Tony Chow
CS 2420
Assignment 3

Analysis

(1) My partner is Johnny Le. Johnny will be submitting the source code of our program.

(2) We switched whenever the other was tired or wanted to take over which probably average every hour and a half. I liked how often we switched because it was just when someone wanted to jump in and it was relatively laid back. I never really got burned out.

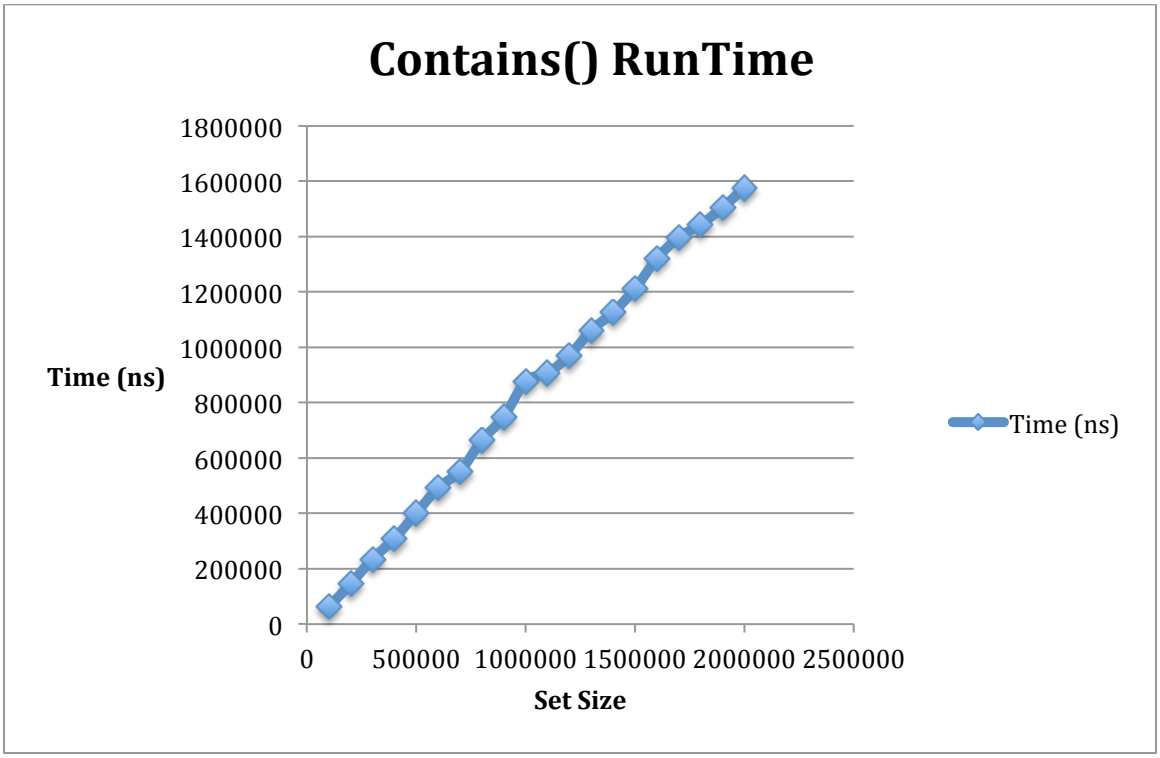
(3) I think Johnny put in a good amount of effort towards our assignment and we even stayed overnight Thursday to get our tests working. I do plan on working with him again.

(4) I think it would decrease program development time since we wouldn't have to worry about doubling or halving the array. It would just grow and shrink depending on add and remove. An ArrayList is a collection, which is what we're trying to make. Development time would definitely decrease. Runtime would be the same if they added an element to correct sorted index then it would be running a binary algorithm, which is what our code does and adds it, or removes the same way. However, I think it might slowdown if we're only adding a few elements. Suppose we want to add a single element since the array is initialized to size 8 it won't be slowed down when it doubles. I'm not sure what size the start array is for

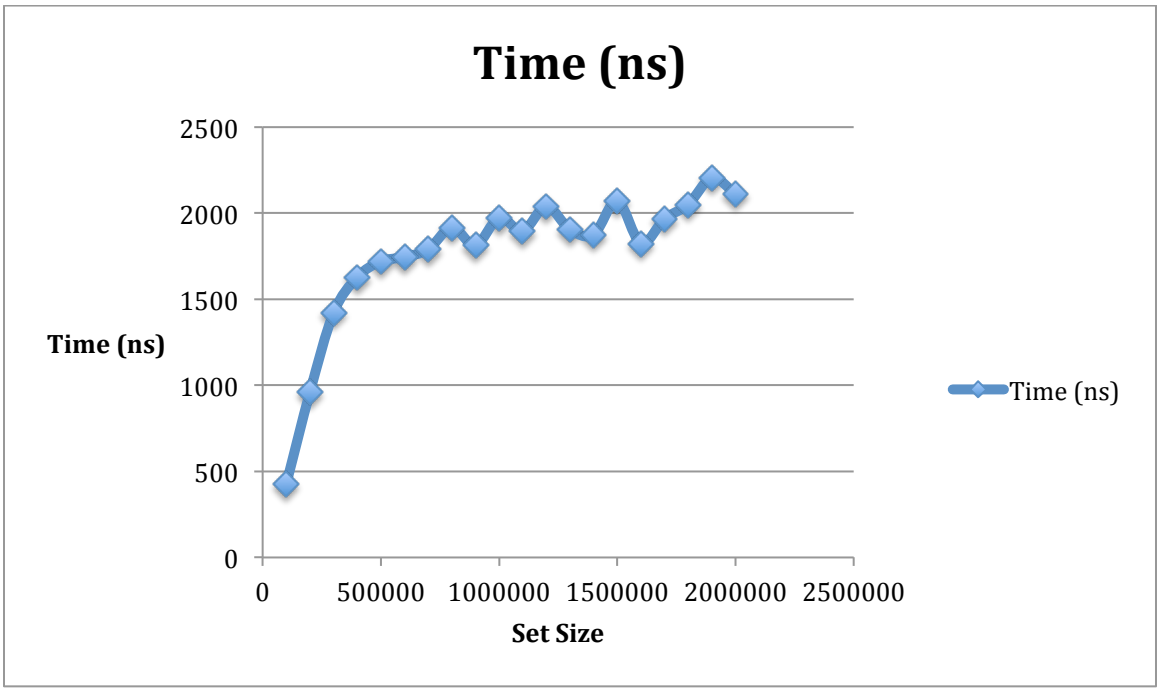
ArrayList is, but if it's one it would have to grow which would end up being slower than just an array of size 8.

(5) Our contains method calls our binarySearch method, it runs c_n, n in natural numbers a constant for initializing all the variables at the beginning. It runs another constant for seeing whether to use a comparator or comparable. Given an array of fixed size to find something it runs a searching algorithm which is a binary search. Our binary search has a Big-O behavior of approximately $\log_2(n)$.

(6) No, our expectations did not match our result for our binary search. It should have had a Big-O of O(log n), but our graph appears to be linear. It should be O(log n) because it goes to the middle then checks which side it belong on thus eliminating half of your array each time.



(7) The answer depends on the set size. We tried with sets of size 100000 to 2000000 by steps of 100000. The graph seemed to be logarithmic, reaching 80% of how long it takes on 2000000 steps at roughly set size of 400000. We did predict the add to be logarithmic because we used the binary search to find the correct index to place the element. Worse case scenario happens if the mid never hits the element till the length of each subsection is 2, so the worse case scenario is $\log_2(n)$ with O(log(n)).



(8) We spent roughly ~16 hours.