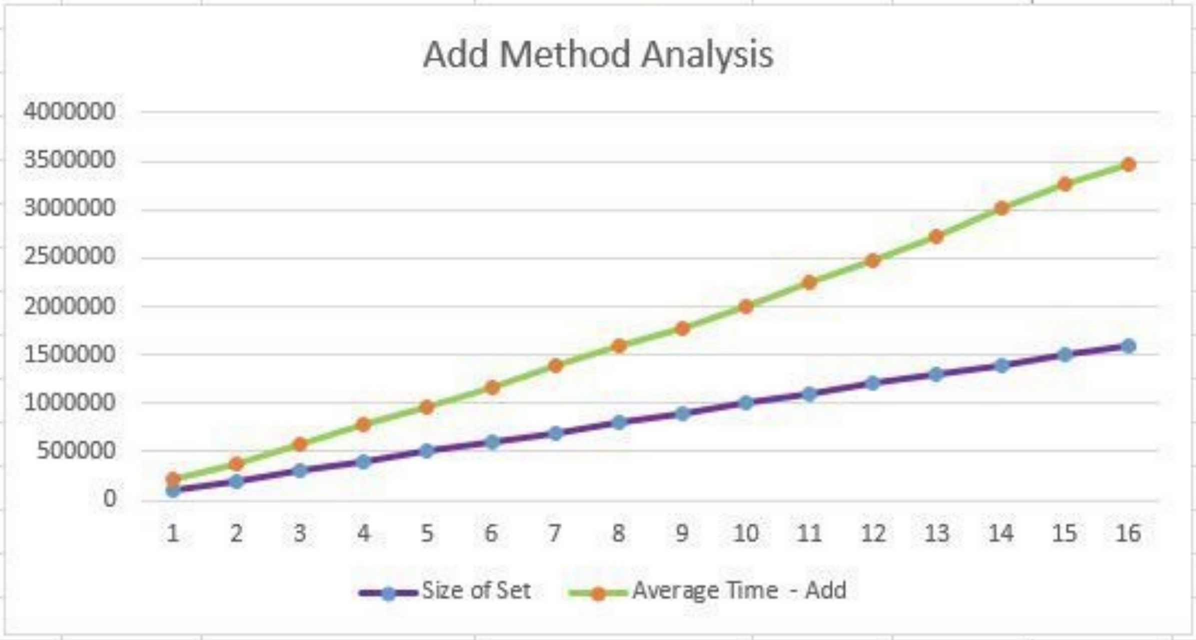
Size of Set	Average Time in NanoSeconds For Contains Method
100000	123
200000	116
300000	117
400000	115
500000	115
600000	128
700000	124
800000	125
900000	124
1000000	125
1100000	134
1200000	132
1300000	132
1400000	133
1500000	134
1600000	137
1700000	137
1800000	135
1900000	132
2000000	135



Makenzie Elliott
U0928473
Due 2/5/15 @ 11:59pm

7. For our add method the worst case in Big-O behavior is O(N), linear. I tested sizes from 100,000 to 1,600,000 by steps of 100,000.

Size of Set	Average Time - Add
100000	213527
200000	383731
300000	583094
400000	774831
500000	960890
600000	1170537
700000	1383170
800000	1599326
900000	1782641
1000000	2003060
1100000	2239309
1200000	2481992
1300000	2724132
1400000	3015361
1500000	3263557
1600000	3472463



8. Overall, Robert and i spend about 20 hours on this assignment.